Oracle® Enterprise Manager

Extensibility Guide

10*g* Release 5 (10.2.0.5)

B40007-03

June 2011

ORACLE®

Oracle Enterprise Manager Extensibility Guide, 10*g* Release 5 (10.2.0.5)

B40007-03

Contents

3 Adding Charts

4 Validating XML

5 Adding Job Types

7 Monitoring Using Web Services and JMX

8 Defining Policies

Part II Reference

9 Management Repository Views

Index

Preface

This manual covers Enterprise Manager framework extensibility and related reference information.

Note that more recent versions of this and other Enterprise Manager books may be available on the Oracle Technology Network:

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

Audience

This guide is written for developers or administrators needing to extend Enterprise Manager's monitoring capability by defining custom target types. You should already be familiar with Enterprise Manager administrative tasks you want to perform.

You should also have a working knowledge of XML and DTDs, as well as being familiar with the operation of your specific UNIX or Windows system. Refer to your platform-specific documentation, if necessary.

Documentation Accessibility

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

Access to Oracle Support

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

Related Documents

For more information, see the following manuals in the Oracle Enterprise Manager 10*g* Release 5 documentation set:

- *Oracle Enterprise Manager Concepts*
- *Oracle Enterprise Manager Grid Control Installation and Basic Configuration*
- *Oracle Enterprise Manager Policy Reference Manual*
- *Oracle Enterprise Manager Metric Reference Manual*
- *Oracle Enterprise Manager Command Line Interface*

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.

Part I

Extensibility

This section of the guide provides intruction on extending the Enterprise Manager framework.

Part I contains the following chapters:

1 Extending Monitoring

Out-of-box, Enterprise Manager can monitor the most common hardware and applications (target types) used in enterprise environments. Because it is not possible to anticipate all possible target types that may exist in your IT environment, Enterprise Manager provides a modular way to extend monitoring capabilities called Management Plug-ins.

A Management Plug-in lets you create custom target types that allow you to monitor applications or hardware unique to your enterprise directly from the Enterprise Manager console. You simply deploy the Management Plug-in to Management Agents throughout your enterprise.

This chapter covers the following topics:

1.1 What You Get

Using Management Plug-ins to define custom target types to be monitored by Enterprise Manager, you are able to centralize all of your management information in the console. By default, Enterprise Manager management and monitoring functionality is automatically extended to target instances of the type defined by your Management Plug-in. For example:

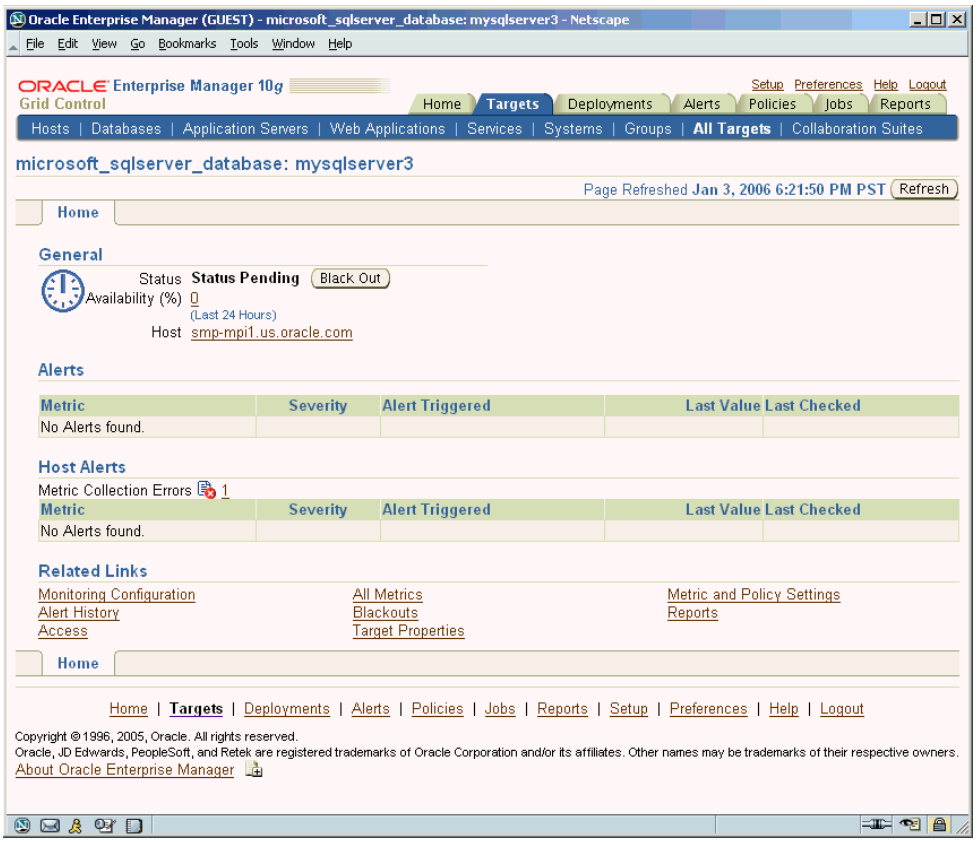
- Core framework functionality such as Alerts, Policies, Blackouts, and Management Templates, Jobs
- Groups and Systems
- Configuration Management
- SYSTEM Reports

These and other features allow you to better diagnose availability and performance problems by allowing you to correlate problems across your enterprise.

Target Home Page

Each target instance added to Enterprise Manager is automatically provided with a comprehensive target type-specific home page. No coding is required. The default target home page provides a consolidated view of target availability and alerts. Links to related monitoring information such as Monitoring Configuration, Metric and Policy Settings, and Alert History are also included. Figure 1–1 shows the default target home page for a Microsoft SQL Server database.

Figure 1–1 Microsoft SQL Server Home Page



Graphic shows the default home page for Microsoft SQL Server target types.

****

1.2 What is a Management Plug-in?

A Management Plug-in is a group of target definition files that has been added to a Management Plug-in Archive using the Enterprise Manager Command Line Interface (EM CLI). Only when the target definition files have been added to an archive are they officially a Management Plug-in. A Management Plug-in Archive file (.jar file) associates the target definition files together as a true Management Plug-in and may consist of multiple Management Plug-ins.

A Management Plug-in consists of several types of files that serve specific functions at different tiers within the Enterprise Manager framework. The target type metadata file is an integral part of defining a new target type and is required by the EM CLI, in addition to a default collection file, to create a new Management Plug-in within the Management Plug-in Archive. The target metadata and collection file, at a minimum,

are required to create a new Management Plug-in. There are, however, other supported file types that may be needed for target type-specific monitoring and reporting functions.

1.3 Management Plug-ins Available from the Oracle Technology Network

Ready-to-use Management Plug-ins are available from the Oracle Technology Network (OTN) at the Oracle Enterprise Manager 10*g* Grid Control Extensions Exchange located at:

http://www.oracle.com/technology/products/oem/extensions/index.html

This site is your central information source for all Enterprise Manager extensibility information. In addition to tutorials and the latest documentation, you can download ready-to-use Management Plug-ins developed by Oracle as well as third-party integrators.

Management Plug-ins that are currently available include:

- BEA WebLogic Plug-in
- Check Point Firewall Plug-in
- F5 BIG-IP Load Balancer Plug-in
- IBM DB2 Database Plug-in
- IBM WebSphere Plug-in
- Juniper Netscreen Firewall Plug-in
- Microsoft SQL Server Plug-in
- NetApp Filer Plug-in

Because this list is continually being updated, you should check the Extensions Exchange site regularly.

1.4 Management Plug-in Lifecycle

As with any custom code implementation, a Management Plug-in has two distinct lifecycles: Development and Deployment.

Development

The development lifecycle, which is the primary focus of this guide, consists of four stages:

1. **Design**: Determine what metrics are required to monitor your target, threshold values for alerts, and collection schedules.
2. **Develop**: Create the methods used to retrieve the metrics by defining the target type metadata and collection files, and any required support files such as monitoring scripts and/or report definition files.
3. **Validate**: Using the supplied XML validation tool (ILINT), validate the XML used to define the target metadata and collection files. Using the Management Agent's Metric Browser, you verify metric data.
4. **Package**: Create a Management Plug-in Archive using the EM CLI.

Deployment

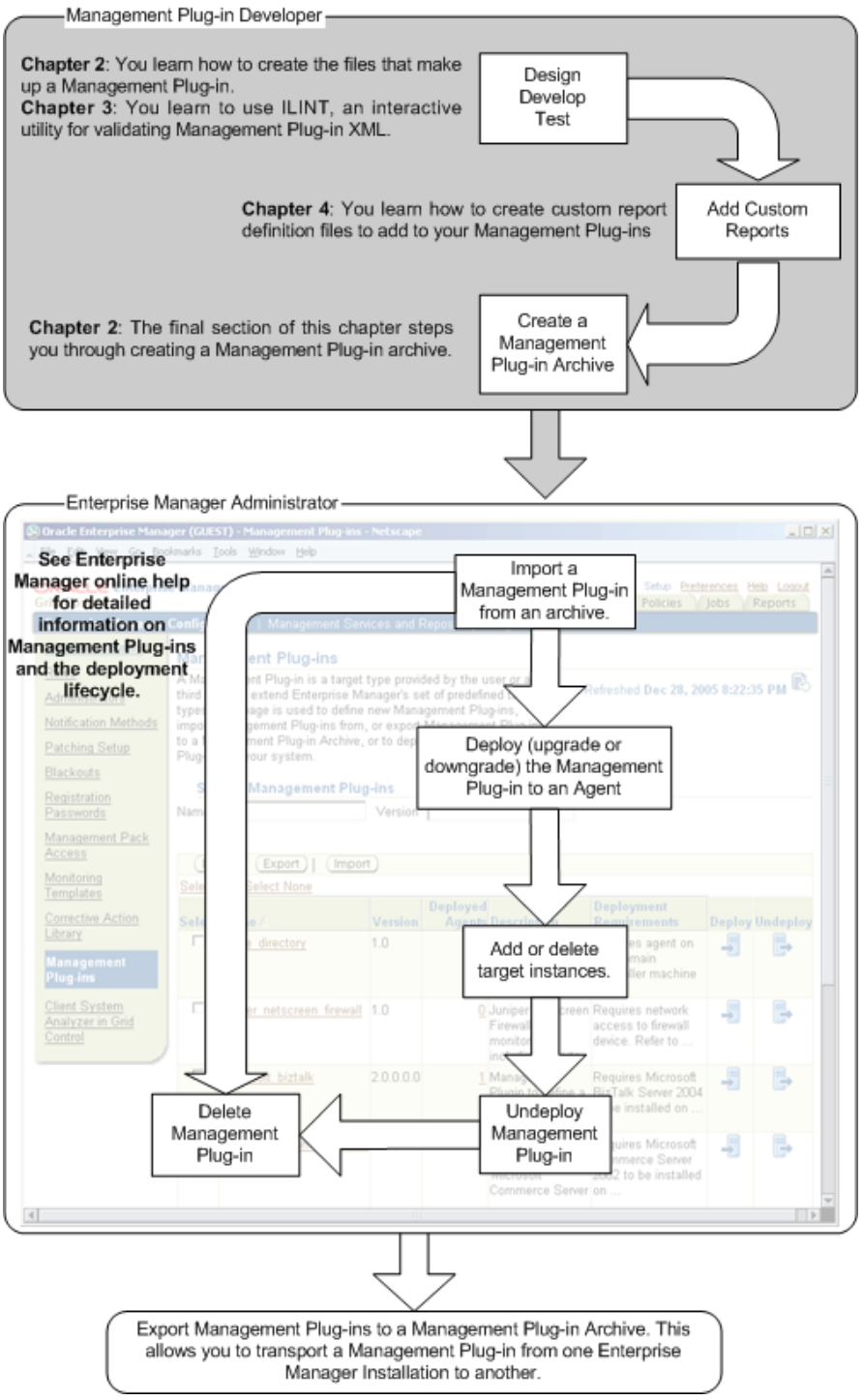
The deployment lifecycle is managed from the Enterprise Manager console and consists of the following four stages:

1. Import the Management Plug-in from the Management Plug-in Archive.
2. Deploy the Management Plug-in to one or more Management Agents.
3. Add or delete target instances defined by your Management Plug-in from the Management Agent home page(s).
4. Upgrade/Downgrade Management Plug-ins. The Management Agent uses the latest Management Plug-in version that has been deployed.

> **Note:** See Enterprise Manager online help for information on Management Plug-ins and the deployment lifecycle.

Figure 1–2 shows the complete Management Plug-in lifecycle.

Figure 1–2 Management Plug-in Lifecycle: Combined Development and Deployment



The graphic shows the combined Management Plug-in development and deployment lifecycle.

****

1.5 About the Extensibility Guide

The Extensibility guide assumes you have sufficient familiarity with the Enterprise Manager Grid Control framework, its components, and functionality. In addition, familiarity with XML and PL/SQL is assumed. For more information about Enterprise Manager Grid Control, see the Enterprise Manager Concepts guide and the Enterprise Manager Grid Control online help. Additional information is available from the Enterprise Manager documentation area on OTN as well as the Extensions Exchange.

HTTP://www.oracle.com/technology/products/oem/extensions/index.html

The Extensibility guide is divided into two sections. The first section provides detailed instructions for creating your Management Plug-in and the second provides reference information that may be required when developing your Management Plug-in. The remaining chapters of this guide consist of the following:

- **Chapter 2**: Instructions on how to define the required target metadata and collection files. Using the Management Agent's Metric Browser is also covered in this chapter.
- **Chapter 3**: Using ILINT to validate the XML used to create your target metadata and collection files.
- **Chapter 4**: Instructions on how to create a report definition file for inclusion in your Management Plug-in. The complete Information Publisher PL/SQL API is documented in this chapter.
- **Chapter 5**: Instructions on using the OJMX fetchlet to monitor targets exposing a Web Service managment interface.
- **Chapter 6**: Reference for all management views used to access information stored in Enterprise Manager's Management Repository.
- **Chapter 7**: Reference for available fetchlets (metric data extraction mechanisms).

2

Developing a Management Plug-in

Each Management Plug-in defines a new type of target that can be monitored by Enterprise Manager. A target, or more specifically, a target instance, can be defined as any monitorable entity within an enterprise. A monitorable entity can be an application running on a server, the server itself, the network or any of its constituent parts. Enterprise Manager makes managing target instances simple by allowing you to add new target instances to the management framework from the Enterprise Manager Grid Control console, at which point you can take advantage of Enterprise Manager's monitoring and administrative functionality.

Enterprise Manager relies on a series of XML files that define a new target type to the system. A Management Plug-in consists of all files required to monitor a specific target from within Enterprise Manager.

Important: Much of Enterprise Manager is built around XML files. For this reason, it is essential that you are familiar with XML and related technologies. **This chapter assumes that you have a working knowledge of XML and DTDs.**

- **XML** (eXtensible Markup Language) - An XML document is a structured text document that conveys a particular kind of information.
- **DTD** (Document Type Declaration) - A DTD provides the grammar for the XML files, thus describing what content is expected in each of its related XML files. When creating a new XML file, you need to carefully study its DTD to understand what content needs to be present in that file.

Once a Management Plug-in has been defined, you add it to a Management Plug-in Archive that is imported into Enterprise Manager. An archive encapsulates the one or more Management Plug-ins within a single jar file.

Creating a Management Plug-in consists of five stages:

1. Designing your Management Plug-in.
2. Developing the requisite target type definition files that make up the Management Plug-in.
3. Validating the target type definition.
4. Using the Enterprise Manager Command Line Interface (EM CLI), Package the target type definition files into a portable Management Plug-in Archive file.

5. Deploy the Management Plug-in throughout your Enterprise Manager environment.

2.1 Designing Your Management Plug-in

Before creating actual plug-in files, you need to define what parameters of the target type are required to accurately monitor and manage your new component. This involves:

- Identifying performance and configuration metrics that should be collected.
- Determining how often each metric should be collected. Oracle recommends that the collection frequency for any metric should not be less than once every five minutes.
- Based on customer-specific operational practices, specifying default warnings and/or critical thresholds on these metrics. Whenever a threshold is crossed, Enterprise Manager generates an alert, informing administrators of potential problems.

2.2 Developing Requisite Management Plug-in Files

Once you have determined how a particular type of target should be monitored, you are ready to begin creating the Management Plug-in files that define this particular target type to Enterprise Manager. To create a target type is to define to Enterprise Manager not only what a target is, but more importantly what should be monitored (metrics), and how monitoring should be performed.

File Locations

The following table lists the key files used by Enterprise Manager for system configuration and extensibility. When a Management Plug-in is deployed to an Agent, the requisite files are copied to their respective directory locations.

Table 2–1 Configuration File Locations

Name	Directory Location	Purpose
target_type.xml (Required to define a target type.)	AGENT_HOME/sysman/admin/metadata/	Defines metrics to be collected/computed for the target type.
target_collections.xml (Required to define a target type.)	AGENT_HOME/sysman/admin/default_collection/	Defines metric collection parameters such as metric data collection intervals and default alert thresholds. Typically, the name of the metadata (target_type.xml) file and the default collection file are identical.
targets.xml	AGENT_HOME/sysman/emd/	Lists user-added target instances added to the Enterprise Manager framework. This file is generated and updated by Enterprise Manager and/or the Target Instance Installer utility. This file should not be edited manually.

DTD files associated with the XML configuration files can be found at the locations specified in Table 2–2.

Table 2–2 Document Type Declaration Files

Name	Directory Location	Purpose
TargetMetadata.dtd	AGENT_HOME/sysman/admin/dtds/	Documentation Type Declaration (DTD) file associated with the target_type.xml file.
TargetInstance.dtd	AGENT_HOME/sysman/admin/dtds/	DTD associated with the targets.xml file.
TargetCollection.dtd	AGENT_HOME/sysman/admin/dtds/	DTD associated with the target_ collections.xml files.

When you deploy a Management Plug-in, Enterprise Manager automatically copies all plug-in files to the appropriate locations. Conversely, when you are ready to update or remove a Management Plug-in, appropriate files are replaced in the case of a plug-in upgrade, or removed if a plug-in is undeployed.

Important: The Management Plug-in deployment mechanism does not permit overwriting of Oracle-supplied metadata files.

2.2.1 Target Definition Files

Adding a target type to the Enterprise Manager framework makes it possible for any instance of that target type to be monitored by an Oracle Management Agent once the target instance has been added to the framework from the Enterprise Manager Grid Control console. Every target type within the Enterprise Manager Grid Control framework must be unique. Defining a target type and making this information available to an Oracle Management Agent involves creating two XML files:

Target Type Metadata File

The definition of a target type is composed primarily of the metrics you wish to expose. The file contains a list of all metrics collected for a particular target type, along with specifics on how to compute each metric.

Target Type Default Collections File

Whether you want to regularly upload metric values to the Management Repository or check these values against specific conditions, you will need to define default collection intervals for each of the target metrics. This is also the file where you specify alert thresholds for each metric, enabling events to be triggered when violations occur. Users will be allowed to override the defaults, but the original collections file must be provided by the target provider.

The following steps take you through the process of creating the target type metadata and collection files, registering them with the Enterprise Manager framework, and finally adding instances of the new target type to the Enterprise Manager Grid Control console for monitoring.

2.2.1.1 Creating the target type metadata file.

Target type metadata consists of the metrics you want to expose and the methods used to retrieve and compute those metrics. The target type metadata file tells the Oracle Management Agent what data to retrieve and how to obtain that data for this particular target type. Users can add a new target type by creating a new target metadata XML file and placing it under $AGENT_ HOME/sysman/admin/metadata/.

Note: Manually placing the target metadata XML file in the AGENT_HOME metadata directory is for development purposes only. The target metadata file is automatically placed in the metadata directory when the Management Plug-in is deployed to the Agent.

Target Type Metadata File Naming Conventions

Although you can specify any name for a target type metadata file, Oracle recommends that users adding new target types adhere to Enterprise Manager naming conventions. This naming convention allows for file naming consistency in environments where similar products from multiple vendors are used. The target naming convention follows the form <vendor>_<product category>. For example:

```
cisco_router.xml
oracle_apache.xml
bea_http.xml
cisco_slb.xml
f5_slb.xml
nortel_slb.xml
```

To account for cases where one product category contains multiple product offerings from a single vendor, Oracle recommends that you use the following file naming convention:

```
<vendor>_<category>_<brand name>
```

For example:

```
f5_slb_bigip.xml
f5_slb_3dns.xml
```

Enterprise Manager ships with a wide array of predefined target type metadata files that cover the most common target types. In situations where the predefined target metadata files do not fit exactly the types of targets you wish to monitor, you can either create a new target metadata file from scratch, or use one of the predefined metadata files as a template for defining a new target type, and then repackage them as a new Management Plug-in. The pre-defined metadata files are located in the $AGENT_HOME/sysman/admin/metadata directory.

Anatomy of a Target Type Metadata File

At the highest definition level, the target type metadata file is composed of four primary XML elements:

- TargetMetadata (top level)
  - Display (Oracle Internal Use)
  - Metric
  - InstanceProperties

Although this section briefly introduces XML elements and some of tag options used to define target metadata, for explicit definitions and other important usage information, see the TargetMetadata.dtd file located in the $AGENT_HOME/sysman/admin/dtds directory.

The following example shows the primary XML elements of the Net App Filer target type metadata file.

Example 2–1 Target Type Metadata File

```
<?xml version="1.0" ?>
<!DOCTYPE TargetMetadata SYSTEM "../dtds/TargetMetadata.dtd">

<TargetMetadata META_VER="2.2" TYPE="netapp_filer" CATEGORY_
PROPERTIES="IsClusterEnabled;IsSnapLicensed;Filer_Version">

<Display>
  <Label NLSID="netapp_filer">Network Appliance Filer</Label>
</Display>

<!-- ************************************************************************
********* -->

  <Metric NAME="Product" TYPE="TABLE">
    <Display>
      <Label NLSID="netapp_filer_info">Product Information</Label>
    </Display>
    <TableDescriptor>
      <ColumnDescriptor NAME="productCategory" TYPE="STRING">
        <Display>
         <Label NLSID="netapp_filer_product_category">Product Category</Label>
        </Display>
      </ColumnDescriptor>
      <ColumnDescriptor NAME="prodVersion" TYPE="STRING">
        <Display>
          <Label NLSID="netapp_filer_product_version">Version</Label>
        </Display>
      </ColumnDescriptor>
      <ColumnDescriptor NAME="prodId" TYPE="STRING">
        <Display>
          <Label NLSID="netapp_filer_product_id">Product ID</Label>
        </Display>
      </ColumnDescriptor>
      <ColumnDescriptor NAME="prodVendor" TYPE="STRING">
        <Display>
          <Label NLSID="netapp_filer_product_vendor">Vendor</Label>
        </Display>
      </ColumnDescriptor>
      <ColumnDescriptor NAME="prodModel" TYPE="STRING">
        <Display>
          <Label NLSID="netapp_filer_product_model">Model</Label>
        </Display>
      </ColumnDescriptor>
         <ColumnDescriptor NAME="firmwareVersion" TYPE="STRING">
            <Display>
               <Label NLSID="netapp_filer_firmware_version">Firmware
Version</Label>
            </Display>
      </ColumnDescriptor>
      <ColumnDescriptor NAME="prodVersionBucket" TYPE="STRING"  COMPUTE_
EXPR="(prodVersion __contains 'NetApp Release 7.') ? '7.X':'6.X'"
TRANSIENT="TRUE">
        <Display>
          <Label NLSID="netapp_filer_prodVersionBucket">Product Version
Bucket</Label>
        </Display>
      </ColumnDescriptor>
    </TableDescriptor>
    <QueryDescriptor FETCHLET_ID="Snmp">
```

```
      <Property NAME="NAME" SCOPE="INSTANCE">NAME</Property>
      <Property NAME="hostname" SCOPE="INSTANCE">AgentHost</Property>
      <Property NAME="COMMUNITY" SCOPE="INSTANCE"
OPTIONAL="TRUE">CommunityString</Property>
      <Property NAME="TIMEOUT" SCOPE="INSTANCE" OPTIONAL="TRUE">Timeout</Property>
      <Property NAME="TABLE"    SCOPE="GLOBAL">FALSE</Property>
      <Property NAME="OIDS"     SCOPE="GLOBAL">1.3.6.1.2.1.1.2.0
                                               1.3.6.1.4.1.789.1.1.2.0
                                               1.3.6.1.4.1.789.1.1.3.0
                                               1.3.6.1.4.1.789.1.1.4.0
                                               1.3.6.1.4.1.789.1.1.5.0
                                               1.3.6.1.4.1.789.1.1.6.0
      </Property>
    </QueryDescriptor>
  </Metric>
.
.
.
<InstanceProperties>

   <DynamicProperties NAME="ClusterConfig" FORMAT="ROW"
                      PROP_LIST="IsClusterEnabled">
     <QueryDescriptor FETCHLET_ID="Snmp">
       <Property NAME="NAME" SCOPE="INSTANCE">NAME</Property>
       <Property NAME="hostname" SCOPE="INSTANCE">AgentHost</Property>
       <Property NAME="COMMUNITY" SCOPE="INSTANCE"
OPTIONAL="TRUE">CommunityString</Property>
       <Property NAME="TIMEOUT" SCOPE="INSTANCE"
OPTIONAL="TRUE">Timeout</Property>
       <Property NAME="TABLE"    SCOPE="GLOBAL">FALSE</Property>
       <Property NAME="OIDS"
SCOPE="GLOBAL">1.3.6.1.4.1.789.1.2.3.1.0</Property>
     </QueryDescriptor>
   </DynamicProperties>

   <DynamicProperties NAME="VersionConfig" FORMAT="ROW"
                      PROP_LIST="Filer_Version">
    <ExecutionDescriptor>
       <GetTable NAME="Product"/>
       <GetView NAME="Config" FROM_TABLE="Product">
          <Column NAME="prodVersionBucket"/>
        </GetView>
     </ExecutionDescriptor>
   </DynamicProperties>

   <DynamicProperties NAME="SnapMirrorConfig" FORMAT="ROW"
                      PROP_LIST="IsSnapLicensed">
     <QueryDescriptor FETCHLET_ID="Snmp">
       <Property NAME="NAME" SCOPE="INSTANCE">NAME</Property>
       <Property NAME="hostname" SCOPE="INSTANCE">AgentHost</Property>
       <Property NAME="COMMUNITY" SCOPE="INSTANCE"
OPTIONAL="TRUE">CommunityString</Property>
       <Property NAME="TIMEOUT" SCOPE="INSTANCE"
OPTIONAL="TRUE">Timeout</Property>
       <Property NAME="TABLE"    SCOPE="GLOBAL">FALSE</Property>
       <Property NAME="OIDS"
SCOPE="GLOBAL">1.3.6.1.4.1.789.1.9.19.0</Property>
     </QueryDescriptor>
   </DynamicProperties>
```

```
    <InstanceProperty NAME="AgentHost" CREDENTIAL="FALSE" OPTIONAL="FALSE">
      <Display>
        <Label NLSID="netapp_filer_agent_host">Hostname or IP Address</Label>
      </Display>
    </InstanceProperty>
    <InstanceProperty NAME="CommunityString" CREDENTIAL="TRUE" OPTIONAL="TRUE">
      <Display>
        <Label NLSID="netapp_filer_community_string">SNMP Read Community String
(Default: public)</Label>
      </Display>
      public
    </InstanceProperty>
    <InstanceProperty NAME="Timeout" CREDENTIAL="FALSE" OPTIONAL="TRUE">
      <Display>
        <Label NLSID="netapp_filer_timeout">SNMP Timeout (Default: 5
seconds)</Label>
      </Display>
      5
    </InstanceProperty>
.
.
.
</InstanceProperties>
</TargetMetadata>
```

From the target metadata file excerpt, you can see that the file consists of the following functional areas where you define your target metadata:

- TargetMetadata and Display
- Metric Name
- Instance Properties

TargetMetadata and Display

The first lines after the header of the target definition file identify the target type. The following example is an excerpt from the Net App Filer target definition file.

Example 2–2 TargetMetadata and Display XML Elements

```
<?xml version="1.0" ?>
<!DOCTYPE TargetMetadata SYSTEM "../dtds/TargetMetadata.dtd">
<TargetMetadata META_VER="2.2" TYPE="netapp_filer" CATEGORY_
PROPERTIES="IsClusterEnabled;IsSnapLicensed;Filer_Version">
<Display>
  <Label NLSID="netapp_filer">Network Appliance Filer</Label>
</Display>
```

These lines define the basic specifications: metadata version (META_VER="2.2"), target type (TYPE="netapp_filer"), the target's NLS Identifier (NLSID="netapp_filer") and display name text ("Network Appliance Filer"). Metadata versioning allows different versions of the same target type metadata to exist concurrently within the managed environment (one metadata version per Management Agent).You should change the metadata version each time you update the target metadata file. Typically a target type consists of the company name followed by the product name. The "Display" element is used internally by Oracle for translation purposes and is not required when defining new target types. "Display" elements are used with various elements in the target metadata for both readability and internationalization.

Metric NAME

The content of the target type metadata file is devoted primarily to metric definitions. As a matter of practice, Oracle recommends that you specify at least two metrics for any target type:

- Metric to monitor target availability (Required for all target types.)
- Metric to monitor target performance (Optional but recommended.)

For target availability to show up correctly on the default target home page, Oracle requires the target metadata file to define a metric with NAME="Response" that contains a column with NAME="Status" and the default collection file must define a critical condition on the "Status" column that represents the target being up or down.

Shown in the example below, the Response metric, which monitors target availability, is required for all target types. The Load metric is used to determine a target's performance level.

The following example shows the "Response" metric definition for a Network Appliance Filer target type.

Example 2–3 Response Metric

```
   <Metric NAME="Response" TYPE="TABLE">
     <Display>
       <Label NLSID="netapp_filer_resp">Response</Label>
     </Display>
     <TableDescriptor>

       <ColumnDescriptor NAME="Status" TYPE="NUMBER">
         <Display>
           <Label NLSID="netapp_filer_resp_status">Status</Label>
         </Display>
       </ColumnDescriptor>

     </TableDescriptor>
     <QueryDescriptor FETCHLET_ID="Snmp">
       <Property NAME="NAME" SCOPE="INSTANCE">NAME</Property>
       <Property NAME="hostname" SCOPE="INSTANCE">AgentHost</Property>
       <Property NAME="COMMUNITY" SCOPE="INSTANCE"
OPTIONAL="TRUE">CommunityString</Property>
       <Property NAME="TIMEOUT" SCOPE="INSTANCE"
OPTIONAL="TRUE">Timeout</Property>
       <Property NAME="PINGMODE" SCOPE="GLOBAL">TRUE</Property>
       <Property NAME="OIDS" SCOPE="GLOBAL">1.3.6.1.4.1.789.1.1.3.0</Property>
     </QueryDescriptor>
   </Metric>
```

Because the "Response" metric is of type "TABLE", it requires a "TableDescriptor" and associated "ColumnDescriptor". Note that all levels of the metric, which includes table and column definitions, are given names, types, and display labels. As mentioned earlier, these elements and usage instructions can be found in the TargetMetadata.dtd file.

For each metric, you must also specify how to obtain the data to fill this structure. This is accomplished using fetchlets—mechanisms that take parameters, query a specified target, and then return some values from the target. Enterprise Manager provides a wide array of fetchlets to meet most data retrieval needs. See Chapter 10, "Fetchlets" and Chapter 7, "Monitoring Using Web Services and JMX" for more information on fetchlets and available fetchlet types.

A fetchlet associated with the metric is declared through the "QueryDescriptor" tag. The following example shows the QueryDescriptor used for the Microsoft SQL Server Response metric.

Example 2–4 Query Descriptor Usage

```
<QueryDescriptor FETCHLET_ID="OSLineToken">
<Property NAME="emdRoot" SCOPE="SYSTEMGLOBAL">emdRoot</Property>
<Property NAME="command" SCOPE="GLOBAL">%emdRoot%/bin/nmefwmi</Property>
<Property NAME="ENVWBEM_WQL" SCOPE="GLOBAL">select name,pathname,processid,state
from win32_service where pathname like '%sqlservr.exe%' or pathname like
'%sqlagent%'</Property>
<Property NAME="ENVWBEM_WQL_COLUMN_ORDER"
SCOPE="GLOBAL">name,pathname,processid,state</Property>
<Property NAME="startsWith" SCOPE="GLOBAL">em_result=</Property>
<Property NAME="delimiter" SCOPE="GLOBAL">|</Property>
<Property NAME="NEED_CONDITION_CONTEXT" SCOPE="GLOBAL">TRUE</Property>
</QueryDescriptor>
```

The "FETCHLET_ID" label is used to specify which fetchlet to use for data retrieval. In the example above, the "Response" metric uses the "OSLineToken" fetchlet. What follows are the definition of required system parameters used by the fetchlet.

The "oraTclHome", "perlBin", and "scriptsDir" properties enable the correct bindings in the "command" property. The remaining properties are "OSLineToken" fetchlet input parameters. See Chapter 10, "Fetchlets" for information on parameters used by each fetchlet.

The "scope" property defines where the property value is to be obtained. The following scope options are available:

- SCOPE="GLOBAL" Obtain the property value from other variables defined within the current target type metadata file. This includes constants, such as the "|" shown in the QueryDescriptor example.
- SCOPE="INSTANCE" Obtain the property value from instance properties.
- **SCOPE="ENVxxx"** Obtain the property value from an environment variable "xxx".
- **SCOPE="SYSTEMGLOBAL"** Obtain the property value from the emd.properties file located in the $AGENT_HOME/sysman/config directory.
- **SCOPE="USER"** Obtain the property value from the collector or user.

Instance Properties

The "InstanceProperties" descriptor is a required top-level specification that defines what properties an administrator must specify in the Enterprise Manager Grid Control console when adding a new target instance of this particular target type.

Although the InstanceProperties section can be defined at various locations within the target type metadata file, Oracle recommends defining this section at the very end of the file for consistency. Instance properties defined in the target type metadata file are resolved to values specified for these instance properties in the targets.xml file. Shown below are some of the instance properties for a Microsoft SQL Server target type.

Example 2–5 Instance Properties

```
<InstanceProperties>
<InstanceProperty NAME="ms_sqlserver_host" CREDENTIAL="FALSE" OPTIONAL="FALSE">
<Display>
```

```
<Label NLSID="ms_sqlserver_host">SQL Server host</Label>
</Display>
</InstanceProperty>
<InstanceProperty NAME="ms_sqlserver_servername" CREDENTIAL="FALSE"
OPTIONAL="FALSE">
<Display>
<Label NLSID="ms_sqlserver_servername">SQL Server name</Label>
</Display>
</InstanceProperty>
<InstanceProperty NAME="url" CREDENTIAL="FALSE" OPTIONAL="FALSE">
<Display>
<Label NLSID="connectString_iprop">JDBC URL</Label>
</Display>
</InstanceProperty>
.
.
.
</InstanceProperties>
```

As an example, when an Oracle Database Instance target type is added to the Enterprise Manager framework, specific information about that target instance is added to the targets.xml file. In the example below, we can see that the InstanceProperty labels such as OracleHome, Port, SID defined in the Oracle Database Instance target type metadata file are resolved to instance-specific values in the targets.xml file.

Example 2–6 Target Instance Information for an Oracle Database

```
<Target TYPE="oracle_database" NAME="emrep.us.oracle.com">
<Property NAME="OracleHome" VALUE="/scratch/OracleHomes/db10g"/>
<Property NAME="UserName" VALUE="f8a6d72528517730" ENCRYPTED="TRUE"/>
<Property NAME="MachineName" VALUE="stacb11.us.oracle.com"/>
<Property NAME="Port" VALUE="1521"/>
<Property NAME="SID" VALUE="emrep"/>
<Property NAME="ServiceName" VALUE="emrep.us.oracle.com"/>
<Property NAME="password" VALUE="7536910dcf78eaad" ENCRYPTED="TRUE"/>
</Target>
```

2.2.1.2 Validate your new target type definitions.

It is always a good idea to test your new target type definitions using the metric browser. The metric browser is a development utility that is an integral part of the Oracle Management Agent. As a subsystem of the Agent, it allows you to quickly access the metric values for targets monitored by the Agent without the overhead of a Management Repository or other components of the Enterprise Manager framework.

Activating the Metric Browser

To configure the Oracle Management Agent's metric browser for debugging metrics without the Enterprise Manager Grid Control console:

1. Uncomment the enableMetricBrowser line in the

   $AGENT_HOME/sysman/config/emd.properties file:

   ```
   #enableMetricBrowser=True
   ```

   change to

   ```
   enableMetricBrowser=True
   ```

2. Make sure the Oracle Management Agent is stopped.

```
emctl stop agent
```

3. Restart the Management Agent.

```
emctl start agent
```

4. Open the emd.properties file and check the EMD_URL line.

   It will have the following format:

```
EMD_URL=http://<host>:<port>/emd/main
```

Adding a Target Instance to the targets.xml File

Before you can view your target metrics in the metric browser, you need to add an actual instance of your new target type to the targets.xml file without re synchronizing the Management Agent with the Management Service. To do this:

1. Create a new XML file containing the instance information for a specific target. The contents of this XML file must conform to the $AGENT_HOME/sysman/admin/dtds/TargetInstance.dtd file. The content for this file consists of the actual values specified for the instance properties you defined in the target type metadata file. For example, given the instance properties defined in a particular target metadata file, you create an XML file called "target2add.xml" that contains the following:

Example 2–7 target2add.xml File

```
<Target TYPE="oracle_listener" NAME="listener" VERSION="1.0">
<Property NAME="ListenerOraDir" VALUE="/myOH/oracle/network/admin"/>
<Property NAME="LsnrName" VALUE="LISTENER"/>
<Property NAME="Machine" VALUE="mymachine.us.oracle.com"/>
<Property NAME="OracleHome" VALUE="/myOH/oracle"/>
<Property NAME="Port" VALUE="15091"/>
</Target>
```

2. Use the command line instance installer to add the contents of the instance information file to the Management Agent's targets.xml file and reload the instance data. For the examples below, Management Agent home is located in /em/ and the instance information file is target2add.xml located in the /tmp/ directory.

   Add the contents of your new target information file to the targets.xml file.

```
emctl config agent addtarget <file location> <Agent Home location>
```

   Example:

```
emctl config agent addtarget /tmp/target2add.xml /em/
```

   Verify that your target information was added correctly:

```
emctl config agent listtargets <emloc>
```

   Example:

```
emctl config agent listtargets /em/
```

   Reload the modified target information.This operation is required in order for the new target instance to appear in the metric browser.

```
emctl reload
```

Looking at Your Metrics

Once the target instance has been added to the targets.xml file and the new information has been reloaded, you can view available targets and metrics through the metric browser. Access the following URL using any web browser

```
http://<host>:<port>/emd/browser/main
```

To find the port number used by the Management Agent, open the emd.properties file and search for the EMD_URL line.

2.2.1.3 Creating a Default Collection File

Defining your target type metrics allows you to view the data belonging to that target instance at any time. However, you may also wish to collect the values of some of your metrics to analyze the variations over some period of time. You set up metric collection intervals for your target instances by defining a default collections file for the specific target type.

> **Note:** Full definitions and usage instructions for descriptor tags used in the collection file can be found in the $AGENT_HOME/sysman/admin/dtds/TargetCollection.dtd file.

2.2.1.3.1 Default Collection File Located in $AGENT_HOME/sysman/admin/default_collection, the default collection file (XML) for a target type specifies the metrics to be collected and sent to the Management Repository, as well the frequency of these scheduled collections. Although you can use any name for a default collections file, it is recommended that you use a naming convention that makes it easy to associate the collection file with the corresponding target type metadata filename, such as using a filename identical to the target type metadata filename.

The following example is an excerpt from the Microsoft SQL Server default collections file.

Example 2–8 Microsoft SQL Server Default Collections File

```
<!DOCTYPE TargetCollection SYSTEM "../dtds/TargetCollection.dtd"[]>
<TargetCollection TYPE="microsoft_sqlserver_database">
<CollectionItem NAME="Response">
<Schedule>
<IntervalSchedule INTERVAL="5" TIME_UNIT="Min" />
</Schedule>
<Condition COLUMN_NAME="Status" CRITICAL="Running" OPERATOR="NE" />
</CollectionItem>

<CollectionItem NAME="SQLServer" UPLOAD_ON_FETCH="TRUE" CONFIG="TRUE">
<Schedule OFFSET_TYPE="INCREMENTAL">
<IntervalSchedule INTERVAL="5" TIME_UNIT="Min" />
</Schedule>
<MetricColl NAME="MSSQL_SQLServer" />
<MetricColl NAME="MSSQL_RegistrySetting" />
</CollectionItem>
.
.
.
<!-- Buffer cache hit ratio that is lower than 90% triggers warning and 80%
 triggers alert -->
<!-- Cache hit ratio that is lower than 80% triggers warning and 70% triggers
 alert -->
```

```
<CollectionItem NAME="MSSQL_MemoryStatistics">
<Schedule>
<IntervalSchedule INTERVAL="10" TIME_UNIT="Min" />
</Schedule>
<Condition COLUMN_NAME="buffer_cache_hit_ratio" WARNING="90" CRITIAL="80"
OPERATOR="LE" />
<Condition COLUMN_NAME="cache_hit_ratio" WARNING="80" CRITIAL="70" OPERATOR="LE"
/>
</CollectionItem>

<CollectionItem NAME="MSSQL_Alert">
<Schedule>
<IntervalSchedule INTERVAL="20" TIME_UNIT="Min" />
</Schedule>
</CollectionItem>

<CollectionItem NAME="MSSQL_LastDatabaseBackup">
<Schedule>
<IntervalSchedule INTERVAL="10" TIME_UNIT="Min" />
</Schedule>
</CollectionItem>

</TargetCollection>
```

For every target instance, the data collection intervals for the four metrics would be as follows:

- **Response**: Collected every 5 minutes.
- **Load**: Collected every 15 minutes.
- **memPool**: Collected every 15 minutes.
- **Interfaces**: Collected every 15 minutes.

If the threshold is not set in the collection file, administrators will not be able to edit/add the column threshold value from the Enterprise Manager console at a later point. To allow an administrator to change a threshold that does not have a default value, you can add a "NotDefined" entry for a specific threshold. For example:

```
<Condition COLUMN_NAME="db_session_osuser" CRITICAL="NotDefined"/>
```

2.2.1.3.2 Target Instance-Specific Collection Schedules Under certain circumstances, you may not want to have all target instances use the same collection schedule. To specify that different instances of a target type have different collection schedules, you can add additional collections files (for specific target instances) to $AGENT_HOME/sysman/emd/collections.

The following example shows a situation where we want two specific target instances ("Simple Server Alpha" and "Simple Server Beta") to have different "Response" and "Load" collection schedules. The two collection files to be added to the collections directory would contain the following:

Example 2–9 Default Collection File for Simple Server Alpha

```
<TargetCollection NAME="Simple Server Alpha" TYPE="simple_server">
    <CollectionItem NAME="Response">
        <Schedule>
            <IntervalSchedule INTERVAL="10" TIME_UNIT="Min"/>
        </Schedule>
    </CollectionItem>
</TargetCollection>
```

Example 2–10 Default Collection File for Simple Server Beta

```
<TargetCollection NAME="Simple Server Beta" TYPE="simple_server">
    <CollectionItem NAME="Load">
        <Schedule>
            <IntervalSchedule INTERVAL="30" TIME_UNIT="Min"/>
        </Schedule>
    </CollectionItem>
</TargetCollection>
```

Here we are collecting results from Simple Server Alpha's "Response" every 10 minutes and Simple Server Beta's "Load" every 30 minutes.

2.2.1.4 Adding SYSTEM Reports (Optional)

Developing a Management Plug-in allows you to add new SYSTEM reports that are associated with the target type(s) defined by your Management Plug-in. SYSTEM reports cannot be edited or deleted from the Enterprise Manager console. All out-of-box reports supplied with Information Publisher are SYSTEM reports.

To add report definitions to your Management Plug-in, you simply create a report definition file for inclusion with the plug-in. A report definition file consists of a conventional PL/SQL block that specifies pertinent information to extract from the Management Repository and the report elements used to format and display that data. You may define multiple reports for a given target type. For more information on the Information Publisher PL/SQL API and creating a plug-in report definition file, see Chapter 6, "Adding Reports".

2.2.1.5 Adding Related Links to Target Home Pages

If reports are defined as part of a Management Plug-in, and at least one of the reports is registered to be shown in the context of the target default home page (it is registered using the “add_report” PL*SQL function) then one or more related links may be added to the target default home page using the process described in this section.

Related Links are defined as dynamic instance properties in the target metadata file using the “DynamicProperties” tag with the following parameters: :

- NAME=”<any name>”
- PROP_LIST=”*RelatedLink_Name_*#;*RelatedLink_Dest_*#[;...]”
  - Where # can be any positive integer (Example: 1, 2, or 3)
  - PROP_LIST must be EXACTLY in the format specified here (including bumpy case). Otherwise, the related links will not show up on the target default home page.

  Also, a “QueryDescriptor” tag block that retrieves the number of “Name/Dest” pairs listed in PROP_LIST is required. The “RelatedLink_Name_#” appears as the hyperlinked text. The “RelatedLink_Dest_#” is the link destination (URL or Javascript).

  Example 2–11 is an excerpt from a target metadata file that contains a “DynamicProperties” block defining three related links that will appear on the default target home page.

Example 2–11 Related Links

```
<InstanceProperty NAME="EMPLOYEE_ID" CREDENTIAL="FALSE" OPTIONAL="FALSE">
  <Display>
        <Label NLSID="EMPLOYEE_ID_iprop">Employee ID</Label>
```

```
  </Display>
</InstanceProperty>
  <DynamicProperties NAME="Links" PROP_LIST="RelatedLink_Name_1;RelatedLink_Dest_
1;RelatedLink_Name_2;RelatedLink_Dest_2;RelatedLink_Name_3;RelatedLink_Dest_3"
FORMAT="ROW">
    <QueryDescriptor FETCHLET_ID="OSLineToken">
      <Property NAME="id" SCOPE="INSTANCE">EMPLOYEE_ID</Property>
      <Property NAME="command" SCOPE="GLOBAL">
          %perlBin%/perl %scriptsDir%/emx/oracle/getLinks.pl %id%
      </Property>
      <Property NAME="startsWith" SCOPE="GLOBAL">em_result=</Property>
      <Property NAME="delimiter" SCOPE="GLOBAL">|</Property>
    </QueryDescriptor>
  </DynamicProperties>
</InstanceProperties>
```

Note that in Example 2–11 the Perl script "getLinks.pl" is used to return the three links. The content of this script is shown in Example 2–12.

Example 2–12 Script getLinks.pl

```
my $id   = $ARGV[0];

my $name1 = "Employee: MY-PC";
my $link1 = "javascript:void window.open('http://my-pc.us.oracle.com', 'aria',
'resizable=1,menubar=1,toolbar=1,titlebar=1,status=1,scrollbars=1')";
my $name2 = "Employee: Hierarchy";
my $link2 = "javascript:void
window.open('http://employee.us.oracle.com:7777/pls/oracle/f?p=8000:3:720062688223
0853285::NO::PERSON_ID:$id', 'employee',
'resizable=1,menubar=1,toolbar=1,titlebar=1,status=1,scrollbars=1')";
my $name3 = "OTN";
my $link3 = "javascript:void window.open('http://otn.oracle.com', 'otn',
'resizable=1,menubar=1,toolbar=1,titlebar=1,status=1,scrollbars=1')";

print "em_result=$name1|$link1|$name2|$link2|$name3|$link3\n";
```

Because "getLinks.pl" is a support script, you add it to the Management Plug-in as a separate file using the EM CLI "add_mp_to_mpa" verb with the argument:

```
-file="MONITORING_SCRIPT:<path_to>/getLinks.pl"
```

Be very careful with the "javascript" syntax you use. Only some of the syntax is common to the popular browsers. For example, the second argument to "window.open" is a window name used for internal reference to the window and it must not contain any spaces.

2.2.1.6 Additional Scripts and Binaries

In addition to the target metadata, collection, and report definition files, you may also include specialized scripts and/or binary files that are used to include obtain status or metric data from your custom targets. An example is the Perl script "getLinks.pl" mentioned in the previous section.

2.2.2 Ensuring Accurate XML

Successfully creating target type metadata and collection files requires valid XML code. To assist with writing valid XML, Enterprise Manager provides ILINT, a development tool you can use to validate the XML used to define your code as you

develop your Management Plug-in. For more information on ILINT, see Chapter 4, "Validating XML".

2.2.3 Development Guidelines

When developing target type definition files for new Management Plug-ins, special consideration must be paid to the way in which you want a particular target type to be monitored. How a target type is monitored can greatly affect Enterprise Manager performance. General guidelines for defining target metadata and collections should be followed in order to optimize system performance.

2.2.3.1 Guidelines for Defining Target Metadata

Metadata is data about data. Generically, the term refers to any data used to aid in the identification, description and location of a network entity. Target metadata for an Enterprise Manager target consists of the metrics a user wants to expose and the methods used to compute those metrics.

2.2.3.1.1 Metadata Version Whenever the target metadata changes, the metadata version (meta_ver) should be incremented.

2.2.3.1.2 Real-Time Only Metrics Performance metrics can be classified into metrics that need to be computed to track performance trends and others that are more useful to drill down to get the details at a particular point in time. Real-time only metrics include those that need contextual information to return detailed information about a particular subset of the system, e.g., a specific tablespace to diagnose further.

2.2.3.1.3 Choice of Key Columns A key column in a metric is used in the management repository to trend performance data on an axis, e.g., the tablespace usage per database tablespace. An inappropriately chosen key column can result in too much collected data within the management repository. For instance, using the process ID in a Processes metric to upload to the repository.

You can have no key columns, but the query descriptor must return a single row.

2.2.3.1.4 Transient Columns In some cases, metric columns can be used to compute the values of other more interesting metric columns. In the cases where the original columns are not interesting for trending, these can be marked transient so that they are not uploaded to the repository and waste space.

2.2.3.1.5 Metrics and Microsoft Windows When creating metrics for custom targets, it is important to take into account the cost (CPU usage) of creating additional operating system (OS) processes. This is especially true for systems running Microsoft Windows where process creation is much more CPU intensive compared to UNIX-based systems such as Linux or Solaris. The percentage CPU utilization increases linearly with creation of child processes. To minimize process creation, avoid executing OS programs or commands from metric collection scripts. For example, when writing Perl scripts, avoid using the system function or backticks (``) to execute an OS command.

2.2.3.1.6 Target Properties (Static Versus Dynamic) Target properties are named values that can be used for computing the metrics of the target, or for display in the home page of the target. The list of target properties is specified in the metadata to allow data driven user interfaces to register targets, and for the Oracle Management Agent to validate that a target instance is complete.

- **Static Instance Properties**: These are properties whose values need to be specified for a target in the targets.xml entry for the target. An instance property can be

marked optional if the target declaration is considered complete even without the specification of the property. The metadata specification of a target property can also provide a default value for use in a configuration user interface.

- **Dynamic Instance Properties**: The Oracle Management Agent also allows for target instance properties to be "computed". Such properties are computed using a QueryDescriptor very similar to the ones used in metrics.

  Use of dynamic properties reduces the work involved in configuring a target by allowing certain properties to be computed rather than requiring the user to correctly specify their values (for example, the "Version" property of a database can be reliably computed given addressing information).

  The Oracle Management Agent allows for the fact that the target needs to be up for the successful computation of these dynamic properties by recomputing the properties each time a target bounce is detected (each time the target status changes to "Up").

2.2.3.1.7 Metrics The metric concept, as it pertains to the Oracle Management Agent, can be used to denote configuration and performance information.

- **Configuration Metrics**: Configuration metrics collect data similar to target properties that denote the configuration of the target. This information is periodically refreshed and can be used to track changes in the setup of a target. The collection interval on such metrics is typically on the order of about 24 hours.
- **Performance Metrics**: Performance metrics are used to track the responsiveness of a target. These metrics are typically collected more often than configuration metrics though the interval of some performance metrics may vary widely from those of others. Also, performance metrics usually ship with thresholds that are the basis of performance alerts for the target.

  A required metric for all targets is the "Response" metric consisting of a "Status" column with a condition on it. This metric is used to track the availability of the target.

Metric Naming Conventions

The conventions used in naming your metrics are extremely important because many areas of the Enterprise Manager user-interface are data-driven. For example, actual metric column labels and key values can be part of the page title, instruction text, or column headings. Specifically, these elements would appear on the Metric Details page, Edit Metric Threshold page, Notification Rules page, and other pages of the Enterprise Manager user-interface. For this reason, Oracle recommends the following metric naming conventions.

- All metric column names (labels) must be unique within a given target type and version, and easily understood by the user (metric units used as needed).

  Example: Tablespace Usage (%)

- All metric column names (labels) should be self-explanatory without dependence on the metric name.

  Example: Table Space Used (%)

- Key Column names should be self-explanatory. Key Column names are used when specifying metric thresholds or setting notifications. The following format should be used: all<key value name> objects

  Example: all (tablespace) objects

- Short names (up to 20 characters) associated with the metric column should be both clear and translatable.
- Across target versions, the same columns should use the same labels. This ensures columns, such as metric columns and short names, have the same NLS IDs across different target versions.

2.2.3.2 Guidelines for Defining Collections

Collections are the mechanism by which the Oracle Management Agent periodically computes the metrics of a target and uploads the data to the Management Repository. The most important thing to keep in mind when creating the collections for a target type is to avoid overburdening the Management Repository with excess data. In a large enterprise with hundreds of Oracle Management Agents and thousands of targets, the key to scalability is to limit the amount of data collected about a target that is uploaded to the repository. This is especially important since raw data is maintained for 24 hours - rollup benefits only accrue beyond that point.

2.2.3.2.1 Alert Message Guidelines Alert messages tell the user when something is wrong. These messages should also assist the user in solving the problem. We recommend following these content guidelines when writing alert messages:

- Alert messages should be meaningful. Avoid using terse, ambiguous messages unless the message is only applicable to the metric.
- Target down messages should, in addition to indicating that the target is down, include information indicating possible reasons why the target may be down.
- Error codes/messages should be included whenever possible.
- When appropriate, include information telling the user how to resolve or diagnose the problem.

2.2.3.2.2 Metric Evaluation Order It is important to pay attention to metric evaluation order so as to avoid metric collection failures. For example, the *Response* metric should be evaluated first in order to prevent a collection failure when a target is down. The Oracle Management Agent will evaluate metrics based on the order they are listed in the collection XML file.

Note: Programmatic logic of the Oracle Management Agent distributes the metric evaluations so that each evaluation is separated by approximately 10 seconds.

2.2.3.2.3 Collection Frequency In general, there is almost never a good reason to collect information at intervals smaller than 5 minutes. In the rare case where data variations occur at a smaller granularity and administrators need to be notified sooner, the Oracle Management Agent provides the capability to use a small collection interval to compute the metrics and threshold information while still only uploading data once in every n computation cycles.

2.2.3.2.4 Controlling Number of Rows Some metrics can result in the creation of a large number of rows in a Management Repository table. In some cases, only a subset of these rows may need to be uploaded to the repository. The Oracle Management Agent allows the specification of filter conditions that can be used to find rows to skip uploading. Also, a "limit_to" clause can be used on metrics that return sorted metric data to upload only the first n rows to the repository.

2.3 Creating a Management Plug-In Archive

Once you have created the Management Plug-in files, the next step is to create a Management Plug-in Archive (MPA). The MPA plays an important role at various stages of the Management Plug-in lifecycle. It serves the following functions:

- As a transport mechanism between your development environment and the Enterprise Manager framework.
- As a transport mechanism between different Enterprise Manager installations.
- As a container for the Management Plug-in. A MPA may contain multiple Management Plug-ins.

A Management Plug-in is created by adding the files previously discussed to an MPA using the Enterprise Manager Command Line Interface (EM CLI). Each call to the EM CLI adds another unique Management Plug-in to the MPA. For each Management Plug-in, the EM CLI allows you to specify a base version of the Management Agent that the plug-in is expected to work against and a base version that the Oracle Management Service must be for the plug-in to be imported into the Management Repository. To create a MPA, perform the following

1. Open a terminal window on a machine where the EM CLI client is installed.
2. At the command prompt issue the add_mp_to_mpa verb. The following example shows the verb parameters that must be supplied. For more information about the add_mp_to_mpa verb, see the command line help.

Example 2–13 Using the EM CLI to Create a Management Plug-in Archive

```
emcli add_mp_to_mpa
          -mpa="/my_dir/my_new_type.jar"
          -mp_version="2.0"
          -ttd="/my_dir/ttd/new_type.xml"
          -dc="/my_dir/dc/new_type.xml"
          -file="REPORT_DEFINITION:/my_dir/report1.sql
          -file="REPORT_DEFINITION:/my_dir/report2.sql
          -file="MONITORING_SCRIPT:/my_dir/script1.pl
          -file="MONITORING_SCRIPT:/my_dir/script2.pl
          -file="MONITORING_BINARY:/my_dir/bin1
          -func_desc="Management Plug-in to define target type new_type"
```

Briefly, the verb options are:

- **mpa**

  The name of the Management Plug-in Archive where the Management Plug-in is to be added.

- **mp_version**

  The version of the Management Plug-in to be created. The Management Plug-in version should be incremented whenever any of the files in the management Plug-in are changed.

- **tdd**

  The explicit path of the target type metadata file.

- **dc**

  The explicit path of the default collection file.

- **oms_version**

The minimum OMS version that is compatible with this Management Plug-in.

- **file**

  The type and path of the other Management Plug-in files to be added. The following types are supported:

  – MONITORING_BINARY

  – POLICY_DEPLOY

  – POLICY_UNDEPLOY

  – MONITORING_SCRIPT

  – REPORT_DEFINITION

- **func_desc**

  The functional description for the Management Plug-in. This description appears in the Enterprise Manager console once the plug-in has been imported.

- **req_desc**

  The Requirements description of the Management Plug-in. This description appears in the Enterprise Manager console and specifies any plug-in deployment requirements.

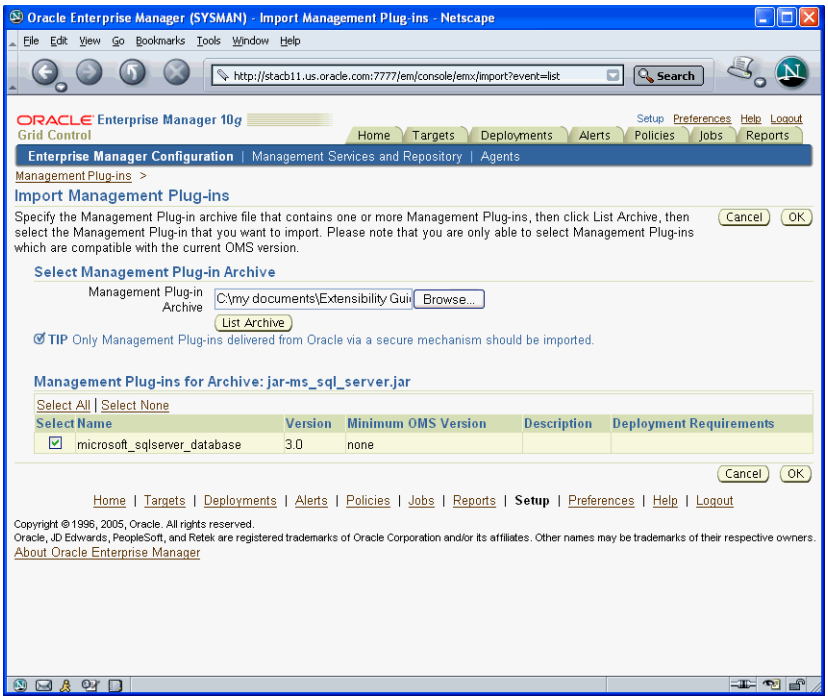
2.4 Uploading the Management Plug-in Archive into Enterprise Manager

After using the EM CLI to create the Management Plug-in Archive, you are ready to upload the Management Plug-in Archive file to Enterprise Manager. Uploading the archive allows you to view all available plug-ins contained in the archive. You can then select which plug-ins you want to import into Enterprise Manager. You must have Super Administrator privileges in order to add Management Plug-ins to the system.

To upload a Management Plug-in Archive:

1. From the Enterprise Manager console, click Setup.
2. Click Management Plug-ins from the navigation bar at the left.
3. Click Import. The Import Management Plug-ins page displays.
4. In the Select Management Plug-in Archive section, specify the Management Plug-in Archive file.
5. Click List Archive to view Management Plug-ins contained within the archive.
6. Select the Management Plug-ins you want to import and click OK.

Figure 2–1 Import Management Plug-ins Page

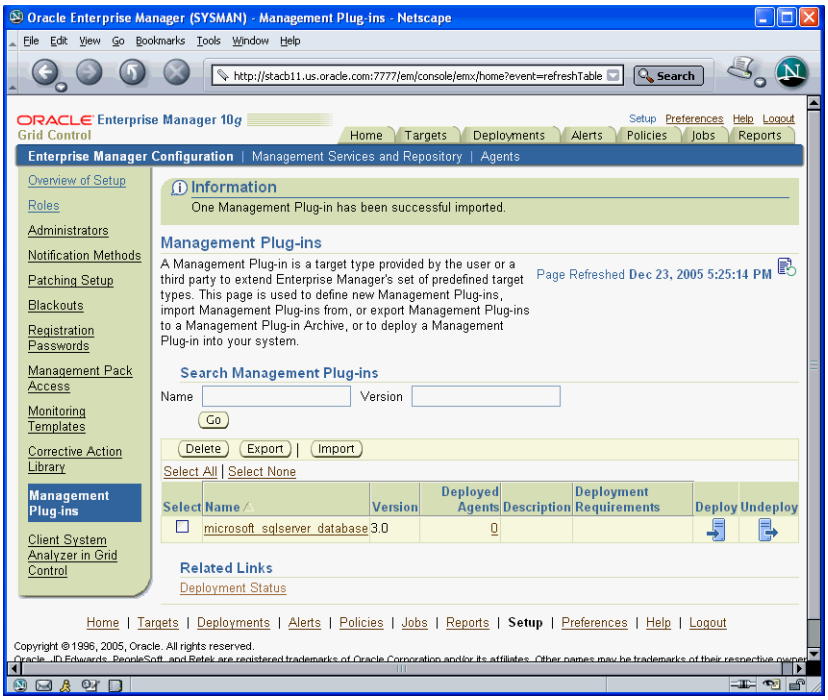


Graphic shows the Import Management Plug-ins page with the Management Plug-in Archive file selected.

****

Upon successful import, the plug-in appears in the Management Plug-in list.

Figure 2–2 Management Plug-in



Graphic shows the Management Plug-in page with a successfully imported plug-in.

****

At this point, the Management Plug-ins have been extracted from the archive file and imported into the Management Repository. The Management Plug-in is now ready to be deployed to Management Agents within your Enterprise Manager environment. To deploy your plug-in, click the Deploy icon. Enterprise Manager guides you through a simple deployment process.

2.5 Adding a Target Instance

After a Management Plug-in has been deployed to an Agent, you are ready to add new target instances of the type defined by your Management Plug-in. When you add a target instance, monitoring and administrative functionality is automatically extended to that target. To add a target instance:

1. From the Monitored Targets section of the Management Agent home page, choose the target type defined by your plug-in from the Add drop-down menu and click Go. The Add target page appears.
2. Enter the requisite target properties and click OK. The newly added target appears in the Agent's Monitored Targets list.

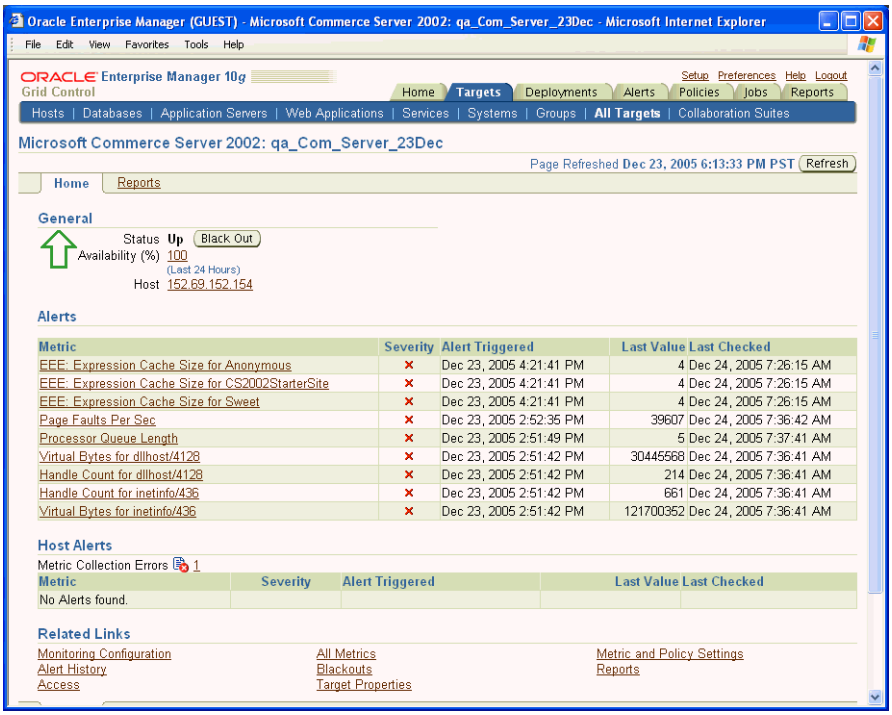
A default target home page is provided that supplies requisite information about the target as shown in Figure 2–3.

2.6 Viewing Results

From a default target home page, a user can drill down to specific metrics that have been defined for the target type.

The following figure shows a target home page for a Microsoft SQL Server instance.

Figure 2–3 Microsoft Commerce Server Target Home Page

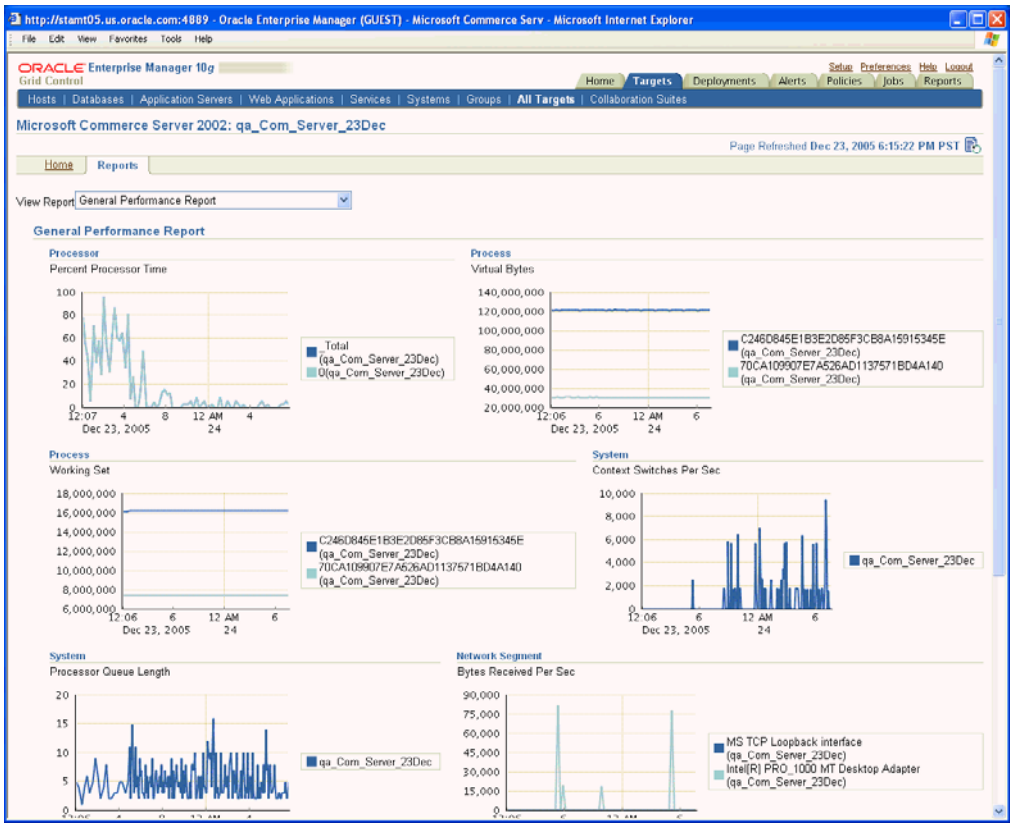


Graphic displays a Microsoft Commerce Server target home page.

****

Because report definitions were included with this Management Plug-in, an additional Reports subtab appears with the home page. As shown in the next figure, adding reports can greatly improve the monitoring capabilities of your Management Plug-in. For more information about creating Management Plug-in report definitions, see Chapter 6, "Adding Reports".

Figure 2–4 Microsoft SQL Server Reports Subtab



Graphic shows the reports subtab.

****

2.7 Troubleshooting Management Plug-ins

On occasion, you may encounter errors when when managing Management Plug-ins. The most common problems are as follows:

- **Deployment fails. "Agent Unreachable"**

  Ensure that the Management Agent is up and running.

  Log in as "root" and execute the root.sh script at the Management Agent.

- **Undeploy fails. "Agent Unreachable"**

  Ensure that the Management Agent is up and running.

- **Unable to delete the Management Plug-in.**

  The Management Plug-in must be undeployed from ALL Agents to which it was previously deployed. Note that this includes any Agents that are currently down.

- **Reports do not show any data.**

  Depending on the collection interval settings, it may take time for the Agent to collect the metric data for the target. Management views used by the reports can also determine when data is available, as would be the case if queries against the Management Repository are performed against rolled up data.

2.8 Management Plug-in Development Kit

Enterprise Manager provides a Plug-in Development Kit (PDK) containing command line tools that help you develop and test Management Plug-ins with ease and efficiency. These tools include the following:

- `check_mp` -- verifies that a Management Plug-in contains correct XML syntax in all metadata files and validates them against DTD files. Additionally, the tool performs semantic checks to determine possible problems.
- `collect_mp_stats` -- collects metric data defined by the Management Plug-in metadata and generates an HTML-based report that can help you evaluate the Management Plug-in's performance impact on the Agent.

To download the PDK directly from the Enterprise Manager console:

1. Navigate to the Management Plug-in home page (Setup-->Management Plug-in)
2. From the **Related Links** area, click **Download PDK**. You are taken to the PDK download page.
3. Read the system requirements and installation instructions.
4. Click **Download the PDK to your workstation**.

3

Adding Charts

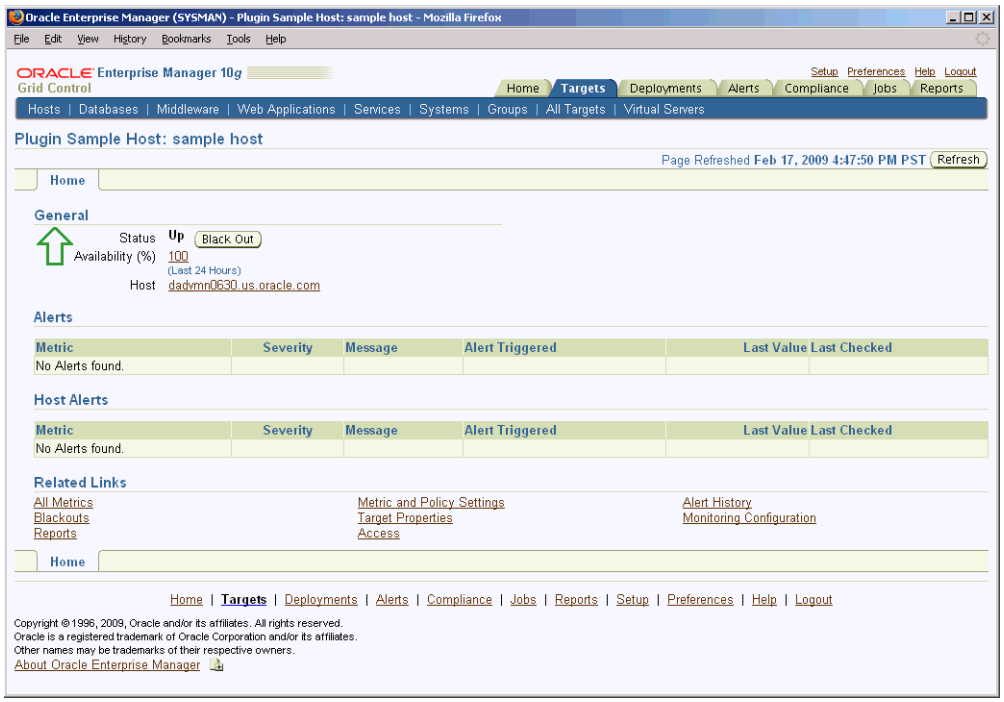
Once a Management Plug-in is deployed and a target instance created, a basic default target home page is automatically generated. The Management Plug-in framework lets you increase the utility of this page allowing you to display performance and configuration data using charts (pie, bar, and time series). By adding charts to a target home page, you can present key monitoring information about the target in an intuitive, easy-to-read format. This allows administrators to monitor and manage target instances with ease. In some cases, adding charts may eliminate the need for creating specialized reports See "Adding Reports" on page 6-1 for more information about Information Publisher.

This chapter covers the following topics:

3.1 Management Plug-in Home Page

The default target home page provides basic information regarding target status such as Availability, Up/Down status, and Alerts. Depending on what level of monitoring support the Management Plug-in provides, there may be additional sections. Most all information is in tabular format.

Figure 3–1 Default Target Home Page without Charts

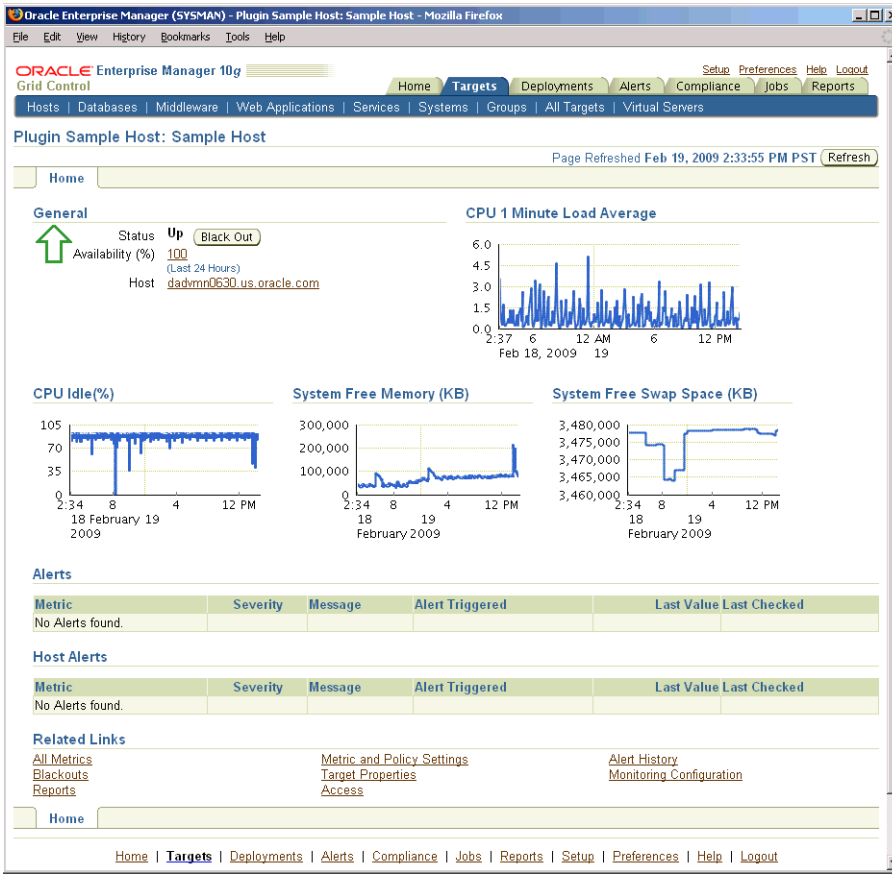


Graphic shows default target home page without charts.

****

Adding charts to the target home page makes key graphical information normally found in Enterprise Manager reports readily available on the target home page.

Figure 3–2 Default Target Home Page with Charts



Graphich shows default target home page with charts.

****

By adding a chart definition XML file to your Management Plug-in Archive, you will be able to specify any number of charts (pie, bar, and time series) on the target home page.

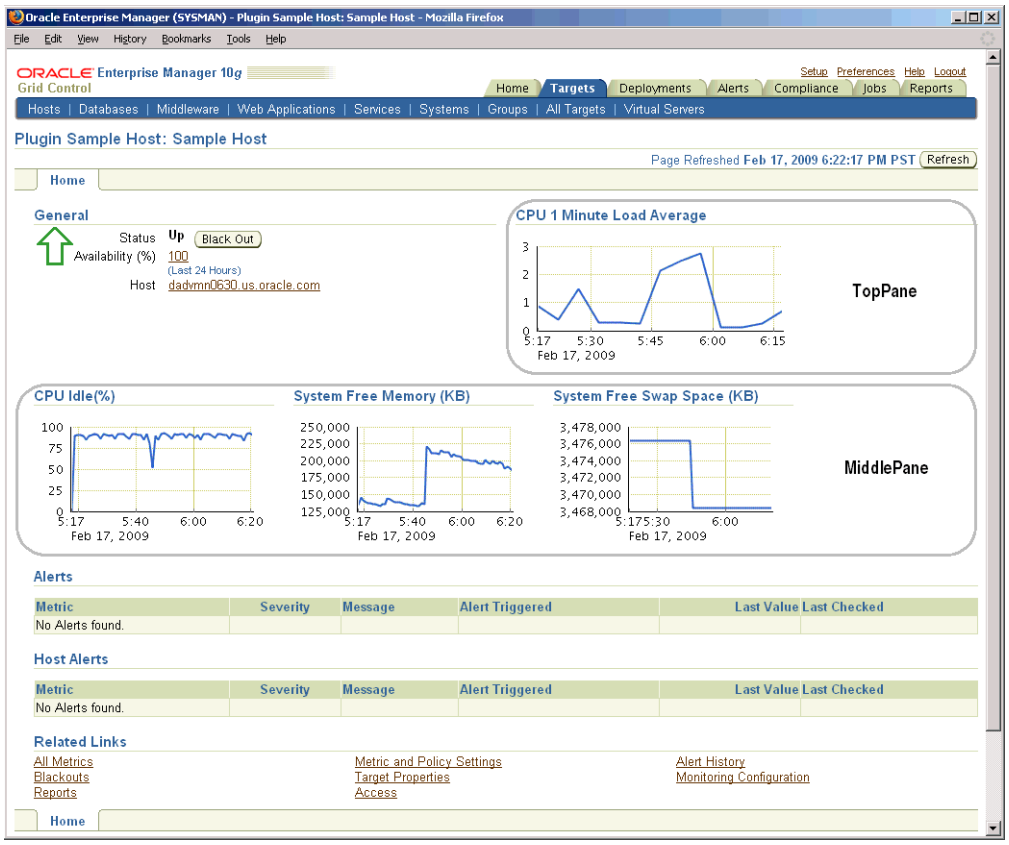
3.2 Home Page Charts File

Like other Management Plug-in files, the XML used to define a home page chart file is straightforward and relatively compact. This file defines the metadata used to render charts using metric or SQL data sources for the target.

Home Page Chart File Structure

From the examples, you can see that the basic structure of the home page charts definition file is organized according to chart location on the target home page. One chart may be placed on the top of the page and to the right of the General Information Region (TopPane). Any number of charts may be placed in a row that renders just below this area (MiddlePane).

Figure 3–3 TopPane and MiddlePane Locations within the Target Home Page



Graphic shows the location of charts within a target home page.

****

Although there is no technical limit to the number of charts you can add, in practice, at most 3 - 4 charts will fit comfortably in the MiddlePane.

3.2.1 Defining the Home Page Chart File

The home page charts file must begin and end with a *HomepageCharts* tag to identify the XML file as a chart definition file for the plug-in. As shown in the following sample, the opening *HomepageCharts* tag requires that you specify a TARGET_TYPE property, which is the target type defined by the TYPE parameter of the *TargetMetadata* tag found in the Management Plug-in's target type metadata file. See "Target Definition Files" on page 3 for more information about the target type metadata file.

```
<HomepageCharts TARGET_TYPE="my_plugin_target_type">
     ...
</HomepageCharts>
```

3.2.2 Defining Charts Sets

Between the opening and closing *HomepageCharts* tags you define the charts and optionally when they should appear on the target home page using the *ChartSet* tag. The *ChartSet* tag functions as a container for all charts that are to rendered on the target home page. You can define multiple *ChartSet* containers within the home page

chart definition file. By using the optional META_VER parameter, you can specify different sets of charts (or no charts at all) be rendered on the target home page for specific versions of the Management Plug-in target type. Shown below are the three acceptable implementations of the *ChartSet* tag:

```
<HomepageCharts TARGET_TYPE="my_plugin_target_type">
     <ChartSet META_VER="1.0">
     ...
     </ChartSet>

     <ChartSet>
     ...
     </ChartSet>

     <ChartSet META_VER="2.0"/>
</HomepageCharts>
```

In the example, each implementation of the *ChartSet* tag operates as follows:

- `<ChartSet META_VER="1.0">`

  Charts defined in this set appear on target home pages for version 1.0 Management Plug-in target types.

- `<ChartSet>`

  Charts defined in this set appear on target home pages for all Management Plug-in target types that are not version 1.0 or 2.0 (charts displayed by default).

- `<ChartSet META_VER="2.0" />`

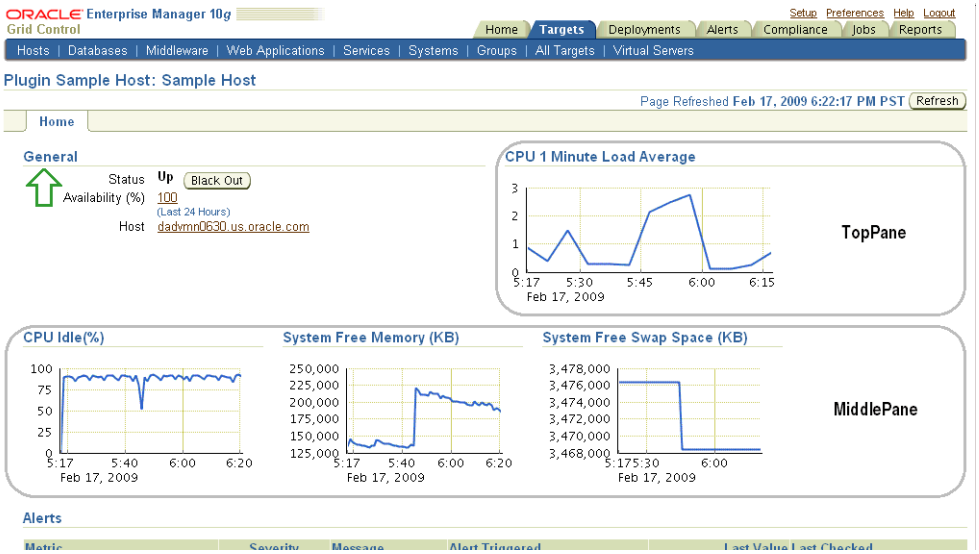
  No charts will appear on the target home pages for version 2.0 Management Plug-in target types.

3.2.3 Defining Chart Location

Each *ChartSet* block gives you the option to define charts in two locations: *TopPane* and *MiddlePane*. The TopPane refers to the top right of the target home page, next to the General Metrics section. This area can accommodate a single chart. The MiddlePane refers to the row immediately below this area. You can define charts for either or both areas.

```
<HomepageCharts TARGET_TYPE="my_plugin_target_type">
     <ChartSet>
          <TopPane>
          ...
          </TopPane>
          <MiddlePane>
          ...
          </MiddlePane>
     </ChartSet>
</HomepageCharts>
```

Figure 3–4 Top and Middle Pane Locations



Graphic shows top and middle pane locations for charts.

****

3.2.4 Creating Charts

Once you have decided the when and where the charts should appear, you need to define the actual charts. Within each pane, you use the *Chart* tag to specify the type of chart to be rendered, chart attributes, as well as any data acquisition methods required to render the chart.

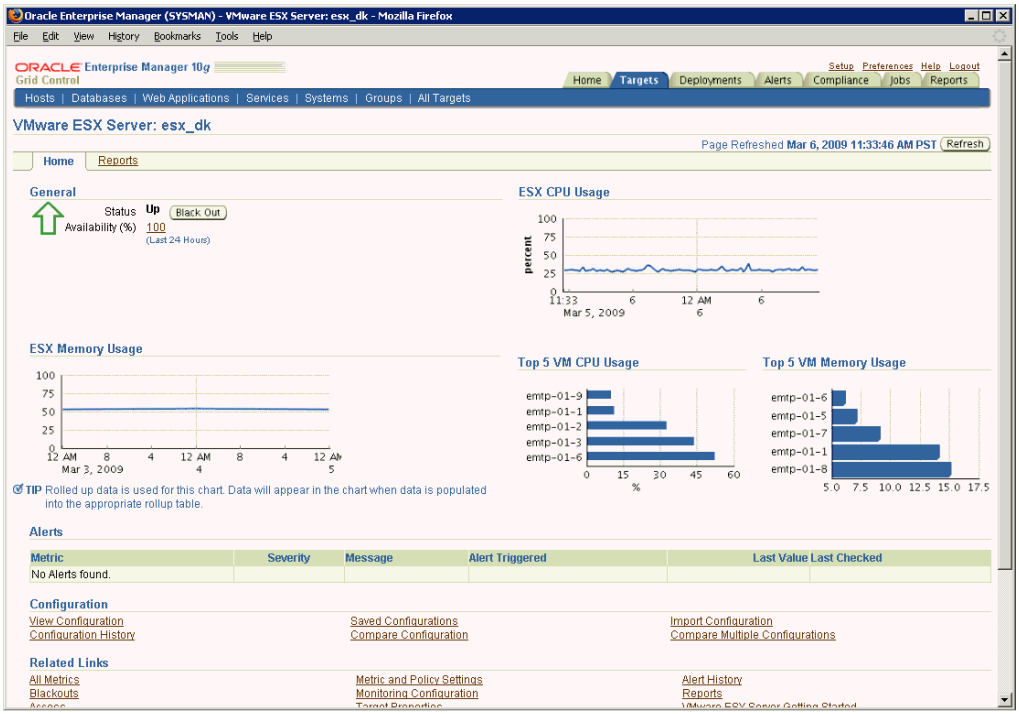
```
<HomepageCharts TARGET_TYPE="my_plugin_target_type">
     <ChartSet>
          <TopPane>
               <Chart TYPE="timeSeriesChart">
               ...
               </Chart>
          </TopPane>
          <MiddlePane>
               <Chart TYPE="pieChart">
               ...
               </Chart>
               <Chart TYPE="barChart">
               ...
               </Chart>
          </MiddlePane>
     </ChartSet>
</HomepageCharts>
```

As shown in the example, each *Chart* tag must specify one of the following TYPE properties that define the type of chart (pie, time series, bar) to be rendered on the target home page:

- *timeSeriesChart*
- *pieChart*
- *barChart*

The following example demonstrates chart implementation for a VMware target type. As shown in Figure 3–5, both bar and time series charts have been created, making use of a variety of optional chart properties, such as *stacked*, *timeGranularity*, *shadowEffect*, *xAxisLabel*, *yAxisLabel*. The code used to define the charts is shown in Example 3–1.

Figure 3–5 VMware Home Page Charts



Graphic displays VMware server target home page with charts.

****

Example 3–1 Home Page Charts XML for VMware

```
<HomepageCharts TARGET_TYPE="vmware_esx_server">

 <ChartSet>
   <TopPane>
     <Chart TYPE="timeSeriesChart">
       <ChartProperty NAME="metric">HostSummCPU</ChartProperty>
       <ChartProperty NAME="column">host_cpu_usage</ChartProperty>
       <ChartProperty NAME="width">375</ChartProperty>
       <ChartProperty NAME="height">150</ChartProperty>
       <ChartProperty NAME="legendPosition">south</ChartProperty>
       <ChartProperty NAME="titleVisible">true</ChartProperty>
       <ChartProperty NAME="title">ESX CPU Usage</ChartProperty>
       <ChartProperty NAME="subtitle"> </ChartProperty>
       <ChartProperty NAME="destination">metricDetail</ChartProperty>
       <ChartProperty NAME="yAxisLabel">percent</ChartProperty>
     </Chart>
   </TopPane>

   <MiddlePane>
     <Chart TYPE="timeSeriesChart">
       <ChartProperty NAME="metric">HostSummMem</ChartProperty>
```

```
        <ChartProperty NAME="column">host_mem_usage</ChartProperty>
        <ChartProperty NAME="width">375</ChartProperty>
        <ChartProperty NAME="height">150</ChartProperty>
        <ChartProperty NAME="legendPosition">south</ChartProperty>
        <ChartProperty NAME="titleVisible">true</ChartProperty>
        <ChartProperty NAME="title">ESX Memory Usage</ChartProperty>
        <ChartProperty NAME="subtitle"> </ChartProperty>
        <ChartProperty NAME="destination">metricDetail</ChartProperty>
        <ChartProperty NAME="timeGranularity">days</ChartProperty>
      </Chart>
      <Chart TYPE="barChart">
        <ChartProperty NAME="sql">select vm, cpu_usage from (select key_value as
vm, value as cpu_usage from mgmt$metric_current where metric_name = 'VMSummCPU'
and metric_column = 'vm_virtual_cpu_usage' and target_guid = ??HC_TARGET_GUID??
order by to_number(value) desc) where rownum &lt; 6</ChartProperty>
        <ChartProperty NAME="width">275</ChartProperty>
        <ChartProperty NAME="height">150</ChartProperty>
        <ChartProperty NAME="orientation">horizontal</ChartProperty>
        <ChartProperty NAME="titleVisible">true</ChartProperty>
        <ChartProperty NAME="title">Top 5 VM CPU Usage</ChartProperty>
        <ChartProperty NAME="legendPosition">south</ChartProperty>
        <ChartProperty NAME="barLegendLabel1">CPU Usage %</ChartProperty>
        <ChartProperty NAME="destination">reportTab</ChartProperty>
        <ChartProperty NAME="reportTitle">ESX Server CPU Usage</ChartProperty>
        <ChartProperty NAME="xAxisLabel">%</ChartProperty>
        <ChartProperty NAME="stacked">true</ChartProperty>
      </Chart>
      <Chart TYPE="barChart">
        <ChartProperty NAME="sql">select vm, mem_usage from (select key_value as
vm, value as mem_usage from mgmt$metric_current where metric_name = 'VMSummMem'
and metric_column = 'vm_mem_usage' and target_guid = ??HC_TARGET_GUID?? order by
to_number(value) desc) where rownum &lt; 6</ChartProperty>
        <ChartProperty NAME="width">275</ChartProperty>
        <ChartProperty NAME="height">150</ChartProperty>
        <ChartProperty NAME="orientation">horizontal</ChartProperty>
        <ChartProperty NAME="titleVisible">true</ChartProperty>
        <ChartProperty NAME="title">Top 5 VM Memory Usage</ChartProperty>
        <ChartProperty NAME="legendPosition">south</ChartProperty>
        <ChartProperty NAME="barLegendLabel1">Memory Usage %</ChartProperty>
        <ChartProperty NAME="destination">reportTab</ChartProperty>
        <ChartProperty NAME="reportTitle">ESX Server Memory Usage</ChartProperty>
        <ChartProperty NAME="shadowEffect">3d</ChartProperty>
      </Chart>
    </MiddlePane>

  </ChartSet>
</HomepageCharts>
```

3.3 Chart Properties

Between the beginning and ending Chart tags, you define how the chart should be rendered by specifying chart properties using *ChartProperty* tags. For example:

```
<ChartProperty NAME="legendPosition">default</ChartProperty>
```

You may specify as many *ChartProperty* tags a necessary to render the exact type of chart you want to appear. For example:

The following table lists chart properties that apply to all chart types.

Important: ChartProperty values are case-sensitive

Table 3–1 Chart Properties

Option	Required	Values	Default	Description
chartType	Yes	pieChart barChart timeSeriesChart	NA	Defines the chart display type: Bar, Time Series, or Pie.
width	No		NA	Defines the width of the chart graphic. This property only controls the size of the chart graphic itself and not any other elements which are rendered along with it, such as the title and subtitle.
height	No		NA	Defines the height of the chart graphic, this attribute only controls the size of the chart graphic itself and not any other elements which are rendered along with it, e.g. the title, subtitle, etc.
timeGranularity	No	SECOND MINUTE HOUR DAY WEEK MONTH YEAR	HOUR	Defines the units used to display data for time series charts.
legendPosition	No	For Pie Charts: ■ north ■ south ■ east ■ west For Time Series/Bar Charts ■ south ■ east	default	Controls the location of the chart.
legendVisible	No	true/false/external	true	Determines whether or not the chart legend is displayed. Set this property to 'false' to hide the legend. To position the legend outside of the chart boundary, set this property to ;external. Position a legend external to a chart if you want to control the size of the image separately from the size of the legend. An "external" legend can contain any number of entries of any length without truncating text. Setting *legendVisible*="external" causes image generation to occur on the first pass.

Table 3–1 (Cont.) Chart Properties

Option	Required	Values	Default	Description
legendWidthFraction	No	> 0 <= 1	NA	Specifies the fraction of the overall chart width that the legend should occupy, no default.
legendHeightFraction	No	> 0 <= 1	NA	Specifies the fraction of the overall chart height that the legend should occupy, no default.
title	No	Text	NA	Defines the title string to be displayed in header element above chart graphic .
titleVisible	No	true/false	false	Disables display of the title (header) created with the chart. This allows for the specification of a title attribute without its display. The *title* property is required for ADA compliance.
subtitle	No	Text	NA	Text to appear in subtitle just below header element which includes title and above chart graphic.
orientation	No	horizontal vertical	Vertical	Defines the visual orientation of the chart. Valid values are either "horizontal" or "vertical". This attribute is only valid with the *chartType* attribute set to *barChart* or *timeSeriesChart*, it has no affect on the *pieChart*.
shadowEffect	No	none shadow 3d default	default	Determines whether charts appear with a shadow or 3-D effect. This property is only applies to bar and pie charts. There are three possible values for this attribute: ■ **3d**--renders the bars or the pie with a 3d effect ■ **shadow**--renders the graphics with a shadow effect not as obvious as the 3d effect ■ **none**--renders the graphic as 2d (flat). The default setting for bar charts is "none" and the default for pie charts is "3d".
noDataMessage	No	Text	"No data is currently available."	Specifies the message that should appear in place of the chart if no data is available from the data source. If you do not specify a message, default message "No Data Available" will appear. In cases where the data source generates an exception message, the text of this message will be displayed in place of the chart graphic.

Table 3–1 (Cont.) Chart Properties

Option	Required	Values	Default	Description
sql	No	Valid SQL		Defines the SQL query used to obtain data for chart generation. Depending on the type of chart, SQL queries must return: ■ 3 columns for time series charts (key, timestamp, value) ■ 2 columns for pie charts (key, value) ■ 2 or more columns for bar charts (key, value[, value]) Note: If the SQL text contains any XML syntax characters, then the SQL text should be embedded in a CDATA block. Also, if the target instance GUID is needed in the SQL query, HC_TARGET_GUID can be used.
axisMin	No	Integer		Defines the Y-Axis minimum value.
axisMax	No	Integer		Defines the Y Axis maximum value.
xAxisLabel	No	Text	NA	For bar and line charts, a label to appear along the X-axis (string series or time series axis) label.
showXAxisLabels	No	true/false	true	Controls the display of X-axis labels. “false” disables display of X-axis values (currently applies to bar charts only).
yAxisLabel	No	Text	NA	For bar and line charts, a label to appear along the Y-axis (numeric axis).
yAxisVisible	No	true/false	false	Disables the display of the Y-axis; valid values "true" or "false" (default).

Table 3–1 (Cont.) Chart Properties

Option	Required	Values	Default	Description
destination	No	metricDetail keyedMetricDetail reportTab		Defines chart hyperlinks. See Section 3.4, "Linking from a Chart to Another Destination" for more information.
reportTitle	No	Text	NA	Defines the chart title entry in the drop-down menu on the target Reports page. This property is only relevant when used in conjunction with the `destination` chart property when it is set to `reportTab`.
fill	No	none absolute cumulative	none	For line charts, fill allows the area under the lines to be filled (solid coloring) using the same color as the line. The fill property also provides a means of creating a stacked area chart. Valid Values: ■ **none** -- default value ■ **cumulative** -- causes the values for the lines to be added or stacked. ■ **absolute** -- appears the same as the "none" setting, but with the area under the lines filled. When using the *fill* property, ensure that there is no confusion for users as to whether the data in the chart is cumulative or absolute

Chart Type-Specific Properties

The following tables list chart properties that are specific to each of the three chart types (pie, time series, and bar).

Table 3–2 Time Series Chart Properties

Option	Required	Values	Default	Description
metric	Yes	Text	NA	A metric and column must be specified or a SQL statement that returns the appropriately formatted data.
column	Yes	Text	NA	
yAxisWidth	No			Forces the width of the Y-Axis to be a fixed size this allows the caller to specify the same width for a number of charts stacked vertically and thus have the axes aligned.

Table 3–3 Pie Chart Properties

Option	Required	Values	Default	Description
pieValuesInLegend	No	value percent none	none	Defines whether or not values for pie slices are included in the legend along with the label for the pie slice. The default value for this attributes is “none”. If specified as either “percent” or “value” then the numeric value is displayed along with the pie slice label in the form, “pie slice label (numeric value)”. If “percent” is specified, then the percentage out of the total of all slice values is calculated and displayed, otherwise, the raw value of the slice is displayed. The *pieValuesInLegend* property is ignored for *chartType*s other than *pieChart* and if the *legendVisible* attribute is set to “false”.
pieSlicesFromColumn	No	true/false	false	Determines how data is retrieved from the data source for display in a pie chart. See "Linking from a Chart to Another Destination" on page 3-14 for more details. This attribute is ignored for *chartType*s other than *pieChart*.
showSlicePercentLabels	No	true/false	false	Determines whether each slice is labeled with a percentage value. This property is ignored for *chartType*s other than *pieChart*.
showSliceValueLabels	No	true/false	false	Determines whether each slice is labeled with the raw numeric value for that slice. This attribute is ignored for *chartType*s other than *pieChart*.
hideZeroSlices	No	true/false	false	When set to "true", this property remove entries from the legend for any slice with a zero value.

Table 3–4 Bar Chart Properties

Option	Required	Values	Default	Description
barLegendLabel#	Yes	Integer	None	Specifies that a legend label be associated with each data stream. # starts at 1 and represents a data stream. If you are only going to have one bar for each key, you would only have barLegendLabel1. If you are going to have more than one bar for each key, you would specify barLegendLabel1,... barLegendLabelN (where N is the number of the last data stream).
stacked	No	true/false	false	When set to "true", displays stacked bars for each key value of a bar chart.

3.4 Linking from a Chart to Another Destination

Any chart appearing on the target home page can be linked to another destination that provides more information/greater detail. This provides users with easy 'drill-down' access to pertinent information.

You make a chart linkable by setting the `destination` chart property. The Management Plug-in framework allows you to link a chart to one of three destinations:

- **metricDetail** - Links the chart to a Metric Detail page displaying a time series chart.
- **keyedMetricDetail** - Links the chart to a Metric Detail page for a metric with keys. A table with values for each key is displayed.
- **reportTab** - Links the chart to a report that has been developed for the Management Plug-in registered so that it appears on the target Reports page (Reports subtab). The administrator must be registered to view the report.

The following example implements a time series chart that is linked to the Response Metric Detail page.

```
<Chart TYPE="timeSeriesChart">
        <ChartProperty NAME="metric">Response</ChartProperty>
        <ChartProperty NAME="column">Load</ChartProperty>
        <ChartProperty NAME="width">300</ChartProperty>
        <ChartProperty NAME="height">150</ChartProperty>
        <ChartProperty NAME="legendPosition">south</ChartProperty>
        <ChartProperty NAME="titleVisible">true</ChartProperty>
        <ChartProperty NAME="title">CPU 1 Minute Load Average</ChartProperty>
        <ChartProperty NAME="subtitle"> </ChartProperty>
        <ChartProperty NAME="destination">metricDetail</ChartProperty>
</Chart>
```

Required Chart Property Values for Destinations

Depending on the destination type, linked charts require that certain chart properties be included with the chart definition.

metricDetail and keyedMetricDetail

To link a chart to either a Metric Detail page or a Metric Detail page for keys, the required chart properties are *column*, *metric*, and *timeGranularity*.

If both *metric* and *column* properties have already been defined, those values will be used to generate a link for that metric. All that is required is to specify the appropriate destination type (*metricDetail* or *keyedMetricDetail*). Implementing links in this way makes it possible to create a chart that can link to a Metric Detail page via custom SQL queries (when *metric* and *column* properties would normally not be specified).

The *timeGranularity* property will default to HOUR if it is not explicitly defined as part of the chart definition. However, if this property had already been specified in the chart definition, it will be used for the link as well.

reportTab

To link a chart to a report, the *reportTitle* chart property is required.

3.5 Adding Home Page Charts to the Management Plug-in Archive

Once you have defined your home page charts file, you use the Enterprise Manager Command Line Interface (EMCLI) verb `add_mp_to_mpa` to add the new home page chart definition file to a Management Plug-in Archive. As shown in the following example, you specify HOMEPAGE_DEFINITION as the file type to add a home page chart definition file to the plug-in archive.

Example 3–2 Using EMCLI to Create a Management Plug-in Archive

```
./emcli add_mp_to_mpa
     -mpa=host_sample.jar -mp_version=1.0
     -ttd=host_sample_ttd.xml
     -dc=host_sample_dc.xml
     -file="MONITORING_SCRIPT:data_collector.pl"
     -file="REPORT_DEFINITION:host_sample_perf_report.sql"
     -file="REPORT_DEFINITION:host_sample_config_report.sql"
     -file="HOMEPAGE_DEFINITION:host_sample_homepage_charts.xml"
     -func_desc="Demo Plug-in: Linux host monitoring."
     -req_desc="Requirements: Requires that the Agent that hosts the target
instances be running on Linux. If the 'Use Fake Data' property is set when adding
a target instance, then all data provided will be generated and a Linux Agent is
not required.";
```

At this point, you are now ready to use Enterprise Manager to import and deploy the new Management Plug-in. Once you add a target instance of the plug-in target type, your newly defined charts will appear on the target home page.

4

Validating XML

Successfully defining new Management Plug-in target definition files relies on accurate XML: It is essential that your Management Plug-in's target metadata and collection files be syntactically and structurally correct. To assist you with developing your XML files, Enterprise Manager supplies an XML verification tool called ILINT. This chapter covers the following topics:

4.1 What is ILINT?

Integrator LINT (ILINT) is a tool that helps you validate target metadata while defining new target types: You can perform real-time validation for the XML you write for target, instance, and collection metadata while you are writing the target metadata. ILINT also allows you to test runtime data collection for a new target type to validate data correctness.

This tool can also be used as an integrator certification tool to ensure that target type metadata conforms to acceptable standards.

4.1.1 What types of validation does ILINT perform?

ILINT allows you to perform two types of validation: static, which checks the correctness of your XML and dynamic, which checks the correctness of the data collected. ILINT validates for proper XML syntax by comparing target type metadata files against their respective DTD's ($AGENT_HOME/sysman/admin/dtds) and management agent semantics.

4.1.1.1 Static XML Validation

ILINT performs a static XML validation by performing heuristic checks, checking Management Repository constraints, and validating XML syntax on the following files:

- <target_metadata>.xml

  Directory Location: $AGENT_HOME/sysman/admin/metadata

- <target_collection>.xml

  Directory Location: $AGENT_HOME/sysman/admin/default_collection

- targets.xml (target list file)

Directory Location: $AGENT_HOME/sysman/emd

Note: In general, the targets.xml file should not be edited manually. However, when developing new metadata files, you may need to add specific target instance information to the target instance file for testing purposes. The targets.xml file must be structurally and syntactically correct because the Management Agent relies on the instance-specific information contained within the file.

Specifically, ILINT validates the metadata and, if it exists, the default collection for a specific target type using a validating parser. ILINT uses all target properties (both static and dynamic) as defined in the targets.xml file.

Heuristic Constraint Checks‘

- Are too many metrics being collected? If more than 50 metrics are collected for a given target type, ILINT generates a warning message.
- Are too many targets being monitored? If more than 50 targets are being monitored for a given target type, ILINT generates a warning message.
- Are metric collections intervals too short? If a specific metric collection interval is less than five minutes, ILINT generates a warning message.

Target Metadata Limitations Based on Management Repository Constraints

- Verify that column name widths do not exceed 64 characters.
- Verify that no more than 5 key columns have been set.
- Check for legal characters in metadata name, type, metric name, and column names.

4.1.1.2 Dynamic Validation

Dynamic validation involves running specific metrics through the fetchlets to validate correctness and view output.

Dynamic validation allows you to check the correctness of the data collected for a target type. ILINT performs dynamic XML validation by executing all metrics defined for the new target and displaying viewable output in much the same way as the metric browser. For more information about the metric browser, see "Validate your new target type definitions." on page 2-10.

While performing dynamic validation, ILINT executes the specified metric and checks for errors using all properties (dynamic and static). ILINT then generates an output table containing:

- Target list
- Metric list for each target
- Metric results

4.1.1.3 Metadata Version Checking

In order for the Enterprise Manager Grid Control console to display metrics for a specific target type, syntactic rules must be followed. ILINT can also be used to verify that the current version of a metadata file is compatible with an earlier version of the same metadata file. This compatibility check is stringent: Two target metadata files being compared by ILINT must be syntactically identical.

4.1.2 ILINT Output

ILINT generates two types of messages, each directed to different output file streams: STDOUT and STDERR.

Message Types

- Informational/Warning--Generated when the XML is technically accurate, but may cause data collection/interpretation inaccuracies or framework performance issues. Tabular data generated during dynamic validation is also part of this message class.

  Informational and Warning messages are printed to STDOUT.

- Error--The XML is technically inaccurate or parameters specified violate management repository constraints.

  Error messages are printed to STDERR.

Specifying Separate Output Files

You can specify separate STDOUT and STDERR output files when running ILINT. File specification may differ depending on the operating system and/or shell used for your environment.

Example 4–1 Specifying STDOUT and STDERR (ksh, bash, and sh)

```
emctl ilint ... 1>my_stdout_file 2>my_stderr_file
```

Example 4–2 Specifying STDOUT and STDERR (csh and tcsh)

```
(emctl ilint ... >my_stdout_file) >&my_stderr_file
```

Generating Syntactically Formatted XML Output

ILINT optionally allows you to generate a syntactically consistent view of a metadata file. For more information on ILINT options and generating valid XML, see "Generating Syntactically Correct XML" on page 4-6.

4.2 Before Using ILINT

In order to validate an XML file, ILINT needs to know the location of the DTD (Document Type Declaration) associated with the XML file being validated. Make sure that the DOCTYPE directive specifying the relative path to the appropriate DTD is included in each XML file.

The following DTDs are used within the Enterprise Manager framework:

- $AGENT_HOME/sysman/admin/dtds/TargetMetadata.dtd
- $AGENT_HOME/sysman/admin/dtds/TargetCollection.dtd
- $AGENT_HOME/sysman/admin/dtds/TargetInstance.dtd

The following example shows a DOCTYPE declaration that should be used when testing a target collection XML file located in the $AGENT_HOME/sysman/admin/default_collection directory or an instance specific collection file in the $AGENT_HOME/sysman/admin/collection directory.

Example 4–3 DOCTYPE Declaration

```
<!DOCTYPE TargetCollection SYSTEM "../dtds/TargetCollection.dtd">
```

Private external DTDs (DTDs shared between multiple documents and intended for use by a single author or group of authors) are identified by the keyword SYSTEM. In this case, anyone having the appropriate OS privileges to the machine running the Management Agent is considered an author.

4.3 Using ILINT

ILINT is part of the Enterprise Manager command line utility and is accessed as an emctl option. ILINT uses the following syntax:

emctl ilint [-o <test_name> [-p]] -m <target_metadata.xml> [-m2 <compare_metadata.xml][-c <target_collection.xml>] [-i <target_instance.xml>] [-t <target_name>] [-r <metric_name>] [-d <0|1|2|3>]

Table 4–1 ILINT Options

Option	Description
-o <test_name>	Automatically locates the metadata directory, default collection directory and collection file, and the targets.xml file. When the "-o" option is specified, the following applies: ■ Just a file name rather than a full path and file name should be used when specifying the value for any of the options: -m, -m2, -c, -i ■ Normally, option -c (target collection file) should be omitted. ■ Normally, option -i (target instance file) should be omitted. ■ If -p is specified, the formatted metadata output file(s) have the test_name as their prefix.
-p	Generate syntactically formatted XML for the metadata file. This option can only be used if the -o option is specified.
-m <target_metadata.xml>	Full path and file name of the metadata file to be checked. Note: If the -o option is specified, the value must be just a file name--the full path must be omitted. This is the only required option to ILINT.
-m2 <compare_metadata.xml>	Full path and file name of a second metadata file to be compared against the target metadata file specified by the -m option. This option can be used to check version compatibility between metadata files. Note: If the -o option is specified, the value must be just a file name--the full path must be omitted.
-c <target_collection.xml>	Full path and file name of the target collection file. Note: If the -o option is specified, the value must be just a file name--the full path must be omitted.
-i <target_instance.xml>	Full path and file name of the target instance file. Note: If the -o option is specified, the value must be just a file name--the full path must be omitted.
-t <target_name>	Name of the target (as specified in the targets.xml file) to be validated.
-r <metric_name>	Name of the metric to be executed.

Table 4–1 (Cont.) ILINT Options

Option	Description
-d (0-3)	Sets the debug level (default value is 1). This option specifies the type of information supplied by ILINT. ■ 0=Error messages only. ■ 1=Warning and Error messages. ■ 2=Informational, Warning, and Error messages. ■ 3=All messages: Debug, Informational, Warning, and Error messages.

4.3.1 ILINT Examples

The following examples demonstrate common ILINT usage scenarios discussed previously.

4.3.1.1 Static Validation

Usage

```
emctl ilint
-o <test_name>
-d <0-3>
-i targets.xml
-m <target_metadata.xml>
-t <target_name>
```

Example 4–4 Validating a Target Metadata File

```
> emctl ilint -o meta_test -d 0 -i targets.xml -m my_database.xml -t my_database3
```

The metadata directory and file (-m my_database.xml) is located automatically and associated with the test name "meta_test" (-o meta_test). The metadata file is validated against the database "my_database3" (-t my_database3) which is defined in the targets.xml file. Only error messages are returned (-d 0).

Example 4–5 Validating a Target Collection File

```
> emctl ilint -o collect_test -d 3 -i targets.xml -m co_database.xml -c my_
collection.xml -t my_database3
```

The collection directory and file (-c my_collection.xml) is located automatically and associated with the test name "collect_test" (-o collect_test). Because this validation is run in full debug mode (-d 3), all error, warning, and informational messages are returned.

Example 4–6 Limited Metadata Validation (target type metadata file validation only)

```
emctl ilint -o test_name -d 0 -i non_existent_targets.xml -d 0 -m my_target_
metadata.xml -t non_existent_target
```

As shown in this example, you can perform a limited target type metadata file XML validation by specifying a non-existent targets.xml file and a non-existent target. Limited validation quickly checks XML syntax used in the target metadata file as well as any limitations in metadata due to repository constraints.

4.3.1.2 Dynamic Validation

Usage

```
emctl ilint
-o <test_name>
-d <0-3>
-i targets.xml
-m <target_metadata.xml>
-t <target_name>
-r <metric_name>
```

Example 4–7 Validating a Metric

```
emctl ilint -o test_dynamic -d 0 -i targets.xml -m database.xml -t payroll_db -r
Database_Resource_Usage
```

In this example, ILINT validates the database metric "Database_Resource_Usage." The metadata directory and file (-m database.xml) is located automatically and associated with the test name "test_dynamic" (-o test_dynamic). This metric is validated against the target database named "payroll_db" (-t payroll_db) which is defined in the targets.xml file. Only error messages are returned (-d 0).

4.3.1.3 Checking Metadata Version Compatibility

Usage

```
emctl ilint
-o <test_name>
-d <0-3>
-i targets.xml
-m <current_target_metadata.xml>
-m2 <old_target_metadata.xml>
-t <target_name>
```

Example 4–8 Validating Target Metadata Versions

```
emctl ilint -o version_test -d 3 -i targets.xml -m database.xml -m2 database.xml
-t payroll_db
```

In this example, ILINT compares two versions of the target type metadata file "database.xml." Because the -o option is used, ILINT looks for the most recent version of "database.xml" in $AGENT_HOME/sysman/admin/metadata and the older version of this file in $AGENT_HOME/sysman/admin/metadata/old.

> **Note:** If you do not want to use the default directory locations, omit the -o <test_name> option and specify absolute pathnames for -m and -m2.

4.3.1.4 Generating Syntactically Correct XML

Usage

```
emctl ilint
-o <test_name>
-i targets.xml
-m <target_metadata.xml>
-m2 <old_target_metadata.xml> (optional)
```

`-p`
XML output is sent to a file named <test_name>.xml.tmp1

Example 4–9 Generating Syntactically Correct XML

```
emctl ilint -o test_generate -i targets.xml -m my_database.xml -p
```

The XML output from this example is sent automatically to the file "test_ generate.xml.tmp1." If the "-m2" option is specified, the XML output for the second target metadata file is sent to "test_generate.xml.tmp2"

4.3.2 Usage Notes

- If the XML/DTD cannot be parsed/validated the error messages tend to be less intuitive. The Java XML parser generates more informative messages.
- The DTD path in the DOCTYPE directive in the XML file must be relative.
- In debug mode, the full information about a metric whose collection depends on the value of a ValidIf element is not displayed.

5 Adding Job Types

By defining new job types, you can extend the utility and flexibility of the Enterprise Manager job system. Adding new job types also allows you to enhance Corrective Actions. This chapter assumes that you are already familiar with the Enterprise Manager job system.

This chapter covers the following:

5.1 About Job Types

Enterprise Manager allows you to define jobs of different types that can be executed using the Enterprise Manager job system, thus extending the number and complexity of tasks you can automate.

By definition, a job type is a specific category of job that carries out a well-defined unit of work. A job type is uniquely identified by a string. For example, *OSCommand* may be a job type that executes a remote command. You define a job type by using an XML specification that defines the steps in a job, the work (command) that each step performs, and the relationships between the steps

The following table shows some of the Enterprise Manager job types and their functions.

Table 5–1 Examples of Job Types

Job Type	Purpose
Backup	Backs up a database
Backup Management	Performs management functions such as crosschecks and deletions on selected backup copies, backup sets, or files.
CloneHome	Clones an Oracle Home directory
DBClone	Clones an Oracle database instance
DBConfig	Configure monitoring for database releases prior to 10*g*.
Export	Exports database contents/objects within an Enterprise Manager user's schemas and tables.
GatherStats	Generate and modify optimizer statistics.
OSCommand	Runs an operating system command or script.
HostComparison	Compares the configurations of multiple hosts
Import	Imports the content of objects and tables.
Load	Loads data from a non-Oracle database into an Oracle database
Move Occupant	Moves occupants of the SYSAUX tablespace to another tablespace.
Patch	Patches an Oracle product.
Recovery	Restores or recovers a database, tablespaces, datafiles, or archived logs.
RefreshFromMetalink	Allows Enterprise Manager to download patches and critical patch advisory information from the OracleMetaLink support site.
Reorganize	Rebuilds fragmented database indexes or tables, moves objects to a different tablespace, or optimizes the storage attributes of specified objects.
Multi-Task	Runs a composite job consisting of multiple tasks. See About Multi-Task Jobs.
SQLScript	Runs a SQL or PL/SQL script.

5.2 Introducing New Job Types

An Enterprise Manager job consists of a set of steps. Each step executes a command or script. The job type defines how the steps are assembled. For example, which steps execute serially, which ones execute in parallel, step order and dependencies. You can express a job type, the steps and commands in XML (see Specifying a New Job Type in XML on page 5-2), and the job system will then construct an *execution plan* from the XML specification which will allow it to execute steps in the specified order.

5.3 Specifying a New Job Type in XML

A new job type is specified in XML. The job type specification provides information to the job system about:

- What steps make up the job.

- What command or script to execute in each step.
- How the steps relate to each other. For example, whether steps execute in parallel or serially, or whether one step depends on another.
- How specific job parameters should be computed (optional).
- What locks, if any, a running job execution should attempt to acquire, and what should happen if the locks are not available.
- What privileges users need to have in order to submit a job of that type.

The XML job type specification is then added to a Management Plug-in archive. Once the Management Plug-in is added to Enterprise Manager, the job system will have enough information to schedule the steps of the job, as well as what to execute in each step.

Example 5–1, "Job Type DTD" shows the DTD that defines the XML that will be used to specify a job type.

Example 5–1 Job Type DTD

```
<!ELEMENT jobtype (stepset)>
<!ATTLIST jobtype name CDATA #REQUIRED>
<!ATTLIST jobtype version CDATA #REQUIRED>
<!ATTLIST jobtype agentBound (true|false) false>
<!ELEMENT stepset (step|stepset|job)+>
<!ATTLIST stepset type (serial|parallel|iterativeSerial|iterativeParallel)
#REQUIRED>
<!ATTLIST stepset ID CDATA #REQUIRED>
<!ATTLIST stepset successOf IDREF #IMPLIED>
<!ATTLIST stepset failureOf IDREF #IMPLIED>
<!ATTLIST stepset abortOf IDREF #IMPLIED>
<!ATTLIST stepset stepsetStatus CDATA #IMPLIED>
<!ATTLIST stepset switchVarName CDATA #IMPLIED>
<!ATTLIST stepset switchCaseVal CDATA #IMPLIED>
<!ATTLIST stepset restartMode (always|failure) failure>
<!-- For iterative serial stepsets only: specify whether the iterative serial
   - step set should halt at the first abort/failure in one of the child stepset
   - executions.
-->
<!ATTLIST stepset iterateHaltOnFailure (true|false) false>
<!-- For iterative stepsets, the vector parameter to iterate over -->
<!ATTLIST stepset iterateParam CDATA #IMPLIED>
<!-- For iterative stepsets only: the value to filter values in the vector
parameter by -->
<!ATTLIST stepset iterateParamFilyer CDATA #IMPLIED>
<!ELEMENT job (target_list), (param_list)?>
<!ATTLIST job type CDATA #REQUIRED>
<!ATTLIST job ID CDATA #REQUIRED>
<!ATTLIST job successOf IDREF #IMPLIED>
<!ATTLIST job failureOf IDREF #IMPLIED>
<!ATTLIST job switchCaseVal CDATA #IMPLIED>
<!ATTLIST job restartMode (always|failure) failure>
<!ELEMENT targetList (target)*>
<!ATTLIST targetList allTargets (true|false) false>
<!ELEMENT target EMPTY>
<!ATTLIST target name CDATA #REQUIRED>
<!ATTLIST target type CDATA #REQUIRED>
<!ELEMENT step (param_list)?>
<!ATTLIST step ID CDATA #REQUIRED>
<!ATTLIST step successOf IDREF #IMPLIED>
```

```
<!ATTLIST step failureOf IDREF #IMPLIED>
<!ATTLIST stepset abortOf IDREF #IMPLIED>
<!ATTLIST stepset switchCaseVal CDATA #IMPLIED>
<!ATTLIST step command CDATA #REQUIRED>
<!ATTLIST step restartMode (always|failure) failure>
<!ELEMENT paramList (param)+>
<!ATTLIST paramList allParams (true|false) false>
<!-- A scalar param is just PCDATA. A vector param is
     represented as a set of parameterValue tags, one
     after another
-->
<!ELEMENT param ((#PCDATA)|(parameterValue)*)>
<!ATTLIST param name CDATA #REQUIRED>
<!ATTLIST param encrypted (true|false) false>
<ELEMENT parameterValue (#PCDATA)>
<!-- The following tags deal with how parameters can be fetched -->
<!ELEMENT paramSource (sourceParams)?,(#PCDATA)>
<!-- The type of the parameter source: pre-built sources
     are sql, credential and user
 -->
<!ATTLIST paramSource sourceType CDATA #REQUIRED>
<!-- The names of the parameters that are to be fetched -->
<!ATTLIST paramSource paramNames CDATA #REQUIRED>
<!-- Set this to true if paramSource is "user" and the parameter
     is a required parameter
-->
<!ATTLIST paramSource required (true|false) false>
<!-- Set to true to indicate that the parameter must be stored encrypted -->
<!ATTLIST paramSource encrypted (true|false) false>
<!ELEMENT sourceParams (sourceParam)+>
<!ELEMENT sourceParam EMPTY>
<!ATTLIST sourceParam name CDATA #REQUIRED>
<!ATTLIST sourceParam value CDATA #REQUIRED>
```

5.3.1 Job Type Categories

Depending on how a specific job type performs tasks on targets to which it is applied, a job type will fall into one of the following categories :

- **Single-Node**: A single-node job type is a job type that executes the same set of steps in parallel on every target on which the job is run. Typically, the target lists for these job types is not fixed: They can take any number of targets. Examples of single-node job types are the *OSCommand* job type, which executes an OS command or script on all of its targets, and the *SQL* type, which executes a specified SQL script on all of its targets.
- **Multi-Node/Combination**: A multi-node job type is a job type that performs different, possibly inter-related tasks on multiple targets. Such job types typically operate on a fixed set of targets. For example, a *Clone* job that clones an application schema might require two targets: a source database and a target database.

> **Note:** Iterative stepsets may be used for multi-node and combination job types to repeat the same activity over multiple targets.

5.3.2 Agent-Bound Job Types

An Agent-bound job type is one whose jobs cannot be run unless the Agent of one or more targets in the target list is functional and responding. A job type that fits this category must declare itself to be Agent-bound by setting the `agentBound` attribute of the `jobType` XML tag to true. If a job type is Agent-bound, the job system does not schedule any executions for the job type if one or more of the Agents corresponding to the targets in the target list of the job execution are down (or not responding); the job (and all its scheduled steps) are set to a special state called *Suspended/Agent down*. The job is kept in this state until the Enterprise Manager repository tier detects that the emd has come back up. At this point the job (and it's steps) are set to scheduled status again, and the job can now execute. By declaring their job types to be Agent-bound, a job-type writer can ensure that the job system will not schedule the job when it has detected that the Agent is down.

Note: Single-node job types are Agent-bound by default while multi-node job types are not.

If an Agent-bound job has multiple targets in its target list, it is marked as suspended even if one of the Agents goes down.

A good example of an Agent-bound job type would be the `OSCommand` job type, which executes an `OSCommand` using the Agent of a specified target. Note, however, that not all job types are Agent-bound: a job type that executes SQL in the repository is not Agent-bound.

Oracle Enterprise Manager has a heartbeat mechanism which enables the repository tier to quickly determine when a remote emd goes down. Once an emd is marked as "down", all Agent-bound job executions that have that emd as one of the targets in their target list are marked "Suspended/Agent Down". There is, however, still a possibility that the job system might try to dispatch some remote operations between when the emd went down and when the repository detects the fact. In such cases, when the step executes, if the Agent cannot be contacted, then the step is set back to SCHEDULED state, and is retried by the job system. The series of retries continues until the heartbeat mechanism marks the node as down, at which point the job is suspended.

Once a job is marked as Suspended/Agent Down, by default the job system keeps the job in that state until the emd comes back up. There is a parameter called the *execution timeout*, which if defined, can override this behavior. The execution timeout is the maximum amount of time (in hours) that the job will be kept in suspended state; if the Agent is not back up within this interval, the job (and it's suspended steps) are all set to ABORTED state. Note that the execution timeout is an attribute of a job, not a job type. The execution timeout can be set by using one of the flavors of submit_job in the **Commands** section on page 5-8.

Note that the only way for a job execution that is in Suspended/Agent down state to resume is for the Agent(s) to come back up. The `resume_execution()` APIs cannot be used to resume the job.

5.3.3 Job Steps

The unit of execution in a job is called a *step*. A step has a *command*, which determines what work the step will be doing. Each command has a java class, called a *command executor,* that implements the command. A command also has a set of parameters, which will be interpreted by the command executor. The job system will offer a fixed

set of pre-built commands, such as the *remote operation* command (which executes a command remotely), the *file transfer* command that transfers a file between two Agents, and a *get file* command that streams a log file produced on the Agent tier into the Management Repository).

Steps are grouped into sets called *stepsets.* Stepsets can contain steps, or other stepsets and can be categorized into the following types:

Serial Stepsets: Serial stepsets are step sets whose steps execute serially, one after another. Steps in a serial stepset can have dependencies on their execution. For example, a job can specify that step S2 executes only if step S1 completes successfully, or that step S3 executes only if S1 fails. Steps in a serial stepset can have dependencies only on other steps or stepsets within the same stepset. By default, a serial stepset is considered to complete successfully if the last step in the stepset completed successfully. It is considered to have aborted/failed if the last step in the stepset was aborted. This behavior may be overridden by using the *stepsetStatus* attribute. Overriding is allowed only when the step is not a dependency on another (no *successOf*/*failureOf*/*abortOf* attribute).

Parallel Stepsets: Parallel step sets are stepsets whose steps execute in parallel (execute simultaneously). Steps in a parallel stepset cannot have dependencies. A parallel stepset is considered to have succeeded if all the parallel steps in it have completed successfully. It is considered to have aborted if any step within it was aborted. By default, a parallel stepset is considered to have failed if one or more of its constituent steps failed, and no steps were aborted. By default, a parallel stepset is considered to have failed if one or more of its constituent steps failed, and no steps were aborted. This behavior can be overridden by using the stepsetStatus attribute..

Iterative Stepsets: Iterative stepsets are special stepsets that iterate over a vector parameter. The target list of the job is a special case of a vector parameter, called `job_target_names`. An iterative stepset "iterates" over the target list or vector parameter (as the case may be), and essentially executes the stepset *N* times, once for each value of the target list or vector parameter. Iterative stepsets can execute in parallel (*N* stepset instances execute at simultaneously), or serially (*N* stepset instances are scheduled serially, one after another). An iterative stepset is said to have succeeded if all its *N* instances have succeeded. Otherwise, it is said to have aborted if at least one of the *N* stepsets aborted. It is said to have failed if at least one of the *N* stepsets failed and none were aborted. Steps within each iterative stepset instance execute serially, and can have serial dependencies, similar to those within serial stepsets. Iterative serial stepsets have an attribute called `iterateHaltOnFailure`. If this is set to true, the stepset halts at the first failed or aborted child iteration. By default, all iterations of an iterative serial stepset execute, even if some of them fail (`iterateHaltOnFailure=false`).

Switch Stepsets: Switch stepsets are stepsets where only one of the steps in the stepset is executed based on the value of a specified job parameter. A switch stepset has an attribute called `switchVarName` , which is a job (scalar) parameter whose value will be examined by the job system to determine which of the steps in the stepset should be executed. Each step in a switch stepset has an attribute called `switchCaseVal` , which is one of the possible values the parameter specified by `switchVarName` can have. The step in the switch stepset that is executed is the one whose `swithCaseVal` parameter value matches the value of the `switchVarName` parameter of the switch stepset. Only the selected step in the switch stepset is executed. Steps in a switch stepset cannot have dependencies with other steps or stepsets within the same stepset or outside. By default, a switch stepset is considered to complete successfully if the selected step in the stepset completed successfully. It is considered to have aborted/failed if the selected step in the stepset was aborted/failed. Also, a switch stepset will succeed if no step in the stepset was selected. For example, there is a switch stepset with two steps,

S1 and S2. One can specify that `switchVarName` is "sendEmail" and specify `switchCaseVal` for S1 to be true and for S2 to be false. If the job is submitted with the job parameter `sendEmail` set to true, then S1 will be executed. If the job is submitted with the job parameter `sendEmail` set to false , then S2 will be executed. If the value of `sendEmail` is anything else, the stepset would still succeed but instead do nothing.

Nested Jobs: One of the steps in a stepset may itself be a reference to another job type. A job type can therefore include other job types within itself. However, a job type cannot reference itself. Nested jobs are a convenient way to reuse blocks of functionality. For example, performing a database backup could be a job in its own right, with a complicated sequence of steps; however, other job types (such as *patch* and *clone*) might use the backup facility as a nested job. With nested jobs, the job type writer can choose to pass all the targets of the containing job to the nested job, or only a subset of the targets. Likewise, the job type can specify whether the containing job should pass all its parameters to the nested job, or whether the nested job has it's own set of parameters (derived from the parent job's parameters). Please see the examples section for examples on how to use nested job types. The status of a nested job is determined by the status of the individual steps and stepsets (and possibly other nested jobs) within the nested job.

Affecting the Status of a Stepset

The default algorithm by which the status of a stepset is computed from the status of its steps can be altered by the job type, using the `stepsetStatus` attribute of a stepset. By setting `stepsetStatus` to the name (ID) of a step/stepset/job contained within it, a stepset can indicate that the status of the stepset depends on the status of the specific step/stepset/job named in the `stepStatus` attribute. This feature is useful if the author of a job type wishes a stepset to succeed even if certain steps within it fail. A good example would be a step that runs as the final step in a stepset in a job that sends e-mail about the status of the job to a list of administrators. The actual status of the job should be set to the status of the step (or steps) that actually did the work, not the status of the step that sent email. Note that only steps that are unconditionally executed can be named in the `stepsetStatus` attribute: a step/stepset/job that is executed as a `successOf` or `failureOf` dependency cannot be named in the `stepsetStatus` attribute.

Passing Job Parameters

The parameters of the job can be passed to steps by enclosing the parameter name in %: these are called *placeholders*. for example, `%patchNo%` would represent the value of a parameter named `patchNo`. The job system will substitute the value of this parameter when it is passed to the command executor of a step. Placeholders can be defined for vector parameters as well, by using the [] notation: For example, the first value of a vector parameter called `patchList` would be referenced as `%patchList%[1]`, the second would be `%patchList%[2]`, and so on.

The job system provides a pre-defined set of placeholders that can be used. These are always prefixed by `job_`. The following placelholders are provided:

- job_iterate_index: The index of current value of the parameter in an iterative stepset, when iterating over any vector parameter. The index refers to the closest enclosing stepset only. In case of nested iterative stepsets, the outer iterate index cannot be accessed.
- job_iterate_param: The name of the parameter being iterated over, in an iterative stepset.

- job_target_names[n]: The job target name at position n. For single-node jobs, the array would always be only of size 1 and refer only to the current node the job is execution on, even if the job was submitted against multiple nodes
- job_target_types[n]: The type of the job target at position n. For single-node jobs, the array would always be only of size 1 and refer only to the current node the job is execution on, even if the job was submitted against multiple nodes
- job_name: The name of the job.
- job_type: The type of the job.
- job_owner: The Enterprise Manager user that submitted the job.
- job_id: The job id. This is a string representing a GUID.
- job_execution_id: The execution id. This is a string representing a GUID.
- job_step_id: The step id. This is an integer.

In addition to the above placeholders, the following target-related placeholders are also supported:

- emd_root: The root location of the emd install
- perlbin: The location of the (Enterprise Manager) Perl install.
- scriptsdir: The location of emd-specific scripts

The above placeholders are not interpreted by the job system, but by the Agent. For example, when `%emd_root%` is used in the `remoteCommand` or `args` parameters of the `remoteOp` command, or in any of the file names in the `putFile` , `getFile` and `transferFile` commands, the Agent substitutes the actual value of the Agent root location for this placeholder.

Job Step Output and Errors

A step consists of a *status* (which indicates whether it succeeded, failed, or aborted); some *output*, which is the log of the step; and an *error message*. If a step failed, the command executed by the step could indicate the error in the error message column. By default, the standard output **and** standard error of a asynchronous remote operation is set to be the output of the step that requested the remote operation. A step can choose to insert error messages by either using the `getErrorWriter()` method in `CommandManager` (synchronous), or by using the `insert_step_error_message` API in the `mgmt_jobs` package (typically, this would be called by a remotely executing script in a command channel).

5.4 Commands

This section describes available commands and associated parameters. Note that targets of any type can be provided for the target name and target type parameters below: The job system will automatically identify and contact the Agent that is monitoring the specified targets.

5.4.1 Remote Operations

The remote operation command has the identifier "`remoteOp`". It takes the following parameters:

- **remoteCommand:** The path name to the executable/script to execute (for example, /usr/local/bin/perl)
- **args:** A comma-separated list of arguments to the remoteCommand

- **targetName:** The name of the target to execute the command on. Note that placeholders can be used to represent targets, see below.
- **targetType**: The type of the target to execute the command on.
- **username:** The OS user to execute as
- **password:** The password used to authenticate the user
- **executeSynchronous:** This option defaults to false: a remote command always executes asynchronously on the Agent, and the status of the step is updated after the command is done executing. If this is set to true, the command executes synchronously, waiting until the Agent completes the process. Typically, this parameter would be set to true for quick, short-lived remote operations (such as starting up a listener). For remote operations that take a long time to execute, this parameter should always be set to false.
- **successStatus:** A comma-separated list of integer values that determines the "success" of the step. If the remote command returns any of these numbers as the exit status, the step is considered successful. The default is zero. These values are only applicable when `executeSynchronous` is set to true.
- **failureStatus:** A comma-separated list of integer values that determines the "failure" of the step. If the remote command returns any of these numbers as the exit status, the step is considered to have failed. The default is all non-zero values. These values are only applicable when `executeSynchronous` is set to true.
- **input:** If specified, this is passed as standard input to the remote program.
- **outputType:** `outputType` specifies what kind of output the remote command is expected to generate. This can have two values, *normal* (the default) or *command*. Normal output is output that is stored in the log corresponding to this step and is not interpreted in any way. Command output is output that could contain one or more "command blocks", command-blocks are XML sequences that map to pre-registered SQL procedure calls. By using the command output option, a remote command can generate command blocks that can be directly loaded into schema in the Enterprise Manager repository database.

The standard output generated by the executed command is stored by the job system as the output corresponding to this step.

5.4.2 fileTransfer

The file transfer command has the identifier `fileTransfer`. It transfers a file from one Agent to another. It can also execute a command on the source Agent and transfer its standard output as a file to the destination Agent, or as standard input to a command on the destination Agent. The `fileTransfer` command is always asynchronous. It takes the following parameters:

- **sourceTargetName:** The name of the target corresponding to the source Agent.
- **sourceTargetType**: The name of the target corresponding to the source Agent.
- **destTargetName**: The name of the target corresponding to the destination Agent.
- **destTargetType**: The type of the target corresponding to the destination Agent.
- **sourceFile:** The file to be transferred from the source Agent
- **sourceCommand:** The command to be executed on the source Agent. If this is specified, then the standard output of this command is streamed to the destination Agent. `sourceFile` and `sourceCommand` cannot both be specified.

- **sourceArgs:** A comma-separated set of command-line parameters for the `sourceCommand`.
- **sourceUsername:** The username to use on the source Agent. This is a user that has at least read permissions on the source file.
- **sourcePassword:** The password to use on the source Agent
- **destFile:** The location/filename where the file is to be stored on the destination Agent
- **destCommand:** The command to be executed on the destination emd. If this is specified, then the stream generated from the source emd (whether from a file or from a command) is sent to the standard input of this command. `destFile` and `destCommand` cannot both be specified.
- **destArgs:** A comma-separated set of command-line parameters for the destCommand.
- **destUsername:** The (OS) username to use on the destination Agent. This is a user that has at least write permissions on the destination location.
- **destPassword:** The password to use on the destination Agent

The `fileTransfer` command succeeds (and returns a status code of 0) if the file was successfully transferred between the Agents. If there was an error, it returns error codes appropriate to the reason for failure.

5.4.3 putFile

The `putFile` command has the identifier `putFile`. It provides the capability to transfer large amounts of data from the Enterprise Manager repository to a file on the Agent. The data transferred could come from a blob in the repository, or a file on the file system, or could be embedded in the specification (inline).

If a file is being transferred, the location of the file must be accessible from the repository installation. If a blob in a database is being transferred, it must be in a table in the repository database that is accessible to the repository schema user (typically mgmt_rep).

The `putFile` command requires the following parameters:

- **sourceType:** The type of the source data. This may be "sql"or "file" or "inline".
- **targetName:** The name of the target where the file is to be transferred (destination Agent).
- **targetType**: The type of the destination target.
- **sourceFile:** The file to be transferred from the repository, if the `sourceType` is set to "fileSystem". This must be a file that is accessible to the repository installation.
- **sqlType:** The type of SQL data (if the `sourceType` is set to "sql"). Valid values are CLOB, BLOB.
- **accessSql:** A SQL statement that is used to retrieve the blob data (if the `sourceType` is set to "sql"). For example, " select output from `my_output_table` where blob_id=%blobid%"
- **destFile:** The location/filename where the file is to be stored on the destination Agent.

- **contents**: If the sourceType is set to "inline", this parameter contains the contents of the file. Note that the text could include placeholders for parameters in the form %param%
- **username:** The (OS) username to use on the destination Agent. This is a user that has write permissions on the destination location.
- **password:** The password to use on the destination Agent

The `putFile` command succeeds if the file was transferred successfully, and the status code is set to 0. On failure, the status code is set to an integer appropriate to the reason for failure.

5.4.4 getFile

The `getFile` command has the identifier "getFile". It transfers a file from an Agent to the repository. The file is stored as the output of the step that executed this command.

The `getFile` command has the following parameters:

- **sourceFile:** The location of the file to be transferred on the Agent.
- **targetName:** The name of the target whose Agent will be contacted to get the file.
- **targetType**: The type of the target.
- **username:** The (OS) username to use on the Agent. This is a user that has read permissions on the file to be transferred.
- **password:** The password to use on the Agent.

The `getFile` command succeeds if the file was transferred successfully, and the status code is set to 0. On failure, the status code is set to an integer appropriate to the reason for failure.

5.5 Command Error Codes

The `remoteOp`, `putFile`, `fileTransfer` and `getFile` commands return the following error codes. In the messages below, "command process" refers to a process that the Agent executes that actually execs the specified remote command, and grabs the standard output and standard error of the executed command. On a Unix install, this process is called *nmo*, and lives in $EMD_ROOT/bin. It must be SETUID to root before it can be used successfully. This does not pose a security risk since *nmo* will not execute any command unless it has a valid username and password).

0: No error

1: Could not initialize core module. Most likely, something is wrong with the install or environment of the Agent.

2: The Agent ran out of memory.

3: The Agent could not read information from its input stream.

4: The size of the input parameters was too large for the Agent to handle.

5: The command process was not `setuid` to root. (Every Unix Agent install has an executable called *nmo*, which must be `setuid` root)

6: The specified user does not exist on this system.

7: The password was incorrect.

8: Could not run as the specified user.

9: Failed to fork the command process (nmo).

10: Failed to execute the specified process.

11: Could not obtain the exit status of the launched process.

12: The command process was interrupted before exit.

13: Failed to redirect the standard error stream to standard output.

5.6 Executing Long-Running Commands at the OMS

The job system allows integrators to write commands that perform their work at the Management Service level. For example, a command that reads two LOBs from the database and performs various transformations on them, and writes them back. The job system expects such commands to implement an (empty) interface called `LongRunningCommand`, which is an indication that the command executes synchronously on the middle tier, and could potentially execute for a long time. This will allow a component of the job system called the dispatcher to schedule the long-running command as efficiently as possible, in such as way as to not degrade the throughput of the system.

Configuring the Job Dispatcher to Handle Long-Running Commands

The dispatcher is a component of the job system that executes the various steps of a job when they are ready to execute. The command class associated with each step is called, and any asynchronous operations requested by it are dispatched, a process referred to as dispatching a step. The dispatcher uses thread-pools to execute steps. A thread-pool is a collection of a specified number of worker threads, any one of which can dispatch a step. The job system dispatcher uses two thread-pools: a short-command pool for dispatching asynchronous steps and short synchronous steps, and a long-command pool for dispatching steps that have long-running commands. Typically, the short-command pool will have a larger number of threads (say, 25) compared to the long-running pool (say, 10). The theory is that long-running middle-tier steps will be few compared to more numerous, short-running commands. However, the sizes of the two pools will be fully configurable in the dispatcher to suit the job mix at a particular site. Since multiple dispatchers can be run on different nodes, the site administrator will be able to even dedicate a dispatcher to only dispatch long-running or short-running steps.

5.7 Specifying Parameter Sources

By default, the job system expects the integrators to provide values for all job parameters either when the job is submitted, or at execution time (by adding/updating parameters dynamically). Typically, an application would supply these parameters in one of three-ways:

- By asking the user of the application at the time of submitting the job
- By fetching parameter values from application-specific data (such as a table) and then inserting them into the job parameter list
- By generating new parameters dynamically via the command blocks in the output of a remote command; these could be used by subsequent steps.

The job system offers the concept of *parameter sources* so that integrators can simplify the amount of application-specific code they have to write to fetch and populate job or step parameters (such as the second category above). A parameter source is a mechanism that the job system uses to fetch a set of parameters either when a job is

submitted, or when it is about to start executing. The job system supports SQL (a PL/SQL procedure to fetch a set of parameters), *credential* (retrieval of username and password information from the Enterprise Manager credentials table) and *user.* Integrators can use these pre-built sources to fetch a wide variety of parameters. When the job system has been configured to fetch one or more parameters using a parameter source, the parameter(s) need not be specified in the parameter list to the job when a job is submitted: the job system will automatically fetch the parameters and add them to the parameter list of the job.

A job type can embed information about the parameters it needs fetched by having an optional `paramInfo` section in its XML specification. The following is a snippet of a job type that executes a SQL query on an application-specific table called `name_value_pair_table` to fetch three parameters, a, b and c.

```
<jobType version="1.0" name="OSCommand" >
<paramInfo>
    <!-- Set of scalar params -->
    <paramSource paramNames="a,b,c" sourceType="sql" overrideUser="true">
        select name, value from name_value_pair_table where
            name in ('a', 'b', 'c');
    </paramSource>
</paramInfo>
.... description of job type follows ....
</jobType>
```

As can be seen from the example, a `paramInfo` section consists one or more `paramSource` tags. Each `paramSource` tag references a parameter source that can be used to fetch one or more parameters. The `paramNames` attribute is a comma-separated set of parameter names that the parameter source is expected to fetch. The sourceType attribute indicates the source that will be used to fetch the parameters (one of *sql*, *credential* or *user*). The `overrideUser` attribute, if set to true, indicates that this parameter-fetching mechanism will always be used to fetch the value of the parameter(s), even if the parameter was specified by the user (or application) at the time the job was submitted. The default for the `overrideUser` attribute is false: the parameter source mechanism will be disabled if the parameter was already specified when the job was submitted. A parameter source could have additional source-specific properties that describe the fetching mechanism in greater detail: these will be described in the following sections.

5.7.1 SQL Parameter Source

The SQL parameter source allows the integrator to specify a SQL query or a PL/SQL procedure that will fetch a set of parameters.

5.7.1.1 Using a SQL Query to Fetch a Set of Scalar Parameters

By default, all parameters specified in the `paramNames` attribute of the `paramSource` tag are assumed to be scalar. Scalar parameters can be fetched by an arbitrary SQL query. The SQL query should generate a cursor that has exactly two columns: the first column should reference the parameter name, and the second column should reference the parameter value. In the example below, the following query fetches from an application-specific table called `name_value_pair_table`. The table is assumed to have two columns, 'name' and 'value', that hold the names and values of needed application parameters, respectively.

```
<paramInfo>
    <!-- Set of scalar params -->
    <paramSource paramNames="a,b,c" sourceType="sql">
```

```
        select name, value from name_value_pair_table where
            name in ('a', 'b', 'c') and app_name='app';
    </paramSource>
</paramInfo>
```

5.7.1.2 Using a SQL Query to Fetch a Mixture of Scalar and Vector Parameters

Assume you have a table called `parameter_values` that holds both scalar and vector parameter values, one value to a row, as shown below:

Table 5–2 Example Table Containing Scalar and Vector Parameter Values

Parameter_Name	Parameter_Value	Parameter_Index
vector1	pv1	1
vector1	pv2	2
vector1	pv3	3
scalar1	s1	1
scalar2	s2	1

The above table holds a vector parameter called vector1 that has three values (pv1, pv2, pv3), and two scalar parameters, scalar1 and scalar2. For vector parameters, the parameter_index column imposes an ordering on the parameters. The following block fetches the parameters using one single query. The source parameters `scalarParams` and `vectorParams` specify which of the parameters are scalar or vector, respectively. In the example below, they tell the job system that the parameter vector1 is a vector parameter, and that the parameters scalar1 and scalar2 are scalar parameters. This allows the job system to appropriately construct the parameters. Note that if a parameter is not specifically included in a scalar or vector directive, it is assumed to be scalar.

```
<paramInfo>
    <!-- Mixture of scalar and vector parameters -->
    <paramSource paramNames="vector1,scalar1,scalar2" sourceType="sql">
        <sourceParam name="scalarParams" value="scalar1, scalar2" />
        <sourceParam name="vectorParams" value="vector1" />
        select parameter_name, parameter_value from parameter_values where
            parameter_name in ('scalar1', 'scalar2', 'vector1') order by
                parameter_name, parameter_index;
    </paramSource>
</paramInfo>
```

The above table holds a vector parameter called vector1 that has three values (pv1, pv2, pv3), and two scalar parameters, scalar1 and scalar2. For vector parameters, the parameter_index column imposes an ordering on the parameters. The following block fetches the parameters using one single query. The source parameters scalarParams and vectorParams specify which of the parameters are scalar or vector, respectively. In the example below, they tell the job system that the parameter vector1 is a vector parameter, and that the parameters scalar1 and scalar2 are scalar parameters. This allows the job system to appropriately construct the parameters. Note that if a parameter is not specifically included in a scalar or vector directive, it is assumed to be scalar.

```
<paramInfo>
    <!-- Mixture of scalar and vector parameters -->
    <paramSource paramNames="vector1,scalar1,scalar2" sourceType="sql">
```

```
        <sourceParam name="scalarParams" value="scalar1, scalar2" />
        <sourceParam name="vectorParams" value="vector1" />
        select parameter_name, parameter_value from parameter_values where
            parameter_name in ('scalar1', 'scalar2', 'vector1') order by
                  parameter_name, parameter_index;
    </paramSource>
</paramInfo>
```

5.7.1.3 Using a PL/SQL Procedure to Fetch Scalar and Vector Parameters

You can also write a PL/SQL procedure that will fetch parameters. The PL/SQL procedure can have any number of parameters, but it must have one special input and one special output parameter reserved for the job system. The input parameter must be of the type SMP_Agent_STRING_ARRAY: it is an array of varchar2 values, which are the names of the parameters to fetch. The output parameter must be either a cursor or a MGMT_JOB_PARAM_LIST. The "special" input and output parameters must be the first and second bind parameters in the call to the procedure (see example below). Also, the `outProc` source parameter must be set to 1 and the type of the output parameter must be specified using the source parameter `sqloutparamtype` . This must be set to "**cursor**" (the output parameter is a cursor) or "**paramList**" (the output parameter is a MGMT_JOB_PARAM_LIST).

Let us illustrate these concepts with a couple of examples. Let us assume that you wrote a PL/SQL procedure that had the following signature:

```
PROCEDURE fetch_params(application_name varchar2, param_names SMP_Agent_STRING_
ARRAY,
                       param_values OUT MGMT_JOB_PARAM_LIST);
```

The "special" input and output parameters required by the job system are at the first and second bind parameter positions, respectively. The following XML block shows how you would configure the job system to fetch parameters using this procedure. Note that the job system binds the values of the input parameters and extracts the output parameter.

```
<paramInfo>
    <!-- pl/sql proc to fetch a mixture of scalar and vector parameters -->
    <paramSource paramNames="vector1,scalar1,scalar2" sourceType="sql">
        <sourceParam name="scalarParams" value="scalar1, scalar2" />
        <sourceParam name="outProc" value="1" />
        <sourceParam name="vectorParams" value="vector1" />
        <sourceParam name="sqloutparamtype" value="paramList" />
        BEGIN fetch_params('pts', :1, :2); END;
    </paramSource>
</paramInfo>
```

Next, let us assume that there exists a PL/SQL procedure that returns a cursor having two columns specifying the name and value of parameters:

```
TYPE MYCURSOR is REF CURSOR;
FUNCTION fetch_params_cursor(param_names SMP_Agent_STRING_ARRAY, params OUT
MYCURSOR);
```

The following XML block shows how you would use this procedure to fetch a set of parameters:

```
<paramInfo>
    <!-- pl/sql proc to fetch a mixture of scalar and vector parameters -->
    <paramSource paramNames="vector1,scalar1,scalar2" sourceType="sql">
        <sourceParam name="scalarParams" value="scalar1, scalar2" />
```

```
        <sourceParam name="vectorParams" value="vector1" />
        <sourceParam name="outProc" value="1" />
        <sourceParam name="sqloutparamtype" value="cursor" />
        BEGIN fetch_params_cursor(:1, :2); END;
    </paramSource>
</paramInfo>
```

Please see the examples section for more examples on specifying job parameters.

5.7.2 Credentials Parameter Source

The Enterprise Manager credentials table provides a convenient storage mechanism for credentials that an application needs to perform its tasks. Credentials are a set of name value pairs. For example, node credentials include two name-value pairs: one for the username, and another for the password. Database credentials might have three name-value pairs, one each for username, password and role. The conceptual structure of the credentials table is given below:

Target	Credential Column Name	Credential Value	User Name	Container Location
o815.dlsun966	node_username	skini	USER1	null
o815.dlsun966	node_password	hahaha	USER1	null
o815.dlsun966	patch_node_ username	patchman	SYSTEM	/private/skini
o815.dlsun966	patch_node_ password	patchpwd	SYSTEM	null
o815.dlsun966	patch_db_username	system	SYSTEM	/private/oracle
o815.dlsun966	patch_db_password	manager	SYSTEM	/private/oracle
o815.dlsun966	patch_db_role	sysdba	SYSTEM	/private/oracle

In the table above, the columns "node_username" and "node_password" are used to store node credentials for the target o815.dlsun966 for user USER1. The set of credentials columns with the "patch" prefix (such as patch_node_username, patch_ node_password) are together used to store a set of credentials that a user would need to patch a database.

Two types of credentials can be stored: user-specific and system credentials. Typically, system-specific credentials are associated with a privilege (for example, "Patch"). They apply to all users that carry out a specific operation (in this case, patching a database). User-specific credentials are associated with specific users and are typically user preferences.

Notice that credentials could have several rows associated with them. For example, the "Patch" credential for a database consists of a set of node credentials, as well as a set of database credentials (the credential columns are all prefixed with "patch_")

Some credentials may also be optionally associated with a "container location". A container location conceptually corresponds to the pathname of the appltop or Oracle home where a specific database (or application) is installed. If credentials are not associated with a container location (in general, other than application credentials, most credentials will not be), the container location can be set to null.

If an application stores the credentials it needs in the Enterprise Manager credentials table, the job system provides a parameter source that can be used to pull out values of specific credentials from the credential table. The following XML block specifies how

the node username and password could be pulled out of the credentials table for a specific target, using the patch credentials. Typically, when parameters are pulled out of the credentials table, the job type author would want to set `overrideUser` to true to avoid a user from submitting a job using a different set of credentials.

```
<paramInfo>
    <!-- Fetch params from the credentials table -->
    <paramSource paramNames="username,password" overrideUser="true"
sourceType="credentials">
      <sourceParams>
            <sourceParam name="credential_columns" value="node_username,node_
password" />
            <sourceParam name="target_names" value="dlsun966" />
            <sourceParam name="target_types" value="host" />
            <sourceParam name="credential_scope" value="system" />
      </sourceParams>
    </paramSource>
</paramInfo>
```

In the XML above, the `credential_columns` parameter is a comma-separated list of columns that must be fetched. Note that they have a one-to-one correspondence with the parameters username and password specified in the `paramNames` attribute. The job system will fetch the `node_username` value into the username parameter, and the `node_password` value into the `password` parameter.

> **Note:** The credential source always fetches into vector parameters. In the example above, the credential source would fetch into two vector parameters, username and password, each having one value.

The `credential_scope` parameter specifies whether the credentials are "system" credentials or "user" credentials. If set to "user", credentials corresponding to the user that submitted the job are pulled out. If set to "system", the system credentials are used. Note that it is not possible for the submitter of a job to use some other user's credentials.

A set of credentials can also be fetched into a set of vector parameters. In the example below, the `targetNames`, `targetTypes` and `containerPaths` attributes are comma-separated. The `containerPaths` attribute is optional; if it is not specified, the container location is not considered while fetching the credentials. If it is specified, it must have valid values for all the targets.

```
<paramInfo>
    <!-- Fetch params from the credentials table into vector parameters -->
    <paramSource paramNames="vec_usernames,vec_passwords" overrideUser="true"
sourceType="credentials">
      <sourceParams>
            <sourceParam name="credentialType" value="patch" />
            <sourceParam name="credentialColumns" value="node_username,node_
password" />
            <sourceParam name="targetNames" value="dlsun966,ap952sun" />
            <sourceParam name="targetTypes" value="host,host" />
            <sourceParam name="containerPaths"
value="/private/skini,/private/oracle" />
            <sourceParam name="credentialScope" value="system" />
      </sourceParams>
    </paramSource>
</paramInfo>
```

Finally, the target names and target types can be specified using vector parameters as well. The example below uses the `targetNamesParam` and `targetTypesParam` to specify two vector parameters that are used to provide the target names and values while fetching the credentials, which will be put into the vector parameters `vec_usernames` and `vec_passwords` , respectively. Also note the use of the `containerPathsParam` parameter. This is a job parameter that is expected to contain the corresponding container locations for each target. If a `containerPathsParam` is specified, it must have non-null values for all targets.

```
<paramInfo>
    <!-- Fetch params from the credentials table into vector parameters -->
    <paramSource paramNames="vec_usernames,vec_passwords" overrideUser="true"
sourceType="credentials">
      <sourceParams>
            <sourceParam name="credentialType" value="patch" />
            <sourceParam name="credentialColumns" value="node_username,node_
password" />
            <sourceParam name="targetNamesParam" value="job_target_names" />
            <sourceParam name="targetTypesParam" value="job_target_types" />
            <sourceParam name="containerPathsParam" value="container_paths" />
            <sourceParam name="credentialScope" value="system" />
      </sourceParams>
    </paramSource>
</paramInfo>
```

5.7.3 User Parameter Source

The job system also offers a special parameter source called "user" which indicates that a set of parameters must be supplied when a job of that type is submitted. If a parameter is declared to be of source "user" and the "required" attribute is set to "true", the job system will validate that all specified parameters in the source are provided when a job is submitted.

The user source can be evaluated at job submission time or job execution time. When evaluated at submission time, it causes an exception to be thrown if any required parameters are missing. When evaluated at execution time, it causes the execution to abort if there are any missing required parameters.

```
<paramInfo>
    <!-- Indicate that parameters a, b and c are required params -->
    <paramSource paramNames="a, b, c" required="true" sourceType="user" />
</paramInfo>
```

The user source can also be used to indicate that a pair of parameters are target parameters. For example,

```
<paramInfo>
    <!-- Indicate that parameters a, b, c, d, e, f are target params -->
    <paramSource paramNames="a, b, c, d, e, f" sourceType="user" >
        <sourceParam name="targetNameParams" value="a, b, c" />
        <sourceParam name="targetTypeParams" value="d, e, f" />
    </paramSource>
</paramInfo>
```

The above block indicates that parameters (a,d), (b,e), (c,f) are parameters that hold target information. Parameter "a " holds target names, and "d" holds the corresponding target types. Similarly with parameters "b" and "e", and "c" and "f". For each parameter that holds target names, there must be a corresponding parameter that holds target types. The parameters may be either scalar or vector.

5.7.4 Inline Parameter Source

The inline parameter source allows job types to define parameters in terms of other parameters. It is a convenient mechanism to construct parameters that can be reused in other parts of the job type. For example, the section below creates a parameter called filename based on the job execution id, presumably for use in other parts of the job type.

```
<jobType>
    <paramInfo>
        <!-- Indicate that value for parameter filename is provided inline -->
        <paramSource paramNames="fileName"  sourceType="inline" >
            <sourceParam name="paramValues" value="%job_execution_id%.log" />
        </paramSource>
    </paramInfo>
.....
    <stepset ID="main" type="serial">
        <step command="putFile" ID="S1">
             ...
             <param name="destFile">%fileName%</param>
             ...
        </step>
    </stepset>
</jobType>
```

The following example sets a vector parameter called `vparam` to be a vector of the values v1, v2, v3 and v4. Only one vector parameter at a time can be set using the inline source.

```
<jobType>
    <paramInfo>
        <!-- Indicate that value for parameter vparam is provided inline -->
        <paramSource paramNames="vparam"  sourceType="inline" >
            <sourceParam name="paramValues" value="v1,v2,v3,v4" />
            <sourceParam name="vectorParams" value="vparam" />
        </paramSource>
    </paramInfo>
.....
```

5.7.5 checkValue Parameter Source

The `checkValue` parameter source allows job types to have the job system check that a specified set of parameters have a specified set of values. If a parameter does not have the specified value, the job system will either abort or suspend the job.

```
<paramInfo>
    <!-- Check that the parameter halt has the value true. If not, suspend the job
-->
    <paramSource paramNames="halt"  sourceType="checkValue" >
        <sourceParam name="paramValues" value="true" />
        <sourceParam name="action" value="suspend" />
    </paramSource>
</paramInfo>
```

The following example checks whether a vector parameter v has the values v1,v2,v3, and v4. Only one vector parameter at a time can be specified in a checkValue parameter source. If the vector parameter does not have those values, in that order, then the job is aborted.

```
<paramInfo>
```

```
    <!-- Check that the parameter halt has the value true. If not, suspend the job
-->
    <paramSource paramNames="v"  sourceType="checkValue" >
        <sourceParam name="paramValues" value="v1,v2,v3,v4" />
        <sourceParam name="action" value="abort" />
        <sourceParam name="vectorParams" value="v" />
    </paramSource>
</paramInfo>
```

5.7.6 properties Parameter Source

The properties parameter source fetches a named set of target properties for each of a specified set of targets and stores each set of property values in a vector parameter.

The example below fetches the properties "OracleHome" and "OracleSID" for the specified set of targets (dlsun966 and ap952sun), into the vector parameters `ohomes` and `osids`, respectively. The first vector value in the `ohomes` parameter will contain the OracleHome property for dlsun966, and the second will contain the OracleHome property for ap952sun. Likewise with the OracleSID property.

```
<paramInfo>
    <!-- Fetch the OracleHome and OracleSID property into the vector params ohmes,
osids -->
    <paramSource paramNames="ohomes,osids" overrideUser="true"
sourceType="properties">
      <sourceParams>
            <sourceParam name="propertyNames" value="OracleHome,OracleSID" />
            <sourceParam name="targetNames" value="dlsun966,ap952sun" />
            <sourceParam name="targetTypes" value="host,host" />
      </sourceParams>
    </paramSource>
</paramInfo>
```

As with the credentials source, vector parameter names can be provided for the target names and types.

```
<paramInfo>
    <!-- Fetch the OracleHome and OracleSID property into the vector params ohmes,
osids -->
    <paramSource paramNames="ohomes,osids" overrideUser="true"
sourceType="properties">
      <sourceParams>
            <sourceParam name="propertyNames" value="OracleHome,OracleSID" />
            <sourceParam name="targetNamesParam" value="job_target_names" />
            <sourceParam name="targetTypes" value="job_target_types" />
      </sourceParams>
    </paramSource>
</paramInfo>
```

5.7.7 Parameter Sources and Parameter Substitution

Parameter sources are applied in the order they are specified. Parameter substitution (of the form %param%) can be used inside `sourceParam` tags, but the parameter that is being substituted must exist when the parameter source is evaluated. Otherwise, the job system will substitute an empty string in its place.

5.7.8 Parameter Encryption

The job system offers the facility of storing specified parameters in encrypted form. Parameters that contain sensitive information, such as passwords, must be stored encrypted. A job type can indicate that parameters fetched through a parameter source be encrypted by setting the encrypted attribute to true in a parameter source. For example:

```
<paramInfo>
    <!-- Fetch params from the credentials table into vector parameters; store
them encrypted -->
    <paramSource paramNames="vec_usernames,vec_passwords" overrideUser="true"
                                  sourceType="credentials" encrypted="true">
      <sourceParams>
            <sourceParam name="credentialType" value="patch" />
            <sourceParam name="credentialColumns" value="node_username,node_
password" />
            <sourceParam name="targetNames" value="dlsun966,ap952sun" />
            <sourceParam name="targetTypes" value="host,host" />
            <sourceParam name="credentialScope" value="system" />
      </sourceParams>
    </paramSource>
</paramInfo>
```

A job type can also specify that parameters supplied by the user be stored encrypted:

```
<paramInfo>
    <!-- Indicate that parameters a, b and c are required params -->
    <paramSource paramNames="a, b, c" required="true" sourceType="user"
encrypted="true" />
</paramInfo>
```

5.8 Specifying Security Information

Typically, a job type will tend to perform actions that may be considered to be "privileged", for example, patching a production database, or affecting the software installed in an Oracle home or appltop. Accordingly, such job types should only be submitted by Enterprise Manager users that have the appropriate level of privileges to perform these actions. The job system provides a section called `securityInfo`, which the author of a job type can use to specify the minimum level of privileges (system, target) the submitter of a job of that type must have. For more information about the Enterprise Manager user model and system and target privileges, see Enterprise Manager online help.

Having a `securityInfo` section allows the author of a job type to encapsulate the security requirements associated with submitting a job in the job type itself; no further code need to be written to enforce security. Also, it ensures that Enterprise Manager users cannot directly submit jobs of a specific type (using the job system APIs and bypassing the application) unless they have the set of privileges defined by the job type author.

The following shows what a typical securityInfo section looks like. Suppose you are writing a job type that clones a database. This job type will require two targets; let us say the first is a source database and the other is a destination node on which the destination database will be created. This job type will probably require that a user that submits a clone job have a CLONE FROM privilege on the source (database) and a MAINTAIN privilege on the destination (node). In addition, the user will require the CREATE TARGET system privilege in order to be able to introduce a new target into

the system. Let us assume that the job type is written so that the first target in the target list is the source and the second target in the target list is the destination. The security requirements for such a job type could be addressed as shown below:

```
<jobType>
  <securityInfo>
    <privilege name="CREATE TARGET" type="system" />
    <privilege name="CLONE FROM" type="target" evaluateAtSubmission="false" >
        <target name="%job_target_names%[1]" type="%job_target_types%[1]" />
    </privilege>
    <privilege name="MAINTAIN" type="target" evaluateAtSubmission="false">
        <target name="%job_target_names%[2]" type="%job_target_types%[2]" />
    </privilege>
  </securityInfo>
  <!-- An optional <paramInfo> section will follow here, followed by the stepset
       definition of the job
  -->
  <paramInfo>
   ....
  </paramInfo>
  <stepset ...>
  </stepset>
</jobType>
```

The `securityInfo` section is a set of `<privilege>` tags. Each privilege could be a system or target privilege, as indicated by the type attribute of the <privilege> tag. If the privilege is a target privilege, the targets that the privilege is attached to should be explicitly enumerated, or the `target_names_param` and `target_types_param` attributes should be used (as shown in the second example, below). The usual %param% notation can be used to indicate job parameter and target placeholders.

By default, all `<privilege>` directives in the `securityInfo` section are evaluated at job submission time, after all submit-time parameter sources have been evaluated. The job system throws an exception if the user does not have any of the privileges specified in the `securityInfo` section. Note that execution-time parameter sources will not have been evaluated at job submission time, so care should be taken to not use job parameters that may not have been evaluated yet. You could also direct the job system to evaluate a privilege directive at job execution time by setting the `evaluateAtSubmission` parameter to false (as in the second and third privilege tags in the example above): The only reason one might want to do this is if the exact set of targets that the job is operating on is unknown until job execution time (for example, it is computed via an execution-time parameter source). Execution-time privilege directives are evaluated after all execution-time parameter sources are evaluated.

As a second example, suppose you are writing a job type that requires MODIFY privilege on each one of its targets, but the exact number of targets is unknown at the time of writing. The `target_names_param` and `target_types_param` attributes could be used for this purpose. These specify vector parameters that the job system will get the target names and the corresponding target types from. These could be any vector parameters; this example uses the job target list (`job_target_names` and `job_target_types`).

```
<securityInfo>
    <privilege name="MODIFY" type="target" target_names_param="job_target_names"
               target_types_param="job_target_types" />
</securityInfo>
```

5.9 Specifying Lock Information

Often, executing jobs will need to acquire *resources.* For example, a job applying a patch to a database may need a mechanism to ensure that other jobs (submitted by other users in the system) on the database are prevented from running while the patch is being applied. In other words, it may wish to acquire a *lock* on the database target so that other jobs that try to acquire the same lock block (or abort). This will allow a patch job, once it starts, to perform its work without disruption. Sometimes, locks could be at more than one level: A "hot" backup of a database, for example, can allow other hot backups to proceed (since they do not bring down the database), but cannot allow "cold" backups or database shutdown jobs to proceed (since they will end up shutting down the database, thereby causing the backup to fail). A job execution can indicate that it is reserving a resource on a target by acquiring a *lock* on the target. A lock is really a proxy for reserving some part of the functionality of a target. When an execution acquires a lock, it will block other executions that try to acquire the same lock on the target. A lock is identified by a *name,* and a *type.*A lock can be of the following types

- **Global:** These are locks that are not associated with a target. An execution that holds a global lock will block other executions that are trying to acquire the same global lock (such as a lock with the same name).
- **Target Exclusive:** These are locks that are associated with a target. An execution that holds an exclusive lock on a target will block executions that are trying to acquire any named lock on the target, as well as executions trying to acquire an exclusive lock on the target. Target exclusive locks have no name: there is exactly one exclusive lock per target.
- **Target Named**: A named lock on a target is analogous to obtaining a lock on one particular functionality of the target. A named lock has a user-specified name. An execution that holds a named lock will block other executions that are trying to acquire the same named lock, as well as executions that are trying to acquire an exclusive lock on the target.

Locks that a job type wishes to acquire can be obtained by specifying a `lockInfo` section in the job type. This section lists the locks that the job is to acquire, their types, as well as the target(s) that it wishes to acquire the locks on. Consider the section below:

```
<lockInfo action="suspend">
    <lock type="targetExclusive">
        <targetList>
            <target name="%backup_db%" type="oracle_database" />
        </targetList>
    </lock>
    <lock type="targetNamed" name="LOCK1" >
        <targetList>
            <target name="%backup_db%" type="oracle_database" />
            <target name="%job_target_names%[1]" type="%job_target_types%[1]" />
            <target name="%job_target_names%[2]" type="%job_target_types%[2]" />
        </targetList>
    </lock>
    <lock type="global" name="GLOBALLOCK1" />
</lockInfo>
```

The section above shows a job type that acquires a target-exclusive lock on a database target whose name is given by the job parameter `backup_db`. It also acquires a named target lock named "LOCK1" on three targets: the database whose name is stored in the job parameter `backup_db`, and the first two targets in the target list of the job. Finally, it

acquires a global lock named "GLOBALLOCK1". The "action" attribute specifies what the job system should do to the execution if any of the locks in the section cannot be obtained (presumably because some other execution is holding them). Possible values are `suspend` (all locks are released and the execution state changes to "Suspended:Lock") and `abort` (the execution aborts). The following points can be made about executions and locks:

- An execution can only attempt to obtain locks when it starts (although it is possible to override this by using nested jobs; see below).
- An execution can acquire multiple locks. Locks are always acquired in the order specified. Note that because of this, executions can potentially deadlock each other if they attempt to acquire locks in the wrong order.
- Target locks are always acquired on targets in the same order as they are specified in the <targetList> tag.
- If a target in the target list is null or does not exist, the execution will abort.
- If an execution attempts to acquire a lock it already holds, it will succeed.
- If an execution cannot acquire a lock (usually because some other execution is holding it), it has a choice of suspending itself or aborting. If it chooses to suspend itself, all locks it has acquired so far will be released, and the execution is put in the state Suspended/Lock.
- All locks held by an execution will be released when an execution finishes (whether it completes, aborts or is stopped). There may be several waiting executions for each released lock: they are sorted by time, and the earliest request gets the lock.

Locks and nested jobs: When jobs that have the `lockInfo` section are nested inside each other, the nested job's locks are obtained when the nested job first executes, not when an execution starts. If the locks are not available, the parent execution could be suspended/aborted, possibly after a few steps have executed.

lockInfo Examples

(1) Let us consider two job types called HOTBACKUP and COLDBACKUP. They perform hot backups and cold backups, respectively, on the database. The difference is that the cold backup brings the database down, but the hot backup leaves it up. Only one hot backup can execute at a time; it should keep out other hot backups as well as cold backups. When a cold backup is executing, no other job type can execute (since it shuts down the database as part of its execution). Let us consider a third job type called SQLANALYZE. It performs scheduled maintenance activity that results in modifications to database tuning parameters; two SQLANALYZE jobs cannot run at the same time. The chart below shows the incompatibilities between the job types; An 'X' indicates that the job types are incompatible. An 'OK' indicates that the job types are compatible.

Job Type	HOTBACKUP	COLDBACKUP	SQLANALYZE
HOTBACKUP	X	X	OK
COLDBACKUP	X	X	X
SQLANALYZE	OK	X	X

The lockInfo sections for the three job types are shown below. The cold backup obtains an exclusive target lock on the database. The hot backup job does not obtain an exclusive lock, but only the named lock "BACKUP_LOCK". Likewise, the

SQLANALYZE job obtains a named target lock called "SQLANALYZE_LOCK". Let us assume that the database that the jobs operate on is the first target in the target list of the job. The lock sections of the two jobs would look as follows:

```
<jobType name="HOTBACKUP">
    <lockInfo action="suspend">
        <lock type="targetNamed" name="BACKUP_LOCK" >
            <targetList>
                <target name="%job_target_names%[1]" type="%job_target_names%[1]"
/>
            </targetList>
        </lock>
    </lockInfo>
    ....... Rest of the job type follows
</jobType>
<jobType name="COLDBACKUP">
    <lockInfo action="suspend">
        <lock type="targetExclusive">
            <targetList>
                <target name="%job_target_names%[1]" type="%job_target_names%[1]"
/>
            </targetList>
        </lock>
    </lockInfo>
    ........ Rest of the job type follows
</jobType>

<jobType name="SQLANALYZE">
    <lockInfo action="abort">
        <lock type="targetNamed" name="SQLANALYZE_LOCK" >
            <targetList>
                <target name="%job_target_names%[1]" type="%job_target_names%[1]"
/>
            </targetList>
        </lock>
    </lockInfo>
    ........ Rest of the job type follows
</jobType>
```

Since a named target locks blocks all target exclusive locks, executing hot backups will suspend cold backups but not analyze jobs (since they try to acquire different named locks). Executing sql analyze jobs will abort other sql analyze jobs and suspend cold backups but not hot backups. Executing cold backups will suspend hot backups and abort SQL analyze jobs.

(2) Let us consider a job type called PATCHCHECK that periodically checks a patch stage area and downloads information about newly staged patches into the Enterprise Manager repository. Two such jobs cannot run at the same time; however, the job is not really associated with any target. The solution is for the job type to attempt to grab a global lock:

```
<jobType name="PATCHCHECK">
    <lockInfo>
        <lock type="global" name="PATCHCHECK_LOCK" />
    </lockInfo>
    ........ Rest of the job type follows
</jobType>
```

(3) Let us consider a job type that nests the SQLANALYZE type within itself, as shown below. Note that the nested job executes after the first step, S1 executes.

```
<jobType name="COMPOSITEJOB">
    <stepset ID="main" type="serial">
        <step ID="S1" ...>
           ....
         </step>
         <job name="nestedsql" type="SQLANALYZE">
            ....
         </job>
    </stepset>
</jobType>
```

In the example above, the nested job tries to acquire locks when it executes (since the SQLANALYZE has a lockInfo section). If the locks are currently held by other executions, then the nested job aborts (as specified in the lockInfo), which will in turn end up aborting the parent job.

5.10 Suspending a Job or Step

"Suspended" is a special state that indicates that steps in the job will not be considered for scheduling and execution. A step in an executing job can suspend the job, through the `suspend_job` PL/SQL API. This suspends both the currently executing step, as well as the job itself.

Suspending a job has the following semantics: all steps in the job that are currently in "scheduled" state will be marked as "suspended", and will thereafter not be scheduled or executed. All currently executing steps (this could happen, for example, in parallel stepsets) will continue to execute. However, when any currently executing step completes, the next step(s) in the job (if any) will not be scheduled: they will be put in suspended state. When a job is suspended on submission, the above applies to the first step(s) in the job that would have been scheduled.

Suspended jobs may be restarted at any time by calling the `restart_job()` PL/SQL API. However, jobs that are suspended because of serialization (locking) rules are not restartable manually; the job system will restart such jobs automatically when currently executing jobs of that job type complete. Restarting a job will effectively change the state of all suspended steps to scheduled: job execution will proceed normally thereafter.

5.11 Restarting a Job

If a job has been suspended, failed or aborted, it is possible to *restart* it from any given step (typically, the stepset that contains a failed or aborted step). For failed or aborted jobs, what steps actually get scheduled again when a job is restarted depends on which step the job is restarted from.

5.11.1 Restarting Versus Resubmitting

If a step in a job is *resubmitted,* it means that it executes regardless of whether the original execution of the step completed or failed. If a stepset is resubmitted, then the first step/stepset/job in the stepset is resubmitted, recursively. When a job is resubmitted, therefore, the entire job is executed again, by recursively resubmitting its initial stepset. The parameters and targets used are the same that were used when the job was first submitted. Other than that, the job executes as if it were submitted for the first time with the specified set of parameters and targets. A job can be resubmitted by using the `resubmit_job` API in the `mgmt_jobs` package. Note that jobs can be resubmitted even if the earlier executions completed successfully.

Job executions that were aborted or failed can be *restarted*. Restarting a job generally refers to resuming job execution from the last failed step (although the job type can control this behavior using the `restartMode` attribute of steps/stepsets/jobs; see below). In the common case, steps from the failed job execution that actually succeeded are not re-executed. A failed/aborted job can be restarted by calling the `restart_job` API in the `mgmt_jobs` package. A job that completed successfully cannot be restarted.

5.11.2 Default Restart Behavior

Restarting a job creates a new execution called the *restart execution.* The original, failed execution of the job is called the *source execution.* All parameters and targets are copied over from the source execution to the restart execution. Parameter sources are *not* re-evaluated, unless the original job aborted because of a parameter source failure.

To restart a serial (or iterative stepset), the job system first examines the status of the serial stepset. If the status of the serial stepset is "Completed", then all the entries for its constituent steps are copied over from the source execution to the restart execution. If the status of the stepset is "Failed" or "Aborted", then the job system starts top down from the first step in the stepset. If the step previously completed successfully in the source execution, it is copied to the restart execution. If the step previously failed or aborted, it is rescheduled for execution in the restart execution. After such a step has finished executing, the job system determines the next step(s) to execute. These could be `successOf` or `failureOf` dependencies, or simply steps/stepsets/jobs that execute after the current step. If the subsequent step completed successfully in the source execution, then it will not be scheduled for execution again; the job system merely copies the source execution status to the restart execution for that step. It continues in this fashion until it reaches the end of the stepset. It then recomputes the status of the stepset based on the new execution(s).

To restart a parallel stepset, the job system first examines the status of the parallel stepset, as before. If the status of the stepset is "Completed", then all the entries for its constituent steps are copied over from the source execution to the restart execution. If the status of the stepset is "Failed" or "Aborted", the job system copies over all successful steps in the steps from the source to the restart execution. It reschedules all steps that failed or aborted in the source execution, in parallel. After these steps have finished executing, the status of the stepset is recomputed.

To restart a nested job, the restart algorithm is applied recursively to the first (outer) stepset of the nested job.

Note that in the above paragraphs, if one of the entities being considered is a stepset or a nested job, the restart mechanism is applied recursively to the stepset or job. When entries for steps are copied over to the restart execution, the child execution entries point to the same output CLOB entries as the parent execution.

5.11.3 Using the restartMode Directive

A job type can affect the restart behavior of each step/stepset/job within it by the use of the `restartMode` attribute. This can be set to "failure" (the default) or "always". When set to failure, when the top-down copying process described in the previous section occurs, the step/stepset/job is copied without being re-executed if it succeeded in the source execution. If it failed or aborted in the source execution, it is restarted recursively at the last point of failure.

When the `restartMode` attribute is set to "always" for a step, the step is *always* re-executed in a restart, regardless of whether it succeeded or failed in the source execution. The use of this attribute is useful when certain steps in a job must always be

re-executed in a restart (for example, a step that shuts down a database prior to backing it up)

For a stepset or nested job, if the `restartMode` attribute is set to "always", then all steps in the stepset/nested job are restarted, even if they completed successfully in the source execution. If it is set to "failure", then restart is attempted only if the status of the stepset or nested job was set to Failed or Aborted in the source execution. Note that individual steps inside a stepset or nested job may have their `restartMode` set to "always"; such steps are always re-executed.

Restart Examples

Example 1

Consider the serial stepset with the sequence of steps below:

```
<jobtype ...>
<stepset ID="main" type="serial" >
    <step ID="S1" ...>
     ...
    </step>
    <step ID="S2" ...>
     ...
    </step>
    <step ID="S3" failureOf="S2"...>
     ...
    </step>
    <step ID="S4" successOf="S2"...>
     ...
    </step>
</stepset>
</jobtype>
```

In the above stepset, let us assume the source execution had S1 execute successfully and step S2 and S3 (the failure dependency of S2) fail. When the job is restarted, steps S1 is copied to the restart execution from the source execution without being re-executed (since it successfully completed in the source execution). Step S2, which failed in the source execution, is rescheduled and executed. If S2 completes successfully, then S4, its success dependency (which never executed in the source execution) is scheduled and executed. The status of the stepset (and the job) is the status of S4. On the other hand, if S2 fails, then its failure dependency, S3, is rescheduled and executed (since it had failed in the source execution), and the status of the stepset (and the job) is the status of S3.

Now let us assume that steps S1 succeeded but S2 failed, and S3 (its failure dependency) succeeded in the source execution. As a result the stepset (and therefore the job execution) succeeded. This execution cannot be restarted, since the execution completed successfully although one of its steps failed.

Finally, let us assume that steps S1 and S2 succeed, but S4 (S2's success dependency) failed. Note that S3 is not scheduled in this situation. When the execution is restarted, the job system copies over the executions of S1 and S2 from the source to the restart execution, and reschedules and executes S4. The job succeeds if S4 succeeds.

Example 2

Consider the following:

```
<jobtype ...>
<stepset ID="main" type="serial" stepsetStatus="S2" >
    <step ID="S1" restartMode="always" ...>
```

```
     ...
    </step>
    <step ID="S2" ...>
     ...
    </step>
    <step ID="S3" ...>
     ...
    </step>
</stepset>
</jobtype>
```

In the example above, assume that step S1 completes and S2 fails. S3 executes (since it does not have a dependency on S2) and succeeds. The job, however, fails, since the stepset main has its stepsetStatus set to S2. When the job is restarted, S1 is executed all over again, although it completed the first time, since the restartMode of S1 was set to "always". Step S2 is rescheduled and executed, since it failed in the source execution. After S2 executes, step S3 is *not* rescheduled for execution again, since it executed successfully in the source execution. If the intention is that S3 must execute in the restart execution, its restartMode must be set to "always".

If, in the above example, S1 and S2 succeeded and S3 failed, the stepset main would still succeed (since S2 determines the status of the stepset). In this case, the job would succeed, and cannot be restarted.

Example 3

Consider the following example:

```
<jobtype ...>
<stepset ID="main" type="serial"  >
  <stepset type="serial" ID="SS1" stepsetStatus="S1">
    <step ID="S1" ...>
     ...
    </step>
    <stepset ID="S2" ...>
     ...
    </step>
  </stepset>
  <stepset type="parallel" ID="PS1" successOf="S1" >
    <step ID="P1" ...>
     ...
    </step>
    <step ID="P2" ...>
     ...
    </step>
    <step ID="P3" ...>
     ...
    </step>
  </stepset>
</stepset>
</jobtype>
```

In the above example, let us assume that steps S1 and S2 succeeded (and therefore, stepset SS1 completed successfully). Thereafter, the parallel stepset PS1 was scheduled, and let us assume that P1 completed, but P2 and P3 failed. As a result, the stepset "main" (and the job) failed. When the execution is restarted, the steps S1 and S2 (and therefore the stepset SS1) will be copied over without execution. In the parallel stepset PS1, both the steps that failed (P2 and P3) will be rescheduled and executed.

Now assume that S1 completed and S2 failed in the source execution. Note that stepset SS1 still completed successfully since the status of the stepset is determined by S1, not S2 (because of the stepsetStatus directive). Now, assume that PS1 was scheduled and P1 failed, and P2 and P3 executed successfully. When this job is rescheduled, the step S2 will not be re-executed (since the stepset SS1 completed successfully). The step P1 will be rescheduled and executed.

(4) Consider a slightly modified version of the XML in example (3):

```
<jobtype ...>
<stepset ID="main" type="serial"  >
  <stepset type="serial" ID="SS1" stepsetStatus="S1" restartMode="always" >
    <step ID="S1" ...>
     ...
    </step>
    <stepset ID="S2" ...>
     ...
    </step>
  </stepset>
  <stepset type="parallel" ID="PS1" successOf="S1" >
    <step ID="P1" ...>
     ...
    </step>
    <step ID="P2" ...>
     ...
    </step>
    <step ID="P3" ...>
     ...
    </step>
  </stepset>
</stepset>
</jobtype>
```

In the above example, let us assume that S1 and S2 succeeded (and therefore, stepset SS1 completed successfully). Thereafter, the parallel stepset PS1 was scheduled, and let us assume that P1 completed, but P2 and P3 failed. When the job is restarted, the entire stepset SS1 is restarted (since the restartMode is set to "always"). This means that steps S1 and S2 are successively scheduled and executed. Now the stepset PS1 is restarted, and since the restartMode is not specified (it is always "failure" by default), it is restarted at the point of failure, which in this case means that the failed steps P2 and P3 are re-executed, but not P1.

5.12 Adding Job Types to the Job Activity and Job Library Pages

In order to make a new job type accessible from the Enterprise Manager console **Job Activity** and/or **Job Library** page, you need to modify specific XML tag attributes.

To display the job type on **Job Activity** page, set *useDefaultCreateUI* to "true" as shown in the following example.

```
<displayInfo useDefaultCreateUI="true"/>
```

To display the job type on the **Job Library** page, in addition to setting *useDefaultCreateUI* attribute, you must also set the *jobtype editable* attribute to "true."

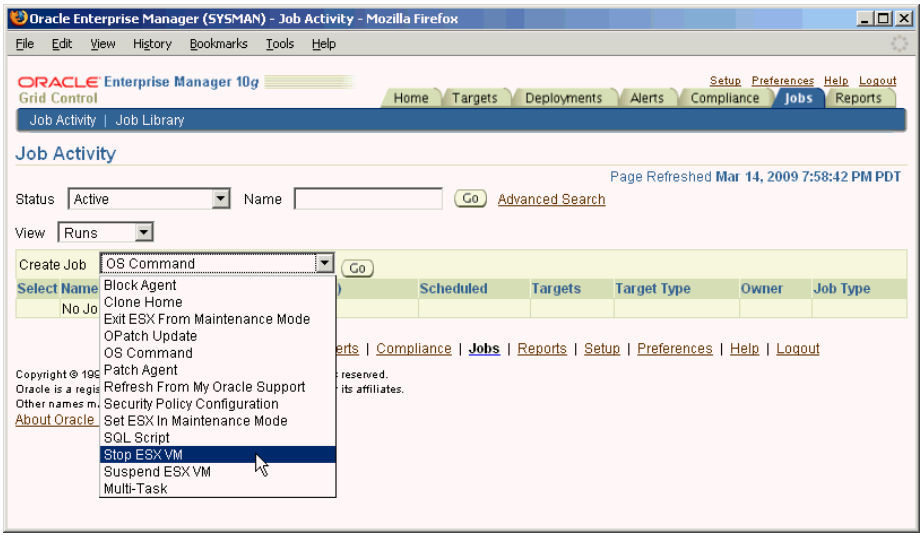
```
<jobtype name="jobType1" editable="true">
```

If only *useDefaultCreateUI*="true" and *editable*="false", then the job type will only be displayed on the Job Activity page and not on Job Library page. Also the job definition will be not editable .

5.12.1 Adding a Job Type to the Job Activity Page

As shown it Figure 5–1, setting the *useDefaultCreateUI* attribute to true allows users creating a job to select the newly added job type from the **Create Job** menu.

Figure 5–1 Available Job Types from the Job Activity Page

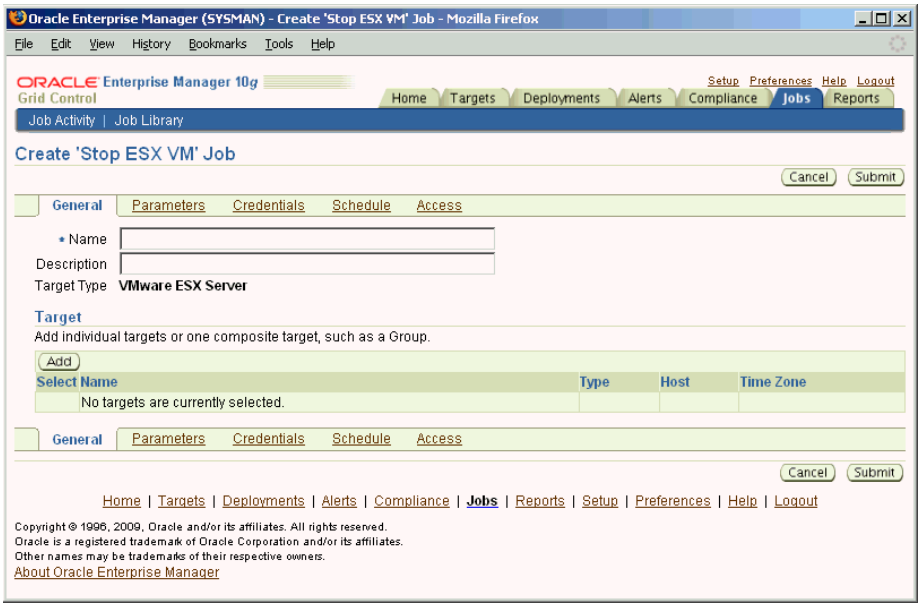


Graphic displays the drop-down menu listing available job types from the Job Activity page.

****

Making the job type available from the Job Activity page also permits access to the default **Create Job** user interface when a user attempts to create a job using the newly added job type.

Figure 5–2 Default Create Job User Interface



Graphic shows the default Create Job user interface for a newly added job type.

****

Adding the displayInfo Tag

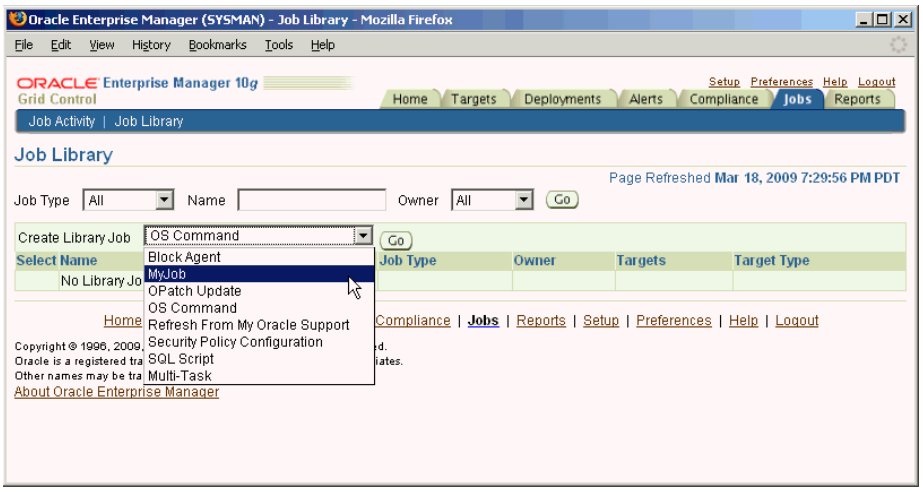
The *displayInfo* tag can be added to the job definition file at any point after the </stepset> tag and before the </jobtype> tag at the end of the job definition file, as shown in the following example.

```
<jobtype ...>
<stepset ID="main" type="serial"  >
  <stepset type="serial" ID="SS1" stepsetStatus="S1">
    <step ID="S1" ...>
     ...
    </step>
    <stepset ID="S2" ...>
     ...
    </step>
  </stepset>
  <stepset type="parallel" ID="PS1" successOf="S1" >
    <step ID="P1" ...>
     ...
    </step>
    <step ID="P2" ...>
     ...
    </step>
    <step ID="P3" ...>
     ...
    </step>
  </stepset>
</stepset>
<displayInfo useDefaultCreateUI="true"/>
</jobtype>
```

5.12.2 Adding a Job Type to the Job Library Page

To make the job type available from the **Job Library** page, you must also set the *jobType* tag's *editable* attribute to "true" in addition to adding the *displayInfo* tag, As shown it Figure 5–3, this makes the newly added job type a selectable option from the **Create Library Job** menu.

Figure 5–3 Job Type in the Job Library Page



Graphic shows the job library page and the location of the job types within the page.

****

Making the Job Type Editable

The *editable* attribute of the *jobtype* tag is set at the beginning of the job definition file, as shown in the following example.

```
<jobtype name="jobType1" editable="true">
<stepset ID="main" type="serial"  >
  <stepset type="serial" ID="SS1" stepsetStatus="S1">
    <step ID="S1" ...>
     ...
    </step>
    <stepset ID="S2" ...>
     ...
    </step>
  </stepset>
  <stepset type="parallel" ID="PS1" successOf="S1" >
    <step ID="P1" ...>
     ...
    </step>
    <step ID="P2" ...>
     ...
    </step>
    <step ID="P3" ...>
     ...
    </step>
  </stepset>
</stepset>
<displayInfo useDefaultCreateUI="true"/>
</jobtype>
```

5.13 Examples: Specifying Job Types in XML

Example 1

The following XML describes a job type called jobType1 that defines four steps, S1, S2, S3, and S4. It executes S1 and S2 serially, one after another. It executes step S3 only if

step S2 succeeds, and step S4 only if S2 fails. Note that all the steps execute within an iterative subset, so these actions are performed in parallel on all targets in the job target list of type database. Note also, the use of % signs to indicate parameters, %patchno%, %username%, %password%, and %job_target_name%. The job system will substitute the value of a job parameter named "patchno" in place of the %patchno%. Likewise, it will substitute the values of the corresponding parameters for %username% and %password%. %job_target_name% and %job_target_type% are "pre-built" placeholders that will substitute the name of the target that the step is currently executing against.

The steps S2, S3 and S4 illustrate how the remoteOp command can be used to execute a SQL*Plus script on the Agent.

The status of job is failed if any of the following occurs:

- S2 fails and S4 fails
- S2 succeeds and S3 fails

Note that since S2 executes after S1 (regardless of whether S1 succeeds or fails), the status of S1 does not affect the status of the job in any way.

```
<jobtype name="jobType1" editable="true" version="1.0">
  <stepset ID="main" type="iterativeParallel" iterate_param="job_target_types"
        iterate_param_filter="oracle_database" >
        <step ID="s1" command="remoteOp"">
          <paramList>
            <param name="remoteCommand">myprog</param>
            <param name="targetName">%job_target_names%[%job_iterate_
index%]</param>
            <param name="targetType">%job_target_types%[%job_iterate_
index%]</param>
            <param name="args">-id=%patchno%</param>
            <param name="successStatus">3</param>
            <param name="failureStatus">73</param>
            <param name="username">%username%</param>
            <param name="password">%password%</param>
          </paramList>
        </step>
        <step ID="s2" command="remoteOp"">
          <paramList>
            <param name="remoteCommand">myprog2</param>
            <param name="targetName">%job_target_names%[%job_iterate_
index%]</param>
            <param name="targetType">%job_target_types%[%job_iterate_
index%]</param>
            <param name="args">-id=%patchno%</param>
            <param name="successStatus">3</param>
            <param name="failureStatus">73</param>
            <param name="username">%username%</param>
            <param name="password">%password%</param>
          </paramList>
        </step>
         <step ID="s3" successOf="s2" command="remoteOp">
           <paramList>
             <param name="command">prog1</command>
             <param name="script">
             <![CDATA[
                   select * from MGMT_METRICS where
                       target_name=%job_target_type%[%job_iterate_param_index%]
              ]]>
              </param>
```

```
            <param name="args">%db_username%/%db_passwd%@%db_alias%</param>
            <param name="targetName">%job_target_names%[%job_iterate_
index%]</param>
            <param name="targetType">%job_target_types%[%job_iterate_
index%]</param>
            <param name="successStatus">0</param>
            <param name="failureStatus">1</param>
            <param name="username">%username%</param>
            <param name="password">%password%</param>
            </paramList>
          </step>
<step ID="s4" failureOf="s2" command="remoteOp">
           <paramList>
            <param name="input">
            <![CDATA[
                This is standard input to the executed progeam. You can use
                    placeholders for parameters, such as
                    %job_target_name%[%job_iterate_param_index%]
            ]]>
            </param>
            <param name="remoteCommand">prog2</param>
            <param name="targetName">%job_target_names%[%job_iterate_
index%]</param>
            <param name="targetType">%job_target_types%[%job_iterate_
index%]</param>
            <param name="args"></param>
            <param name="successStatus">0</param>
            <param name="failureStatus">1</param>
            <param name="username">%username%</param>
            <param name="password">%password%</param>
           </paramList>
          </step>
  </stepset>
<displayInfo useDefaultCreateUI="true"/>
</jobtype>
```

Example 2

The following XML describes a job type that has two steps, S1 and S2, that execute in parallel (within a parallel stepset ss1) and a third step, S3, that will execute only after both S1 and S2 have completed successfully. This is achieved by placing the step S3 in a serial stepset ("main") that also contains the parallel stepset ss1. This job type is a "multi-node" job. Note that use of %job_target_name%[1], %job_target_name%[2] in the parameters to the commands. In stepsets other than an iterative stepset, job targets can only be referred to using their position in the targets array (which is ordered). So, %job_target_name%[1] refers to the first target, %job_target_name%[2] to the second, and so on. The assumption is that most multi-node jobs will expect their targets to be in some order. For example, a clone job might expect the source database to be the first target, and the target database to be the second target. This job fails if any of the following occurs:

- The parallel stepset SS1 fails (either one of S1 or S2, or both fail).
- Both S1 and S2 succeed, but S3 fails.

Also note that the job type has declared itself to be Agent-bound. This means that the job will be set to Suspended/Agent Down state if either emd (corresponding to the first target or the second target) goes down.

```
<jobtype name="jobType2" version="1.0" agentBound="true" >
```

```
  <stepset ID="main" type="serial" editable="true">
      <!-- All steps in this stepset ss1 execute in parallel -->
      <stepset ID="ss1" type="parallel" >
          <step ID="s1" command="remoteCommand" >
            <paramList>
              <param name="remoteCommand">myprog</param>
              <param name="targetName">%job_target_names%[1]</param>
              <param name="targetType">%job_target_types%[1]</param>
              <param name="args">-id=%patchno%</param>
              <param name="successStatus">3</param>
              <param name="failureStatus">73</param>
              <param name="username">%username%</param>
              <param name="password">%password%</param>
            </paramList>
          </step>
          <step ID="s2" command="remoteCommand" >
            <paramList>
              <param name="remoteCommand">myprog</param>
              <param name="targetName">%job_target_names%[2]</param>
              <param name="targetType">%job_target_types%[2]</param>
              <param name="args">-id=%patchno%</param>
              <param name="successStatus">3</param>
              <param name="failureStatus">73</param>
              <param name="username">%username%</param>
              <param name="password">%password%</param>
            </paramList>
          </step>
      </stepset>
      <!-- This step executes after stepset ss1 has executed, since it is
           inside the serial subset "main"
      -->
      <step ID="s3" successOf="ss1" command="remoteCommand" >
           ...
      </step>
  </stepset>
<displayInfo useDefaultCreateUI="true"/>
</jobtype>
```

Example 3

The following example defines a new job type called jobType3 that executes jobs of type jobType1 and jobType2, one after another; the job job2 of type jobType2 is executed only if the first job fails. In order to execute another job, the target list and the param list must be passed. The targetList tag has a parameter called `allTargets`, which when sent to true, passes along the entire target list passed to this job. By setting `allTargets` to false, a job type has the option of passing along a subset of its targets to the other job type. In the example below, jobType3 passes along all its targets to the instance of the job of type jobType1 , but only the first two targets in its target list (in that order) to the job instance of type jobType2. There is another attribute called `allParams` (associated with `paramList`) that performs a similar function with respect to parameters: if `allParams` is set to true, then all parameters of the parent job are passed to the nested job. More typically, however, the nested job will have a different set of parameters (with different names). If allParams is set to false (the default), then the job type can name the nested job parameters explicitly; they need not have the same names as those in the parent job. Parameter substitution can be used to express the nested job parameters in terms of the parent job parameters, as shown in this example.

Note that dependencies can be expressed between nested jobs just as if they were steps or stepsets. In this example, a job of type jobType3 succeeds if either the nested job job1 succeeds or if job1 fails and job2 succeeds.

```
<jobType name="jobType3" editable="true" version="1.0">
  <stepset ID="main" type="serial">
     <job type="jobType1" ID="job1" >
        <target_list allTargets="true" />
        <paramList>
          <param name="patchno">%patchno%</param>
          <param name="username">%username%</param>
          <param name="password">%password%</param>
        </paramList>
      </job>
      <job type="jobType2" ID="job2" failureOf="job1" >
        <targetList>
          <target name="%job_target_names%[1]" type="%job_target_types%[1]" />
          <target name="%job_target_names%[2]" type="%job_target_types%[2]" />
        </targetList>
        <paramList>
          <param name="patchno">%patchno%</param>
          <param name="username">%username%</param>
          <param name="password">%password%</param>
        </paramList>
      </job>
  </stepset>
<displayInfo useDefaultCreateUI="true"/>
</jobType>
```

Example 4

This example illustrates the use of the `generateFile` command. Let us assume that you are executing a sequence of scripts, all of which need to source a common file that sets up some environment variables, which are known only at runtime. One way to do this is to generate the variables in a file with a unique name. All subsequent scripts are passed this file name as one of their command-line arguments, which they read to set the needed environment/shell variables.

The first step, S1, in this job uses the `generateFile` command to generate a file named <app-home>/<execution-id>.env. Since the execution id of a job is always unique, this ensures a unique file name. It generates three environment variables, ENVVAR1, ENVVAR2, ENVVAR3, which are set to the values of the job parameters param1, param2 and param2, respectively. These parameters must be set to the right values when the job is submitted. Note that %job_execution_id% is a placeholder provided by the job system, while %app-home% is a job parameter which must be explicitly provided when the job is submitted.

The second step, S2, executes a script called myscript. The first command-line argument to the script is the generated filename. This script must "source" the generated file, which will set the required environment variables, and then go about it's other tasks, in the manner shown below:

```
#!/bin/ksh
ENVFILE=$1
# Execute the generated file, sets the required environment vars
. $ENVFILE
# I can now reference the variables set in the file
doSomething $ENVVAR1 $ENVVAR2 $ENVVAR3...
```

The full job type specification is given below. Note the step, S3, that removes the file that was created by the first step S1. It is important to clean up after yourself when using the putFile and generateFile commands to write temporary files on the Agent. The cleanup is done here explicitly as a separate step, but it could also be done by one of the scripts that execute on the remote host.

Additionally, note the use of the `securityInfo` section that specifies that the user that submits a job of this job type must have maintain privilege on both the targets that the job operates on.

```
<jobtype name="jobType4" editable="true" version="1.0">
  <securityInfo>
    <privilege name="MAINTAIN" type="target" evaluateAtSubmission="false">
        <target name="%job_target_names%[1]" type="%job_target_types%[1]" />
        <target name="%job_target_names%[2]" type="%job_target_types%[2]" />
    </privilege>
  </securityInfo>

<stepset ID="main" type="serial">
          <step ID="s1" command="putFile" >
            <paramList>
              <param name=sourceType>inline</param>
              <param name="destFile">%app-home%/%job_execution_id%.env</param>
              <param name="targetName">%job_target_names%[1]</param>
              <param name="targetType">%job_target_types%[1]</param>
              <param name="username">%username%</param>
              <param name="password">%password%</param>
              <param name=contents">
              <![CDATA[#!/bin/ksh
                 export ENVVAR1=%param1%
                 export ENVVAR2=%param2%
                 export ENVVAR3=%param3%
              ]]>
              </param>
            </paramList>
          </step>
          <step ID="s2" command="remoteCommand" >
            <paramList>
              <param name="remoteCommand">myscript</param>
              <param name="targetName">%job_target_names%[2]</param>
              <param name="targetType">%job_target_types%[2]</param>
              <param name="args">%app-home%/%job_execution_id%.env</param>
              <param name="successStatus">3</param>
              <param name="failureStatus">73</param>
              <param name="username">%username%</param>
              <param name="password">%password%</param>
            </paramList>
          </step>
          <step ID="s3" command="remoteCommand" >
            <paramList>
              <param name="remoteCommand">rm</param>
              <param name="targetName">%job_target_names%[2]</param>
              <param name="targetType">%job_target_types%[2]</param>
              <param name="args">-f, %app-home%/%job_execution_id%.env</param>
              <param name="successStatus">0</param>
              <param name="username">%username%</param>
              <param name="password">%password%</param>
            </paramList>
          </step>
  </stepset>
```

```
<displayInfo useDefaultCreateUI="true"/>
</jobtype>
```

Example 5

This example illustrates the use of the `repSQL` command to execute SQL statements and anonymous PL/SQL blocks against the repository. The job type specification below calls a simple SQL statement in the first step S1, and a PL/SQL procedure in the second step. Note the use of the variables %job_id% and %job_name%, which are special job-system placeholders. Other job parameters can be similarly escaped as well. Also note the use of bind parameters in the SQL queries. The parameters `sqlinparam[n]` can be used to specify bind parameters. There must be one parameter of the form `sqlinparam[n]` for each bind parameter. Bind parameters must be used as far as possible to make optimum use of database resources.

```
<jobtype name="repSQLJob" editable="true" version="1.0">
 <stepset ID="main" type="serial">
   <step ID="s1" command="repSQL" >
    <paramList>
       <param name="sql">update mytable set status='executed' where name=?</param>
       <param name="sqlinparam1">%job_name%</param>
    </paramList>
   </step>
 <step ID="s2" command="repSQL" >
   <paramList>
      <param name="sql">begin mypackage.job_done(?,?,?); end;</param>
      <param name="sqlinparam1">%job_id%</param>
      <param name="sqlinparam2">3</param><param name="sqlinparam3">mgmt_
rep</param>
   </paramList>
</step>
</stepset>
<displayInfo useDefaultCreateUI="true"/>
</stepset>
</jobtype>
```

Example 6

This example illustrates the use of the switch stepset. The main stepset of this job is a switch stepset whose `switchVarName` is a job parameter called `stepType`. The possible values (`switchCaseVal`) that this parameter can have are "simpleStep", "parallel", and "OSJob", which will end up selecting, respectively, the step SWITCHSIMPLESTEP, the parallel stepset SWITCHPARALLELSTEP, or the nested job J1.

```
<jobType version="1.0" name="SwitchSetJob" editable="true">
  <stepset ID="main" type="switch" switchVarName="stepType" >
    <step ID="SWITCHSIMPLESTEP" switchCaseVal="simpleStep" command="remoteOp">
      <paramList>
        <param name="remoteCommand">%command%</param>
        <param name="args">%args%</param>
        <param name="targetName">%job_target_names%[1]</param>
        <param name="targetType">%job_target_types%[1]</param>
        <param name="username">%username%</param>
        <param name="password">%password%</param>
      </paramList>
    </step>
    <stepset ID="SWITCHPARALLELSTEP" type="parallel" switchCaseVal="parallelStep">
      <step ID="P11" command="remoteOp" >
        <paramList>
```

```
          <param name="remoteCommand">%command%</param>
          <param name="args">%args%</param>
          <param name="targetName">%job_target_names%[1]</param>
          <param name="targetType">%job_target_types%[1]</param>
          <param name="username">%username%</param>
          <param name="password">%password%</param>
        </paramList>
      </step>
      <step ID="P12" command="remoteOp" >
        <paramList>
          <param name="remoteCommand">%command%</param>
          <param name="args">%args%</param>
          <param name="targetName">%job_target_names%[1]</param>
          <param name="targetType">%job_target_types%[1]</param>
          <param name="username">%username%</param>
          <param name="password">%password%</param>
        </paramList>
      </step>
    </stepset>
    <job ID="J1" type="OSCommandSerial" switchCaseVal="OSJob" >
        <paramList>
            <param name="command">%command%</param>
            <param name="args">%args%</param>
            <param name="username">%username%</param>
            <param name="password">%password%</param>
        </paramList>
        <targetList>
            <target name="%job_target_names%[1]" type="%job_target_types%[1]" />
        </targetList>
    </job>
  </stepset>
<displayInfo useDefaultCreateUI="true"/>
</jobType>
```

Example 7

This example shows the use of the <securityInfo> tag to ensure that only users that have "CLONE FROM" privilege over the first target and maintain privilege over the second target will be able to submit jobs of the following type...

```
<jobType name="Clone" editable="true" version="1.0" >
  <securityInfo>
    <privilege name="CREATE TARGET" type="system" />
    <privilege name="CLONE FROM" type="target" evaluateAtSubmission="false" >
        <target name="%job_target_names%[1]" type="%job_target_types%[1]" />
    </privilege>
    <privilege name="MAINTAIN" type="target" evaluateAtSubmission="false">
        <target name="%job_target_names%[2]" type="%job_target_types%[2]" />
    </privilege>
  </securityInfo>
  <!-- An optional <paramInfo> section will follow here, followed by the stepset
       definition of the job
  -->
  <paramInfo>
   ....
  </paramInfo>
  <stepset ...>
   .......
  </stepset>
<displayInfo useDefaultCreateUI="true"/>
```

```
</jobType>
```

5.14 Performance Issues

This section briefly touches on issues to keep in mind when designing your job type that might impact performance of your job type, as well as the overall job system.

5.14.1 Using Parameter Sources

- Parameter sources are a convenient way to obtain needed parameters from known sources (such as the repository, or the credentials table). The parameter sources must be used only for quick queries that fetch information already stored somewhere.
- In general, parameter sources that are evaluated at job execution time (`evaluateAtSubmission=true`) will effect the throughput of the job dispatcher and must be used with care. In some cases, fetching of parameters at execution time may be unavoidable; if you don't care whether the parameters are fetched at execution time or submission time, set `evaluateAtSubmission=false`.
- When executing SQL queries to obtain parameters (using the SQL parameter source) the usual performance improvement guidelines apply, such as using indexes only where necessary and avoiding joining large tables.

5.15 Adding a Job Type to Enterprise Manager

To package a new job type with a Management plug-in there are some implementation guidelines must be followed.

New job types packaged with a Management Plug-in will have two new files:

- **a job type definition XML file** - used by the job system during Management Plug-in deployment to define your new job type. There is one XML file for each job type.
- **a job type script file** - installed on selected Agents during Management Plug-in deployment. A single script may be shared amongst different jobs.

The following two properties must be set to "true" in the first line of the job type definition XML file:

- agentBound
- singleTarget

Here is an example:

```
<jobType version="1.0" name="PotatoUpDown" singleTarget="true" agentBound="true"
targetTypes="potatoserver_os">
```

Because use of Java for a new job type is not supported for job types packaged with a Management Plug-in, new job types are *agentBound* and perform their work through a script delivered to the Agent (the job type script file). The job type definition XML file contains a reference to the job type script file and will execute it on the Agent whenever the job is run from the Enterprise Manager console.

The job type definition XML file also contains references to a credential set which allow it to run the script on the Agent. A new Credential Set (and Credential Type) must be created in your target type definition file to create reference to these credentials in the console. Once created, the new Credential Set becomes a Preferred

Credential which you can set in the Enterprise Manager console. Since the credentials are used for running a script on the Agent, they will be credentials for the Agent host. Although Agent host credentials exist in the console, for the job type, the credentials you use must be attached to your target type and created in your target type definition as shown in the following example.

```
<CredentialInfo>
  <!-- The credential type for target type host -->
  <CredentialType NAME="PotatoHostCreds" >
    <Display>
      <Label NLSID="POTATO_HOSTCREDS">Host Credentials</Label>
    </Display>
    <CredentialTypeColumn NAME="HostUsername" IS_KEY="TRUE">
      <Display>
        <Label NLSID="POTATO_HOST_USERNAME">UserName</Label>
      </Display>
    </CredentialTypeColumn>
    <CredentialTypeColumn NAME="HostPassword">
      <Display>
        <Label NLSID="POTATO_HOST_Password">Password</Label>
      </Display>
    </CredentialTypeColumn>
  </CredentialType>

  <CredentialSet NAME="PotatoHostCreds" CREDENTIAL_TYPE="PotatoHostCreds"
             USAGE="PREFERRED_CRED">
    <Display>
      <Label NLSID="CREDS_POTATO_HOST">Potato Host Credentials</Label>
    </Display>

    <CredentialSetColumn TYPE_COLUMN="HostUsername"
       SET_COLUMN="username">
      <Display>
        <Label NLSID="CREDS_POTATO_HOST_UNSERNAME">Agent Host Username</Label>
      </Display>
    </CredentialSetColumn>
    <CredentialSetColumn TYPE_COLUMN="HostPassword"
       SET_COLUMN="password">
      <Display>
        <Label NLSID="CREDS_POTATO_HOST_PASSWORD">Agent Host Password</Label>
      </Display>
    </CredentialSetColumn>
  </CredentialSet>
</CredentialInfo>
```

The `CredentialSetColumn TYPE_COLUMN` must match the `CredentialTypeColumn NAME` for the columns to match up.

Adding a the Job Type to a Management-Plug-in Archive

Once you have created the job type definition XML file and modified the target type definition file, you can add your files to a Management Plug-in Archive (MPA) just as you would any other target type. See Chapter 2, "Developing a Management Plug-in" for more information.

A Management Plug-in is created by adding the files previously discussed to an MPA using the Enterprise Manager Command Line Interface (EM CLI). Each call to the EM CLI adds another unique Management Plug-in to the MPA. For each Management Plug-in, the EM CLI allows you to specify a base version of the Management Agent

that the plug-in is expected to work against and a base version that the Oracle Management Service must be for the plug-in to be imported into the Management Repository. To create a MPA, perform the following

1. Open a terminal window on a machine where the EM CLI client is installed.
2. At the command prompt issue the add_mp_to_mpa verb. The following example shows the verb parameters that must be supplied. To add a job type, you use the JOB_SCRIPT and JOB_DEFINITION filetypes. For more information about the `add_mp_to_mpa verb`, see the EM CLI command line help or the *Oracle Enterprise Manager Command Line Interface* guide.

Example 5–2 Using the EM CLI to Create a Management Plug-in Archive

```
emcli add_mp_to_mpa
  -mpa=$HS_HOME/lib/host_sample.jar -mp_version=1.1
  -ttd=$HS_HOME/metadata/host_sample_ttd.xml
  -dc=$HS_HOME/metadata/host_sample_dc.xml
  -file="REPORT_DEFINITION:$HS_HOME/sql/host_sample_perf_report.sql"
  -file="REPORT_DEFINITION:$HS_HOME/sql/host_sample_config_report.sql"
  -file="MONITORING_SCRIPT:$HS_HOME/scripts/data_collector.pl"
  -file="HOMEPAGE_DEFINITION:$HS_HOME/metadata/host_sample_homepage_charts.xml"
  -file="JOB_DEFINITION:$HS_HOME/jobs/host_sample_job_killprocess.xml"
  -file="JOB_SCRIPT:$HS_HOME/scripts/kill_process.pl"
  -func_desc="Demo Plug-in: Linux host monitoring."
  -req_desc="Requirements:  Requires that the Agent that hosts the target
     instances be running on Linux. If the 'Use Fake Data' property is set
     when adding a target instance, then all the data provided will be
     generated and a Linux Agent is not required";
```

Briefly, the verb options are:

- **mpa**

  The name of the Management Plug-in Archive where the Management Plug-in is to be added.

- **mp_version**

  The version of the Management Plug-in to be created. The Management Plug-in version should be incremented whenever any of the files in the Management Plug-in are changed.

- **tdd**

  The explicit path of the target type metadata file.

- **dc**

  The explicit path of the default collection file.

- **oms_version**

  The minimum OMS version that is compatible with this Management Plug-in.

- **file**

  The type and path of the other Management Plug-in files to be added. The following types are supported:

  – MONITORING_BINARY

  – POLICY_DEPLOY

  – POLICY_UNDEPLOY

  - MONITORING_SCRIPT
  - REPORT_DEFINITION
  - JOB_SCRIPT
  - JOB_DEFINITON

- **func_desc**

  The functional description for the Management Plug-in. This description appears in the Enterprise Manager console once the plug-in has been imported.

- **req_desc**

  The Requirements description of the Management Plug-in. This description appears in the Enterprise Manager console and specifies any plug-in deployment requirements.

Once you have added the Management Plug-in containing your new job type to Enterprise Manager, you can then monitor instances of that target type and have the job system perform the newly defined job types against those target instances.

Important: The Management Plug-in's jobs will not appear until an instance of the plug-in's target type is added to the Enterprise Manager environment.

6

Adding Reports

Defining new target types in Enterprise Manager via Management Plug-ins also provides you with the opportunity to add new report definitions. Plug-ins allow you to add permanent (SYSTEM) target type-specific report definitions to Enterprise Manager using the Information Publisher PL/SQL API.

This chapter covers the following:

> **Note:** You must have a working knowledge of SQL and PL/SQL before using this API. Refer to the *Oracle Database PL/SQL User's Guide and Reference* for more information.

6.1 What You Get

Adding report definitions via Management Plug-in creates target type-specific SYSTEM reports. SYSTEM report definitions are handled differently than definitions created through the Information Publisher user interface. SYSTEM reports are permanent and cannot be deleted or edited by Enterprise Manager administrators. You can add multiple report definitions to a Management Plug-in, thus allowing you to associate multiple reports with a specific target type.

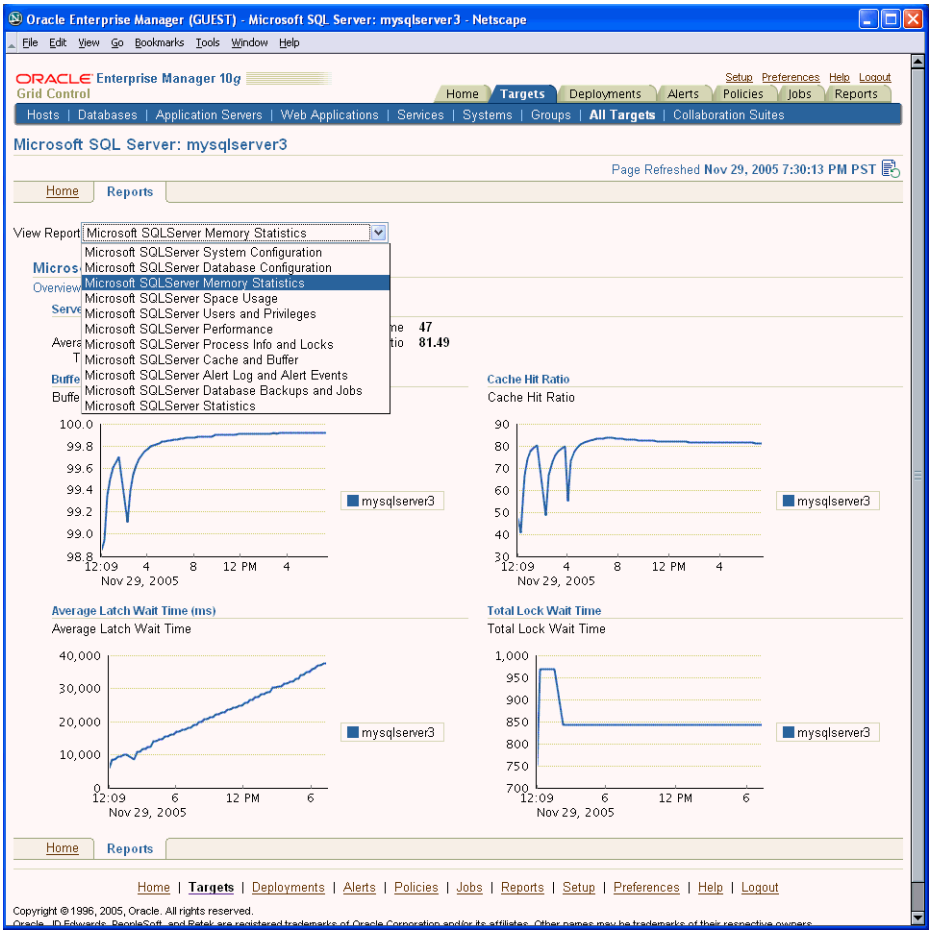
Adding SYSTEM report definitions using Management Plug-ins and the Information Publisher API allows users to access reports from two areas of the Enterprise Manager console:

- Reports page of the target home page (optional).
- Information Publisher's Report Definition page.

6.1.1 Reports Page (Target Home Page)

In the report definition file, you can specify whether or not a report is available from the Reports page of the target home page. Report definitions you add to the Reports page are available from the View Report drop-down menu list. The following figure shows the Reports page for a Microsoft SQL Server target.

Figure 6–1 Microsoft SQL Server Reports Page



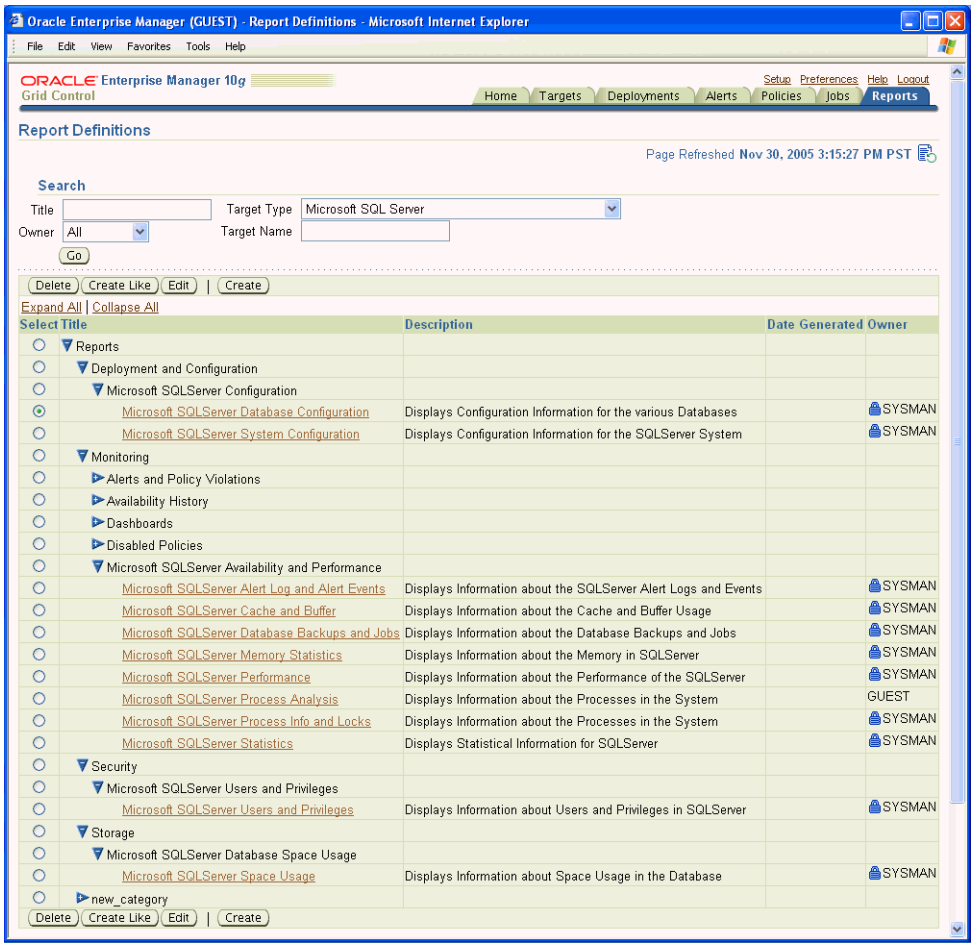
The figure shows the Reports page for a Microsoft SQL Server target.

****

6.1.2 Report Definitions Page

Regardless of whether you specify that a report appear in the Reports page, all report definitions added via Management Plug-ins are available from Information Publisher's Report Definitions page. As with out-of-box SYSTEM report definitions, those added via Management Plug-ins are organized according to report category and subcategory. SYSTEM report definitions cannot be deleted from the Enterprise Manager console. Figure 6–2, "Report Definitions Page" shows available report definitions for Microsoft SQL Server target types.

Figure 6–2 Report Definitions Page



This figure shows the Information Publisher Report Definitions page with Microsoft SQL Server-specific report definitions expanded.

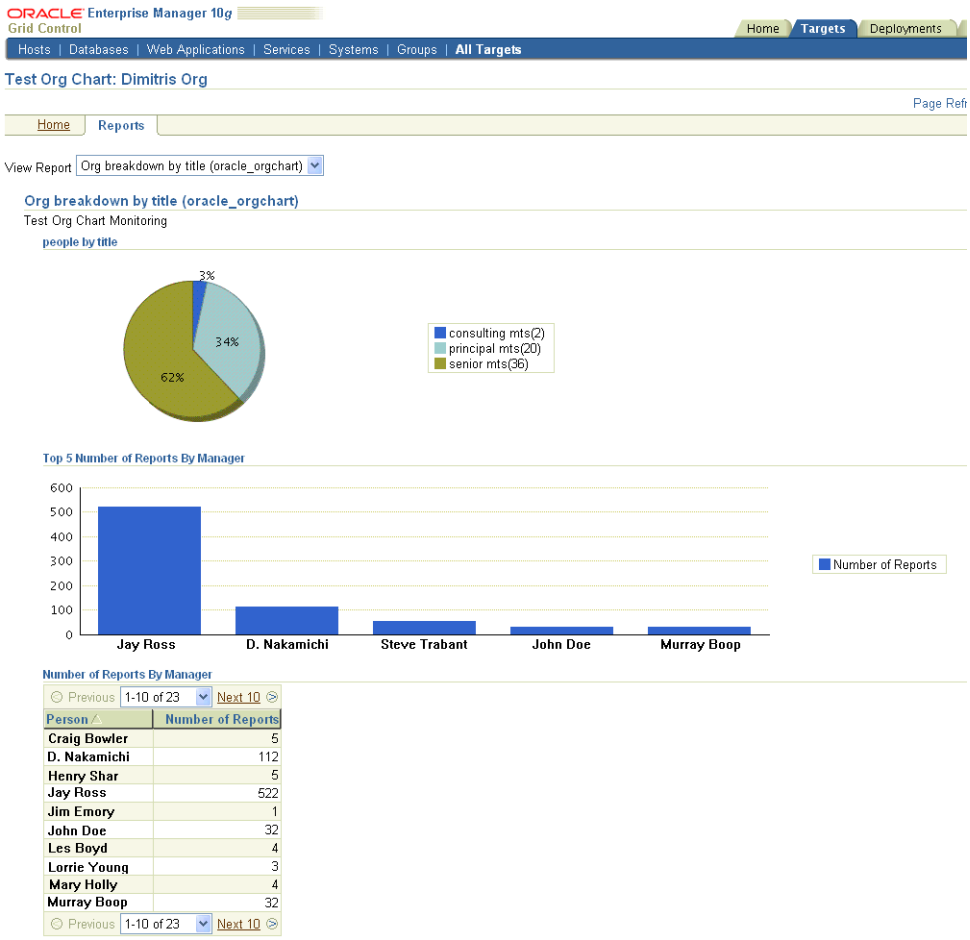
****

6.2 Report Definition File

A report definition file is structured as a conventional PL/SQL block and contains code to extract pertinent information from the Management Repository and the report elements used to format and display that data. The Information Publisher PL/SQL API allows you specify the report elements and parameters that you normally specify interactively when creating a report definition from the Enterprise Manager console.

The following example shows the Reports page for an "oracle_orgchart" target type that was added as a Management Plug-in. A single report titled "Org breakdown by title (oracle_orgchart)" is shown in Figure 6–3. This report consists of three major areas: a pie chart showing the percentage distribution of people by title, a bar chart showing the top five managers having the most people reporting to them, and a table listing all managers in the organization the total number of direct reports.

Figure 6–3 Oracle Organization Report



The figure shows the Reports page for an oracle_orgchart target type.

****

The report definition used for this example utilizes two report element types: *Chart from SQL* (pie and bar), and *Table from SQL*. In general, these two report elements will meet most reporting needs.

6.3 Creating a Report Definition File

Management Plug-ins allow you to define as many report definition files as is required for a particular target type. The content of a report definition file consists of a PL/SQL block that uses the Information Publisher PL/SQL API (part of the MGMT_IP package) to construct a report.

6.3.1 Report Definition File Development Process

The process of developing a valid report definition file involves three steps:

1. Define SQL or PL/SQL queries used to extract information from the Management Repository.
2. Create a test report interactively from the Enterprise Manager console.
3. Use the PL/SQL API to create a report definition file.

Define SQL or PL/SQL queries used to extract information from the Management Repository.

The first step in creating your report definition is to create the SQL or PL/SQL queries used to extract the requisite report information from the Management Repository. Enterprise Manager provides management views with which you can safely extract data from the Management Repository without reading from the base tables. Using repository views protects your queries from changes to the repository schema that may occur in future releases and ensures your SYSTEM report definitions remain functional. A complete listing of repository views can be found in Chapter 9, "Management Repository Views" on page 9-1.

The following query was used to extract repository information about the distribution of employee classifications within an Oracle organization. The query uses the MGMT$METRIC_CURRENT repository view.

```
select 'senior mts', count(value) from mgmt$metric_current
where metric_column = 'Title' and LOWER(value) like '%senior member%' and
      target_guid = ??EMIP_BIND_TARGET_GUID??
union
select 'principal mts', count(value) from mgmt$metric_current
where metric_column = 'Title' and LOWER(value) like '%principal member%'and
      target_guid = ??EMIP_BIND_TARGET_GUID??
           union
           select 'consulting mts', count(value) from mgmt$metric_current
           where metric_column = 'Title' and LOWER(value) like '%consulting%' and
                 target_guid = ??EMIP_BIND_TARGET_GUID?? ;
```

When an administrator views a report from the Enterprise Manager console that contains this SQL query string, Information Publisher automatically binds the unique identifier for the selected target to the ??EMIP_BIND_TARGET_GUID?? placeholder in the SQL query string. The documentation for *Chart from SQL* and *Table from SQL* parameters provide information on this bind variable placeholder as well as others you can include in your SQL query string.

The data returned from the SQL query is then used by the *Chart from SQL* report element to render the pie chart shown in the Reports page of an oracle_orgchart target home page as shown in Figure 6–3.

Create a test report interactively from the Enterprise Manager console.

Once you have written and tested the SQL or PL/SQL query, you can use the Enterprise Manager console to generate a version of your report interactively using the *Chart from SQL* and *Table from SQL* report elements. By using the Information Publisher user-interface, you can easily prototype reports without having to create a report definition file and import Management Plug-in Archives. You can also use this method of interactive prototyping to refine your queries and ensure that the data extracted from the Management Repository and how that information is rendered in your report meets your reporting requirements. See Enterprise Manager online help for the *Chart from SQL* and *Table from SQL* 'Set Parameters' function for information and examples on how to format your queries.

Use the PL/SQL API to create a report definition file.

Once you are satisfied with the way your report is being rendered by Information Publisher, you are ready to create the report definition file. As previously mentioned, the report definition file consists of a PL/SQL block defining the report elements and the SQL or PL/SQL queries used by the elements to extract repository information and render the report. Example 6–1 shows the content of the report definition file associated with the oracle_orgchart target type.

Example 6–1 Oracle Organization Report Definition File

```
BEGIN

 DECLARE
   l_target_types  MGMT_IP_TARGET_TYPES;
   l_param_values  MGMT_IP_PARAM_VALUE_LIST;
   l_target_type   MGMT_IP_TARGET_TYPES;
   l_report_guid   RAW(16);
   l_element_guid  RAW(16);
   l_report_title_nlsid  VARCHAR2(128);
   l_report_owner VARCHAR(100);
   l_curr_order    NUMBER;
   l_curr_row      NUMBER;

   BEGIN

-- specify the target type associated with this report
-- in this case the target type is 'oracle_orgchart'
   l_target_type := MGMT_IP_TARGET_TYPES();
   l_target_type.extend(1);
   l_target_type(1) := 'oracle_orgchart';
   l_report_title_nlsid := 'Org breakdown by title (oracle_orgchart)<>';
   l_report_owner := mgmt_user.get_repository_owner;

-- create a report definition for the report
   l_report_guid := mgmt_ip.create_report_definition (
     p_title_nlsid               => l_report_title_nlsid,
     p_description_nlsid         => 'Charts showing the breakdown by title',
     p_owner                     => l_report_owner,
     p_category_nlsid            => 'Test Org Chart Reports',
     p_sub_category_nlsid        => 'Interesting Org Data',
     p_late_binding_target_types  => l_target_type,
     p_show_table_of_contents    => 0,
     p_system_report             => 1,
     p_component_name            => l_target_type(1)
     );

   -- Add report so that it shows up under "reports" tab in default home page
   mgmt_mp_homepage.add_report (
     p_target_type        => l_target_type(1),
     p_report_title       => l_report_title_nlsid,
     p_report_owner       => l_report_owner,
     p_report_order       => 1
     );

-- create the first element in the report
-- it will be a text element with a description of the report contents

-- set the parameters for the styled text element
-- we?ll provide a message and a display style
   l_param_values := MGMT_IP_PARAM_VALUE_LIST();
   l_param_values.extend(2);
   l_param_values(1) := MGMT_IP_PARAM_VALUE_RECORD(
     'oracle.sysman.eml.ip.render.elem.TextParamBean.textMessage',
     'Test Org Chart Monitoring');
   l_param_values(2) := MGMT_IP_PARAM_VALUE_RECORD(
     'oracle.sysman.eml.ip.render.elem.TextParamBean.textStyle',
     'OraInstructionText');

-- initialize the sequencing variables
```

```
-- l_curr_order should sequentially increase from 1 to the number of elements in
-- the report
-- l_curr_row indicates the row on which to display an element
    l_curr_order := 1;
    l_curr_row := 1;

   l_element_guid := mgmt_ip.add_element_to_report_def (
     p_report_guid        => l_report_guid,
     p_element_name_nlsid => 'IPMSG_STYLED_TEXT',
     p_element_type_nlsid => 'IPMSG_ANY_TARGET_TYPE',

     p_header_nlslid      => null,
     p_element_order      => l_curr_order,
     p_element_row        => l_curr_row,
     p_parameters         => l_param_values,
     p_targets            => null

     );

-- the second element in the report is a pie chart showing
-- the count of employees by title

   l_param_values := mgmt_ip_param_value_list();
   l_param_values.extend(3);
   l_param_values(1) := mgmt_ip_param_value_record(
     'oracle.sysman.eml.ip.render.elem.sqlStatement',
     'select ''senior mts'', count(value)
        from mgmt$metric_current where metric_column = ''Title'' and LOWER(value)
          like ''%senior member%'' and target_guid = ??EMIP_BIND_TARGET_GUID??
 union
        select ''principal mts'', count(value)
          from mgmt$metric_current
        where metric_column = ''Title'' and LOWER(value)
          like ''%principal member%'' and target_guid = ??EMIP_BIND_TARGET_GUID??
          union
          select ''consulting mts'', count(value)
            from mgmt$metric_current where metric_column = ''Title'' and
            LOWER(value) like ''%consulting%'' and
            target_guid = ??EMIP_BIND_TARGET_GUID?? '
     );
   l_param_values(2) := mgmt_ip_param_value_record(
     'oracle.sysman.eml.ip.render.elem.ChartParamController.chartType',
     'pieChart');
   l_param_values(3) := mgmt_ip_param_value_record(
'oracle.sysman.eml.ip.render.elem.ChartParamController.pieShowSlicePercentLabels',
 'true');

   l_curr_order := l_curr_order + 1;
   l_curr_row := l_curr_row + 1;

-- add pie chart to report definiton
--
   l_element_guid := mgmt_ip.add_element_to_report_def (
     p_report_guid        => l_report_guid,
     p_element_name_nlsid => 'IPMSG_USER_CHART_FROM_SQL',
     p_element_type_nlsid => 'IPMSG_ANY_TARGET_TYPE',
     p_header_nlslid      => 'people by title',
     p_element_order      => l_curr_order,
     p_element_row        => l_curr_row,
     p_parameters         => l_param_values ,
```

```
      p_targets            => null

      );

--add a bar chart showing number of reports by manager
--

   l_param_values := MGMT_IP_PARAM_VALUE_LIST();
   l_param_values.extend(3);
   l_param_values(1) := MGMT_IP_PARAM_VALUE_RECORD(
     'oracle.sysman.eml.ip.render.elem.sqlStatement',
     'select * from (select key_value, to_number(value) "Number of Reports" from
      mgmt$metric_current where target_type = ''oracle_orgchart'' and
      metric_column = ''Reports'' and value is not null  and
      target_guid = ??EMIP_BIND_TARGET_GUID?? order by to_number(value) DESC )
      where rownum < 6'
     );
   l_param_values(2) := MGMT_IP_PARAM_VALUE_RECORD(
     'oracle.sysman.eml.ip.render.elem.ChartParamController.chartType',
     'barChart');
   l_param_values(3) := MGMT_IP_PARAM_VALUE_RECORD(
     'oracle.sysman.eml.ip.render.elem.ChartParamController.width',
     '800');

    l_curr_order := l_curr_order + 1;
    l_curr_row := l_curr_row + 1;

   l_element_guid := mgmt_ip.add_element_to_report_def (
     p_report_guid        => l_report_guid,
     p_element_name_nlsid => 'IPMSG_USER_CHART_FROM_SQL',
     p_element_type_nlsid => 'IPMSG_ANY_TARGET_TYPE',
     p_header_nlslid      => 'Top 5 Number of Reports By Manager',
     p_element_order      => l_curr_order,
     p_element_row        => l_curr_row,
     p_parameters         => l_param_values,
     p_targets            => null

     );

-- the next element is a table the manager name and number of reports per manager

   l_param_values := MGMT_IP_PARAM_VALUE_LIST();
   l_param_values.extend(1);
   l_param_values(1) := MGMT_IP_PARAM_VALUE_RECORD(
     'oracle.sysman.eml.ip.render.elem.sqlStatement',
     'select key_value "Person", to_number(value) "Number of Reports" from
      mgmt$metric_current where target_type = ''oracle_orgchart'' and
      metric_column = ''Reports'' and value is not null  and
      target_guid = ??EMIP_BIND_TARGET_GUID?? order by to_number(value) DESC '
     );

   l_curr_order := l_curr_order + 1;
   l_curr_row := l_curr_row + 1;

   l_element_guid := mgmt_ip.add_element_to_report_def (
     p_report_guid        => l_report_guid,
     p_element_name_nlsid => 'IPMSG_USER_TABLE_FROM_SQL',
     p_element_type_nlsid => 'IPMSG_ANY_TARGET_TYPE',
     p_header_nlslid      => 'Number of Reports By Manager',
     p_element_order      => l_curr_order,
```

```
      p_element_row        => l_curr_row,
      p_parameters         => l_param_values ,
      p_targets            => null
      );

   END;

END;
```

6.3.2 Report Lifecycle: Updating Report Definitions

With the ability to add report definitions to Enterprise Manager comes the responsibility of maintaining and updating the report definitions. Familiarity with the way in which Enterprise Manager handles report definitions will allow you to anticipate system behavior and plan for backwards compatibility.

When report definitions are deployed via Management Plug-in, Enterprise Manager only allows newer versions of the report definitions to be installed. Older report definitions are deleted and deregistered so as not to appear on the Reports subtab of a target home page. These actions eliminate potential version conflicts by ensuring that updated report definitions are deployed to clean systems. Enterprise Manager will not install older versions of a report definition.

Report definitions, as with Management Plug-ins in general, should be designed with backwards compatibility in mind. Future versions of report definitions should support previous versions of the target type metadata. Report definition-metadata version incompatibility will be most apparent in the following situations:

- Report definitions included with Management Plug-in version 1 AND not included with Management Plug-in version 2 will disappear when version 2 is deployed, even if version 1 deployments remain on some Management Agents.
- If version 1 and version 2 of a Management Plug-in are both deployed to the system, Management Agents will collect data based on the metadata of the version installed at that Agent; some will collect for version 1 metadata and some for version 2 metadata. Only the version 2 report definitions will be installed (appear in the Enterprise Manager console). For this reason, version 2 report definitions must support both versions of the metadata.

6.4 PL/SQL Application Programmer Interface

The Information Publisher PL/SQL API allows you to create a report definition file.

6.4.1 PL/SQL Methods for Creating Report Definitions

Use the following PL/SQL methods to create and/or manipulate report definitions when creating report definition files.

- mgmt_ip.create_report_definition
- mgmt_ip.create_report_definition
- mgmt_mp_homepage.add_report

6.4.1.1 mgmt_ip.create_report_definition

Call this method to create a new report definition. Once a report definition is created, elements can be added. The create_report_definition method is part of the MGMT_IP PL/SQL package.

Note: All of the Management Plug-in reports should set p_system_report to 1. This parameter defines the report definition as a SYSTEM report, which cannot be deleted or edited by Enterprise Manager administrators. p_owner should be set to `mgmt_user.get_repository_owner` for all Management Plug-in reports.

The p_component_name must be set to the target type of the management plug-in.

Input

Parameter	Description
p_title_nlsid	report title.
p_description_nlsid	description
p_owner	owner name (should be the value returned from mgmt_user.get_repository_owner for Plug-In reports)
p_category_nlsid	category name
p_sub_category_nlsid	subcategory name
p_late_binding_target_types	target type for late binding, or null if not late binding
p_show_table_of_contents	1=show 0=hide
p_system_report	1=system report, 0=end user report. This must be set to 1 for Management Plug-in reports.
p_show_navigation	Show navigation headers in report (tabs, etc) 1=show, 0=hide
p_product_name	Product name, 'EM'(default)
p_component_name	Product component. This must be set to the Management Plug-in target type.
p_version	Version, '10.2' (default)
p_parameters	Parameters for this report definition

Output

Returns the GUID for this report definition.

Code

```
FUNCTION create_report_definition (
     p_title_nlsid                      IN VARCHAR2,
     p_description_nlsid                IN VARCHAR2,
     p_owner                            IN VARCHAR2,
     p_category_nlsid                   IN VARCHAR2,
     p_sub_category_nlsid               IN VARCHAR2,
     p_late_binding_target_types        IN MGMT_IP_TARGET_TYPES DEFAULT NULL,
     p_show_table_of_contents           IN NUMBER DEFAULT 0,
     p_system_report                    IN NUMBER DEFAULT 1,
     p_show_navigation                  IN NUMBER DEFAULT 1,
     p_product_name                     IN VARCHAR2 DEFAULT 'EM',
     p_component_name                   IN VARCHAR2 DEFAULT '',
     p_version                          IN VARCHAR2 DEFAULT '10.2.0.1.0',
     p_parameters                       IN MGMT_IP_PARAM_VALUE_LIST DEFAULT NULL
     ) RETURN RAW;
```

6.4.1.2 mgmt_ip.create_report_definition

Call this method to add a new report element to an existing report definition. The add_element_to_report_def method is part of the MGMT_IP PL/SQL package.

Input

Parameter	Description
p_report_guid	GUID to identify the report definition.
p_element_name_nlsid	The element name.
p_element_type_nlsid	The element type name.
p_header_nlslid	The element header or null.
p_element_order	The order of this element, 1 based.
p_element_row	The row for this element, 1 based.
p_parameters	The parameters for this element.

Output

Returns the GUID for this element instance.

Code

```
FUNCTION add_element_to_report_def(
          p_report_guid                       IN RAW,
          p_element_name_nlsid                IN VARCHAR2,
          p_element_type_nlsid                IN VARCHAR2,
          p_header_nlsid                      IN VARCHAR2 DEFAULT NULL,
          p_element_order                     IN NUMBER,
          p_element_row                       IN NUMBER,
          p_parameters                        IN MGMT_IP_PARAM_VALUE_LIST,
          p_targets                           IN MGMT_IP_TARGET_LIST
          ) RETURN RAW;
```

6.4.1.3 mgmt_mp_homepage.add_report

Call this method to register a report for display in the Reports subtab on the target home page for a report. The add_report method is part of the MGMT_MP_HOMEPAGE PL/SQL package.

The input parameters p_target_type, p_report_title, and p_report_owner MUST be identical to the report definition being registered.

> **IMPORTANT:** The value returned from `mgmt_user.get_repository_owner` must be specified as the report owner in order for the report to appear on the Reports subtab of a target home page.

Input

Parameter	Description
p_target_type	The target type.
p_report_title	The report title.
p_report_owner	The report owner.
p_order	The order the report shows up in the homepage.

Output

None.

Code

```
PROCEDURE add_report(p_target_type IN VARCHAR2, p_report_title IN VARCHAR2,
                     p_report_owner IN VARCHAR2, p_report_order IN NUMBER);
```

6.4.1.4 mgmt_view_util.adjust_tz

Call this function to convert the time zone of `v_date_in` from the `v_from_tz` time zone to the `v_to_tz` time zone.

Input

Parameter	Description
v_date_in	Date to be converted to a different time zone
v_from_tz	time zone of date being converted.
v_to_tz	time zone into which date will be converted.

Output

Returns the adjusted date in the new time zone.

Code

```
FUNCTION ADJUST_TZ(v_date_in DATE,
                   v_from_tz VARCHAR2,
                   v_to_tz VARCHAR2)
RETURN DATE;
```

6.4.2 PL/SQL Type Definitions

Three PL/SQL types are required to use the PL/SQL methods documented in Section 6.4.1, "PL/SQL Methods for Creating Report Definitions" . The definitions for these types are shown below.

MGMT_IP_TARGET_TYPES

Use MGMT_IP_TARGET_TYPES type to pass the target type your report definition supports to the create_report_definition API as the p_late_binding_target_types parameter.

```
CREATE OR REPLACE TYPE MGMT_IP_TARGET_TYPES
AS TABLE OF VARCHAR(64);
```

Example 6–2 MGMT_IP_TARGET_TYPES

```
DECLARE
 l_target_type   MGMT_IP_TARGET_TYPES;
BEGIN
-- specify the target type associated with this report
-- in this case the target type is 'oracle_orgchart'
   l_target_type := MGMT_IP_TARGET_TYPES();
   l_target_type.extend(1);
   l_target_type(1) := 'oracle_orgchart';
END;
```

MGMT_IP_PARAM_VALUE_LIST and MGMT_IP_PARAM_VALUE_RECORD

Use the MGMT_IP_PARAM_VALUE_LIST type to pass parameter values to the create_report_definition API as the p_parameters parameter and to the add_element_to_report_def API as the p_paremeters parameter.

```
CREATE OR REPLACE TYPE MGMT_IP_PARAM_VALUE_LIST
AS TABLE OF MGMT_IP_PARAM_VALUE_RECORD;
```

Use the MGMT_IP_PARAM_VALUE_RECORD type to create a named parameter value pair to add to an object of type MGMT_IP_PARAM_VALUE_LIST.

```
CREATE OR REPLACE TYPE MGMT_IP_PARAM_VALUE_RECORD
AS OBJECT (PARAM VARCHAR2(100),  VALUE CLOB);
```

Example 6–3 MGMT_IP_PARAM_VALUE_RECORD and MGMT_IP_PARAM_VALUE_LIST

```
DECLARE
   l_param_values  MGMT_IP_PARAM_VALUE_LIST;
BEGIN
   l_param_values := MGMT_IP_PARAM_VALUE_LIST();
   l_param_values.extend(2);
   l_param_values(1) := MGMT_IP_PARAM_VALUE_RECORD(
     'oracle.sysman.eml.ip.render.elem.TextParamBean.textMessage',
     'Test Org Chart Monitoring');
   l_param_values(2) := MGMT_IP_PARAM_VALUE_RECORD(
     'oracle.sysman.eml.ip.render.elem.TextParamBean.textStyle',
     'OraInstructionText');
END;
```

6.4.3 Element Parameters

Parameters used by some report elements dictate the operational behavior of those elements. This section lists the parameters associated with specific report elements.

6.4.3.1 Table Element Parameters

The Table Element is used to show a tabular view of query results. The queries must be made against management views.

- Element Name: IPMSG_USER_TABLE_FROM_SQL
- Element Type: IPMSG_ANY_TARGET_TYPE

Time Period

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TimePeriodParam"
Required	No
Default Value	Null
Valid Values	"0:0" for last 24 Hours "0:1" for last 7 Days "0:2" for last 31 Days
Summary	Encoded time period.

Sort Column

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TableRender.initialSortColumn"
Required	No
Default Value	The first column in result set.
Valid Values	Any valid column name.
Summary	If this parameter is set, the sort column indicator will be shown for the column with this column name. If not set, the sort column indicator is shown on the first column. The SQL query should include an 'order by' clause that sorts by this column.

Sort Order

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TableRender.initialSortOrder"
Required	No
Default Value	"ascending"
Valid Values	"ascending" or "descending"
Summary	If this parameter is set, the sort column indicator will be shown either as ascending or descending, according to the value. If not set, the sort column indicator is shown as ascending.

Name Value Pair Display

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TableRender.nameValueDisplay"
Required	No
Default Value	
Valid Values	Positive integer value.
Summary	If this parameter is set and only one row is returned from the query, the results are displayed in a vertical list of name-value pairs. This value should be set to the number of name/value columns that should be displayed, normally "1".

Number of Rows to Show

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TableRender.numRowsToShow"
Required	No
Default Value	"10"
Valid Values	Positive integer value.
Summary	Number of rows to display at one time in the generated table. The user can scroll through additional rows using the UI controls..

Is PL/SQL Statement

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.sqlStatementIsPlSql"
Required	No

Attribute	Description
Default Value	"false"
Valid Values	"true" or "false"
Summary	Whether a SQL statement is PL/SQL.

SQL or PL/SQL Statement

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.sqlStatement"
Required	No
Default Value	<None>
Valid Values	Any valid SQL SELECT statement.
Summary	SQL statement can optionally bind values for targets, locale information, and start/end date. The format of the SQL statement should include a bind variable placeholders for the options to be bound. **Bind Placeholders** ■ ??EMIP_BIND_RESULTS_CURSOR?? For use with PL/SQL statement to bind a return cursor containing results for display. ■ ??EMIP_BIND_TARGET_GUID?? For use with SQL or PL/SQL to bind a target GUID. ■ ??EMIP_BIND_START_DATE?? For use with SQL or PL/SQL to bind a start date. ■ ??EMIP_BIND_END_DATE?? For use with SQL or PL/SQL to bind an end date. ■ ??EMIP_BIND_TIMEZONE_REGION?? For use with SQL or PL/SQL to bind a time zone region. ■ ??EMIP_BIND_LOCALE_COUNTRY?? For use with SQL or PL/SQL to bind a locale country. ■ ??EMIP_BIND_LOCALE_LANGUAGE?? For use with SQL or PL/SQL to bind a locale language. There should be no semi-colon (;) appended to the end of the SQL statement unless it is a PL/SQL statement.

Example 6–4 Specifying an anonymous PL/SQL block as a parameter to an element definition

```
l_param_values(1) := MGMT_IP_PARAM_VALUE_RECORD(
  'oracle.sysman.eml.ip.render.elem.sqlStatementIsPlSql',
  'true');
l_param_values(2) := MGMT_IP_PARAM_VALUE_RECORD(
  'oracle.sysman.eml.ip.render.elem.sqlStatement',
  'BEGIN
    DECLARE
        rep_tz          VARCHAR2(200);
        user_tz_in          VARCHAR2(200);
        TYPE CURSOR_TYPE IS REF CURSOR;
        result_cursor_out CURSOR_TYPE;
        start_date_in DATE DEFAULT NULL;
        end_date_in  DATE DEFAULT NULL;
```

```
        query_string   VARCHAR(6000);
    BEGIN
        result_cursor_out := ??EMIP_BIND_RESULTS_CURSOR??;
        start_date_in := ??EMIP_BIND_START_DATE??;
        end_date_in := ??EMIP_BIND_END_DATE??;
        user_tz_in := ??EMIP_BIND_TIMEZONE_REGION??;
        select TIMEZONE_REGION into rep_tz from mgmt$target where TARGET_TYPE =
''oracle_emrep'';
        query_string := ''WITH dates AS (SELECT
      mgmt_view_util.adjust_tz(:1,:2,:3) as start_date,
      mgmt_view_util.adjust_tz(:4,:5,:6) as end_date
   from dual)
  SELECT
       label,
       time,
       sum(violations) --) as violations
   FROM(
       (
        SELECT
           ''''''||''NEW_LBL''||'''''' as label,
           mgmt_view_util.adjust_tz(rollup_timestamp, :7, :8) as time,
           average as violations --new violations
        FROM
           MGMT$METRIC_HOURLY
        WHERE
           key_value in (''''18'''',''''20'''',''''25'''') AND
           ROLLUP_TIMESTAMP > (SELECT start_date FROM dates) AND
           ROLLUP_TIMESTAMP < (SELECT end_date FROM dates) AND
           target_type = ''''oracle_emrep'''' AND
           metric_name = ''''TARGET_SECURITY_NEW_VIOLATIONS'''' AND
           metric_column = ''''NEW_VIOLATIONS'''' )
       UNION
       (
        SELECT
           ''''''||''FIXED_LBL''||'''''' as label,
           mgmt_view_util.adjust_tz(rollup_timestamp,:9,:10) as time,
           average as violations --cleared violations
        FROM
           MGMT$METRIC_HOURLY
        WHERE
           key_value in (''''18'''',''''20'''',''''25'''') AND
           ROLLUP_TIMESTAMP > (SELECT start_date FROM dates) AND
           ROLLUP_TIMESTAMP < (SELECT end_date FROM dates) AND
           target_type = ''''oracle_emrep'''' AND
           metric_name = ''''TARGET_SECURITY_CLEARED_VIOLATIONS'''' AND
           metric_column = ''''CLEARED_VIOLATIONS''''
         )
       )
   GROUP BY time, label
   ORDER BY time ASC'';
   OPEN result_cursor_out for query_string using start_date_in, user_tz_in, rep
_tz, end_date_in, user_tz_in, rep_tz, user_tz_in, rep_tz, user_tz_in, rep_tz;
   END;
END;');
```

Maximum Number of Rows

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TableRender.maxNumberOfRowsAllowed"
Required	No
Default Value	"2000"
Valid Values	Any scalar numeric value.
Summary	Set the maximum number of rows retrieved for display in the table. For example, show the top 10 xyz's element would set the value to "10".

Null Data String Substitute

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TableRender.nullDataStringSubstitue"
Required	No
Default Value	""
Valid Values	A string.
Summary	A string that will be substituted for null values returned.

Split Table into Multiple Tables by Column

Attribute	Description
Parameter Name	TableRenderBean.TABLE_SPLIT_COLUMN
Parameter String	"oracle.sysman.eml.ip.render.elem.TableRender.tableSplitColumn"
Required	No
Default Value	null
Valid Values	Any valid column name.
Summary	If this parameter is set, the table will be split into separate tables with subheaders as the value in this column changes. The data should be ordered by this column.

Column Group Header

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TableRender.columnGroupHeader"n
Required	No
Default Value	null
Valid Values	Header string to use for a column group.
Summary	This parameter provides a column header string. This column group header will span columns between the columns specified in "oracle.sysman.eml.ip.render.elem.TableRender.columnGroupStart Col"n and oracle.sysman.eml.ip.render.elem.TableRender.columnGroupEndCol"n. The n suffix is a numeric value starting with 1 for the first column group, sequentially ascending for subsequent column groups.

Column Group Start Column

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TableRender.columnGroupStartCol"n

Attribute	Description
Required	No
Default Value	null
Valid Values	Any valid column name.
Summary	Specifies the first column for a given column group. The n suffix is a numeric value starting with 1 for the first column group, sequentially ascending for subsequent column groups.

Column Group End Column

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TableRender.columnGroupEndCol"n
Required	No
Default Value	null
Valid Values	Any valid column name.
Summary	Specifies the last column for a given column group. The n suffix is a numeric value starting with 1 for the first column group, sequentially ascending for subsequent column groups.

Use Separate Rows for Values within a Cell

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TableRender.useSeparateRowsColumns"
Required	No
Default Value	null
Valid Values	Comma separated list of valid column names.
Summary	If this parameter is set, the delimited values of the column with the given name specified will be displayed on separate rows within a containing row cell. More than one column can be designated for this treatment by adding comma-separated column names.

Use Separate Rows as Delimiters

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TableRender.useSeparateRowsDelimiter"
Required	No
Default Value	, (comma)
Valid Values	Any string.
Summary	A character used to delimit tokens within a string.

Severity Icon in Column

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TableRender.severityColumn"
Required	No
Default Value	null
Valid Values	Any valid column names.

Attribute	Description
Summary	A severity icon will be substituted for valid severity values returned. To omit an icon, your result set can contain null values in this column.

Availability Status Icon in Column

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TableRender.availabilityStatusColumn"
Required	No
Default Value	null
Valid Values	Any valid column names.
Summary	An availability status icon will be substituted for valid values returned. To omit an icon your result set can contains null values in this column.

Render Image in Column

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TableRender.imageFilenameColumns"
Required	No
Default Value	null
Valid Values	Comma separated list of column names.
Summary	Optional parameter to display the given image filename in the indicated columns. Indicate for which columns the given image should be rendered. Specify a comma separated list of column names. The image filename returned should contain a relative path starting with '/images' such as '/images/xyz.gif'. Normally, a SQL decode function would be used to translate a numeric value into the appropriate image filename.

Target Type Column

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TableRender.targetTypeColumns"
Required	No
Default Value	null
Valid Values	Comma separated list of column names.
Summary	Optional parameter to indicate for which columns the value returned should be used as an internal target type to be translated into a display string for that type. Specify a comma separated list of column names.

6.4.3.1.1 Filter Elements The following table elements are used to create search filters that allows users to filter on rows for multiple table columns. Three different filter types are permitted:

- text-value
- list of values obtained from a SQL query
- list of values obtained from a comma-separated list in the element definition

Define Filter Name

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TableRender.filterNames"
Required	Yes
Default Value	null
Valid Values	Comma separated list of filter names.
Summary	Defines filter names in a comma-separated list. This parameter also defines the ordering of filter elements.

Define Filter Prompt

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TableRender.filterPrompt<name>"
Required	Yes
Default Value	null
Valid Values	CF
Summary	Defines the prompt used in the Reports page for the filter *name*. The filter value is accessed from the report element's SQL statement via `??EMIP_BIND_PARAM<name>??`. Without any other filter-related parameters, this defines a filter which allows the user to provide a value via a text input field.

SQL Filter

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TableRender.filterSql<name>"
Required	No
Default Value	null
Valid Values	Any valid SQL SELECT statement.
Summary	Defines the SQL query used to populate a list of values for a filter *name* that is presented in the UI as a drill-down menu instead of the text input field.

List of Filter Names

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TableRender.filterList<name>"
Required	No
Default Value	null
Valid Values	Comma-separated list of values.
Summary	Defines a list of values for a filter *name* which is displayed in the UI as a drill-down menu.

Translate List of Filter Names

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TableRender.filterTranslateValues<name>"
Required	No

Attribute	Description
Default Value	no
Valid Values	yes or no
Summary	Defines whether the values provided by `filterSql` or `filterList` should be translated to the client locale.

Filter Tip Text

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TableRender.filterTip<name> "
Required	No
Default Value	null
Valid Values	Alpha-numeric text.
Summary	Defines the text for a tool tip shown if the user moves the mouse over the filter UI elements.

Default Filter Name

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TableRender.filterDefault<name>"
Required	No
Default Value	%
Valid Values	Alpha-numeric text string.
Summary	Defines a default value for filter *name*. If no default value is given, '%' is used instead.

Null Default Filter Name

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TableRender.filterDefaultsToNull<name>"
Required	No
Default Value	null
Valid Values	yes or no
Summary	When this defines the default value to be NULL instead of '%'.

Global Filter Elements

The following parameters act globally on the filter system.

Display Empty Table

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TableRender.filterStartEmpty"
Required	No
Default Value	
Valid Values	'yes or no

Attribute	Description
Summary	If the value of this parameter is 'yes', then the report initially displays an empty table. The table is populated when the user clicks on the filter button in the UI.

Empty Table Headers

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TableRender.filterEmptyTableHeaders"
Required	No
Default Value	null
Valid Values	comma-separated list of table headers
Summary	Defines the table headers used when starting with an empty table.

Table Header Type

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TableRender.filterEmptyTableHeaderTypes"
Required	No
Default Value	VARCHAR
Valid Values	Comma-separated list of column types.
Summary	This defines the table header types (column types) used if when starting with an empty table. This is a comma-separated list. If no header types are specified, the table header types default to VARCHAR.

Overwrite Table Header Text

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TableRender.filterHeaderText"
Required	No
Default Value	Search Filter
Valid Values	Comma separated list of column names.
Summary	Overwrites the default filter section header text.

Overwrite Default Filter Description

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TableRender.filterDescriptionText"
Required	No
Default Value	Enter values to filter what is shown in the table.
Valid Values	Alpha-numeric text string.
Summary	Overwrites the default filter section description text. .

Overwrite Default Filter Tip Text

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TableRender.filterTipText"

Attribute	Description
Required	No
Default Value	The search filter is case sensitive. Use '%' as a wildcard.
Valid Values	Alpha-numeric text string.
Summary	Overwrites the default filter section tip text.

Overwrite Default Button Text

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TableRender.filterButtonText"
Required	No
Default Value	OK
Valid Values	Alpha-numeric text string.
Summary	Overwrites the default filter button text. .

Empty Table Text

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TableRender.filterEmptyTableText"
Required	No
Default Value	(No rows returned)
Valid Values	Alpha-numeric text string.
Summary	Specifies the text to be shown in an empty table before the filter is run.

6.4.3.1.2 Hyperlinks Within Tables The following parameters are used to implement hyperlinks within tables and incorporate improved link navigation between manster/detail views. These hyperlinks link directly to Reports tab on the target homepage (bypassing the target selector page). This method is an alternative to using *oracle.sysman.eml.ip.render.elem.TableRender.columnDestReportTitle<num>*, which first takes the user to the target selector page.

Link to Report

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TableRender.columnDestHomepageReportTitle <num>"
Required	No
Default Value	null
Valid Values	Report definition link.
Summary	Same as oracle.sysman.eml.ip.render.elem.TableRender.columnDestReportTitle<num> except that a link to a report definition on the target homepage is created.

Display Number of Columns

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TableRender.numberOfColumnsShowed"
Required	No
Default Value	Number of columns in the SQL.
Valid Values	Number
Summary	Defines the number of columns from the element SQL to be displayed in the UI. Additional columns from the SQL query are hidden but can be used to create the hyperlinks to expose data in the detail report.

Display Target Name

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TableRender.columnDestTargetIndex<num>"
Required	No
Default Value	null
Valid Values	
Summary	Specifies the column (which may be hidden) that contains the target name. The target name is used in the link to populate the target selection on late binding reports.

Display Target Type

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TableRender.columnDestTypeIndex<num>"
Required	No
Default Value	null
Valid Values	
Summary	Specifies the column (which may be hidden) that contains the target type. The target type is used in the link to populate the target selection on late binding reports.

Display URL

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TableRender.columnDestURLIndex<num>"
Required	No
Default Value	null
Valid Values	
Summary	Specifies the column (which may be hidden) that contains an arbitrary URL for a given table element.

Example 6–5 Report definition defining a master report that allows you to drill down via a link to a detail report.

```
begin
  declare
    l_target_types MGMT_IP_TARGET_TYPES;
    l_param_classes MGMT_IP_PARAM_CLASSES;
```

```
    l_param_values MGMT_IP_PARAM_VALUE_LIST;
    l_report_guid RAW(16);
    l_element_guid RAW(16);

    begin
      l_report_guid := mgmt_ip.create_report_definition (
        p_title_nlsid => 'My Master Report',
        p_description_nlsid => 'A master report to show master/detail',
        p_owner => 'SYSMAN',
        p_category_nlsid => 'Test Reports',
        p_sub_category_nlsid => 'Master and Detail',
        p_late_binding_target_types => null,
        p_late_binding_multi_targets => 0,
        p_system_report => 0,
        p_component_name => 'oracle_database',
        p_show_table_of_contents => 1);
      l_param_values := MGMT_IP_PARAM_VALUE_LIST();
      l_param_values.extend(17);
      l_param_values(1) := MGMT_IP_PARAM_VALUE_RECORD (
        'oracle.sysman.eml.ip.render.elem.sqlStatement',
        'SELECT TARGET_NAME "Target Name", TARGET_TYPE "Target Type",
''/em/console/targets'' FROM MGMT$TARGET WHERE TARGET_NAME LIKE ??EMIP_BIND_
PARAMNAME?? AND TARGET_TYPE LIKE ??EMIP_BIND_PARAMTYPE??');
      l_param_values(2) := MGMT_IP_PARAM_VALUE_RECORD (
        'oracle.sysman.eml.ip.render.elem.TableRender.filterNames',
        'NAME,TYPE');
      l_param_values(3) := MGMT_IP_PARAM_VALUE_RECORD (
        'oracle.sysman.eml.ip.render.elem.TableRender.columnDestReportId1',
        'My Detail Report');
      l_param_values(4) := MGMT_IP_PARAM_VALUE_RECORD (

'oracle.sysman.eml.ip.render.elem.TableRender.columnDestParamColumnIndexes1',
        '0,1');
      l_param_values(5) := MGMT_IP_PARAM_VALUE_RECORD (
        'oracle.sysman.eml.ip.render.elem.TableRender.filterPromptNAME',
        'Name');
      l_param_values(6) := MGMT_IP_PARAM_VALUE_RECORD (
        'oracle.sysman.eml.ip.render.elem.TableRender.filterTipNAME',
        'Filter on the target names');
      l_param_values(7) := MGMT_IP_PARAM_VALUE_RECORD (
        'oracle.sysman.eml.ip.render.elem.TableRender.filterPromptTYPE',
        'Target Type');
      l_param_values(8) := MGMT_IP_PARAM_VALUE_RECORD (
        'oracle.sysman.eml.ip.render.elem.TableRender.filterTipTYPE',
        'Filter on the target types');
      l_param_values(9) := MGMT_IP_PARAM_VALUE_RECORD (
'oracle.sysman.eml.ip.render.elem.TableRender.filterStartEmpty', 'yes');
      l_param_values(10) := MGMT_IP_PARAM_VALUE_RECORD (
        'oracle.sysman.eml.ip.render.elem.TableRender.filterEmptyTableHeaders',
        'Target Name, Target Type');
      l_param_values(11) := MGMT_IP_PARAM_VALUE_RECORD (
        'oracle.sysman.eml.ip.render.elem.TableRender.filterSqlTYPE',
        'select distinct target_type from mgmt$target');
      l_param_values(12) := MGMT_IP_PARAM_VALUE_RECORD (
        'oracle.sysman.eml.ip.render.elem.TableRender.numberOfColumnsShowed',
        '2');
      l_param_values(13) := MGMT_IP_PARAM_VALUE_RECORD (
        'oracle.sysman.eml.ip.render.elem.TableRender.filterHeaderText',
        'My Filter Header');
      l_param_values(14) := MGMT_IP_PARAM_VALUE_RECORD (
```

```
        'oracle.sysman.eml.ip.render.elem.TableRender.filterTipText',
        'My Tip Text');
      l_param_values(15) := MGMT_IP_PARAM_VALUE_RECORD (
        'oracle.sysman.eml.ip.render.elem.TableRender.filterDescriptionText',
        'My Filter description');
      l_param_values(16) := MGMT_IP_PARAM_VALUE_RECORD (
        'oracle.sysman.eml.ip.render.elem.TableRender.filterButtonText',
        'My button text');
      l_param_values(17) := MGMT_IP_PARAM_VALUE_RECORD (
        'oracle.sysman.eml.ip.render.elem.TableRender.columnDestURLIndex2',
        '2');

      l_element_guid := mgmt_ip.add_element_to_report_def (
        p_report_guid => l_report_guid,
        p_element_name_nlsid => 'user_table_from_sql',
        p_element_type_nlsid => 'any_target_type',
        p_header_nlsid => 'My Master Report Table',
        p_element_order => 1,
        p_element_row => 1,
        p_parameters => l_param_values,
        p_targets => null);

      l_report_guid := mgmt_ip.create_report_definition (
        p_title_nlsid => 'My Detail Report',
        p_description_nlsid => 'A detail report to show master/detail reports.',
        p_owner => 'SYSMAN',
        p_category_nlsid => 'Test Reports',
        p_sub_category_nlsid => 'Master and Detail',
        p_late_binding_target_types => null,
        p_late_binding_multi_targets => 0,
        p_system_report => 0,
        p_component_name => 'oracle_database',
        p_show_table_of_contents => 1);
      l_param_values := MGMT_IP_PARAM_VALUE_LIST();
      l_param_values.extend(1);
      l_param_values(1) := MGMT_IP_PARAM_VALUE_RECORD (
        'oracle.sysman.eml.ip.render.elem.sqlStatement',
        'SELECT TARGET_NAME "Target Name", TYPE_VERSION "Version" FROM MGMT$TARGET
WHERE TARGET_TYPE LIKE ??EMIP_BIND_PARAM2?? AND TARGET_NAME LIKE ??EMIP_BIND_
PARAM1??');
      l_element_guid := mgmt_ip.add_element_to_report_def (
        p_report_guid => l_report_guid,
        p_element_name_nlsid => 'user_table_from_sql',
        p_element_type_nlsid => 'any_target_type',
        p_header_nlsid => 'My Detail Report Table',
        p_element_order => 1,
        p_element_row => 1,
        p_parameters => l_param_values,
        p_targets => null);

      commit;
    end;
end;
/
```

6.4.3.2 Chart Element

The Chart Element is used to show a graphical view of query results. The queries must be made against Management Repository views.

- Element Name: IPMSG_USER_CHART_FROM_SQL

- Element Type: IPMSG_ANY_TARGET_TYPE

Chart Type

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.ChartParamController.chartType"
Required	No
Default Value	"pieChart"
Valid Values	"barChart" or "lineChart" or "pieChart" or "timeSeriesChart" "timeSeriesBarChart"
Summary	Chart type to display.

Time Period

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TimePeriodParam"
Required	No
Default Value	Null
Valid Values	"0:0" for last 24 Hours "0:1" for last 7 Days "0:2" for last 31 Days
Summary	Encoded time period.

Fill

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.ChartParamController.fill"
Required	No
Default Value	"none"
Valid Values	"none", "absolute", or "cumulative"
Summary	Indicates if a line chart should fill the area under the lines. "none": no fill under lines. "absolute": lines are identical to "none" setting but with the area under the lines filled. "cumulative": causes the values for the lines to be added or stacked, then the areas underneath the lines are filled. Use caution when using the fill attribute to ensure there is no confusion for the report user as to whether the data in the chart is cumulative or absolute.

Height

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.ChartParamController.height"
Required	No

Attribute	Description
Default Value	"200"
Valid Values	*n*, where *n* is any String that will correctly parse to a positive integer.
Summary	Sets the display height of the chart in pixels.

Horizontal or Vertical

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.ChartParamController.visualOrientation"
Required	No
Default Value	"horizontal"
Valid Values	"horizontal" or "vertical"
Summary	Visual orientation of the chart. This attribute is only valid with the chartType attribute set to barChart or timeSeriesChart. The attribute does not affect the pieChart.

Legend Position

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.ChartParamController.legendPosition"
Required	No
Default Value	"east"
Valid Values	"default", "east", "south"
Summary	Specifies where the legend should be placed relative to the chart.

Is PL/SQL Statement

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.sqlStatementIsPlSql"
Required	No
Default Value	"false"
Valid Values	"true" or "false"
Summary	Set to "true" to indicate that the SQL statement is a PL/SQL statement.

SQL or PL/SQL Statement

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.sqlStatement"
Required	No
Default Value	<None>
Valid Values	Any valid SQL SELECT statement or PL/SQL block.

Attribute	Description
Summary	SQL or PL/SQL statement can optionally bind values for targets, locale information, and start/end date. The format of the statement should include a bind variable placeholders for the options to be bound. **Bind Placeholders** ■ ??EMIP_BIND_RESULTS_CURSOR?? For use with PL/SQL statement to bind a return cursor containing results for display ■ ??EMIP_BIND_TARGET_GUID?? For use with SQL or PL/SQL to bind a target GUID. ■ ??EMIP_BIND_START_DATE?? For use with SQL or PL/SQL to bind a start date. ■ ??EMIP_BIND_END_DATE?? For use with SQL or PL/SQL to bind an end date. ■ ??EMIP_BIND_LOCALE_COUNTRY?? For use with SQL or PL/SQL to bind a locale country. ■ ??EMIP_BIND_LOCALE_LANGUAGE?? For use with SQL or PL/SQL to bind a locale language There should be no semi-colon (;) appended to the end of the SQL statement unless it is a PL/SQL statement.

Stacked Bar Chart

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.ChartParamController.stacked"
Required	No
Default Value	"false"
Valid Values	"true" or "false"
Summary	Indicates if a bar chart should be stacked.

Chart Title

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.ChartParamController.title"
Required	No
Default Value	<None>
Valid Values	
Summary	Chart title to identify chart for Americans with Disabilities Act compliance.

Width

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.ChartParamController.width"
Required	No
Default Value	"400"
Valid Values	*n*, where *n* is any String that will correctly parse to a positive integer.

Attribute	Description
Summary	Specifies the display width of the element in pixels.

Y-Axis Label

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.yAxisLabel"
Required	No
Default Value	<None>
Valid Values	String
Summary	If this parameter is supplied, it is used as the y-axis label for charts that have an y-axis.

Slices as Percentage

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.ChartParamController.pieShowSlicePercentLabels"
Required	No
Default Value	<None>
Valid Values	"true" or "false"
Summary	If this parameter is supplied, it controls whether each slice is labeled with a percentage value. This attribute is ignored for chartTypes other than pieChart.

Show Values in Legend

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.ChartParamController.pieValuesInLegend"
Required	No
Default Value	"value"
Valid Values	"percent", "value" or "none"
Summary	For pie charts, this parameter specifies whether values for pie slices are included in the legend along with the label for the pie slice. The default value for this attributes is "value". If specified as either "percent" or "value" then the numeric value is displayed along with the pie slice label in the form, "pie slice label (numeric value)". If "percent" is specified, then the percentage out of the total of all slice values is calculated and displayed, otherwise, the raw value of the slice is displayed. To omit a value in the legend, specify "none" as a value for this parameter. This attribute is ignored for chartTypes other than pieChart.

6.4.4 Metric Details Element

Classes directly involving the Metric Details report element and its parameters are located in the *oracle.sysman.eml.ip.render.elem* package. This element also accesses parameter-related constants defined in the *oracle.sysman.emSDK.eml.EmlConstants* class.

- Element Name: IPMSG_METRIC_DETAILS
- Element Type: IPMSG_ANY_TARGET_TYPE

Target Type

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.MetDetInternalTargetType"
Required	No
Default Value	"oracle_database"
Valid Values	Any valid internal target type name.
Summary	The type of target to be shown in the graph.

Metric Name

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.MetDetSelectedMetric"
Required	Yes
Default Value	
Valid Values	Valid metric name according to target type selected.
Summary	Metric to be graphed.

Metric Column Name

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.MetDetSelectedMetricColumn"
Required	Yes
Default Value	
Valid Values	Valid column name according to the metric and target type selected.
Summary	Column of metric to be graphed.

Time Period

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TimePeriodParam"
Required	No
Default Value	null
Valid Values	"0:0" for last 24 Hours "0:1" for last 7 Days "0:2" for last 31 Days
Summary	Encoded time period.

Width

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.MetDetWidth"
Required	No
Default Value	300
Valid Values	*n*, where *n* is any String that will correctly parse to a positive integer.

Attribute	Description
Summary	Width of the image in pixels.

Height

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.MetDetHeight"
Required	No
Default Value	300
Valid Values	*n*, where *n* is any String that will correctly parse to a positive integer.
Summary	Height of the image in pixels.

Legend Position

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.MetDetLegendPosition"
Required	No
Default Value	
Valid Values	"south" (default), "east"
Summary	Position of the legend relative to the chart.

6.4.5 Text Element Parameters

The Text Element is used to display any message text you wish to provide for your report.

- Element Name: IPMSG_STYLED_TEXT
- Element Type: IPMSG_ANY_TARGET_TYPE

Message Text

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TextParamBean.textMessage"
Required	No
Default Value	"" (empty String)
Valid Values	Any message.
Summary	Set the message to display in the report.

Message Style

Attribute	Description
Parameter Name	"oracle.sysman.eml.ip.render.elem.TextParamBean.textStyleClass"
Required	No
Default Value	"OraInstructionText"
Valid Values	"OraInstructionText" "OraTipText"
Summary	Specifies the style class for the message text to adopt when displayed.

Link Destination

Attribute	Description
Parameter Name	"oracle.sysmn.eml.ip.render.elem.TextParamBean.textDestination"
Required	No
Default Value	None
Valid Values	Any URI.
Summary	Specifies an optional link destination for this text element.

6.4.6 Report-Wide Parameters

The following parameters apply to all reporting elements within the report definition.

Dynamic Time Selector

You can provide a dynamic time period selector for your report definition that allows the report user to choose a specific time period with which to view the report. The dynamic time period option is available only when viewing reports from the Reports tab, is not available on a target home page report. The time period used on a target home page report is the default time period set for the report definition. To specify this using the PL/SQL API when creating your report definition file, set the following parameters for your report definition.

Example 6–6 Dynamic Time Selector Report Definition Parameters

```
l_param_values := MGMT_IP_PARAM_VALUE_LIST();
   l_param_values.extend(3);

   -- this report has a time period associated with it
   l_param_values(1) := MGMT_IP_PARAM_VALUE_RECORD(
     'oracle.sysman.eml.ip.render.elem.TimePeriodOption',
     'true');

   -- the user can customize the time period while viewing the report
   l_param_values(2) := MGMT_IP_PARAM_VALUE_RECORD(
     'oracle.sysman.eml.ip.render.elem.TimePeriodUserCust',
     'true');

   -- set the default value to last 24 hours
   l_param_values(3) := MGMT_IP_PARAM_VALUE_RECORD(
     'oracle.sysman.eml.ip.render.elem.TimePeriodParam',
     '0:0');
```

Pass `l_param values` to `mgmt_ip.create_report_definition` as the `p_parameters` argument.

If you are using *Table from SQL* or *Chart from SQL* report elements, you can structure your SQL statement such that the start and end dates will be bound automatically for you by Information Publisher. You achieve this by inserting placeholders (for example, `??EMIP_BIND_START_DATE??`) for the start and end date values as shown in Example 6–7.

Example 6–7 Automatic Binding of Start and End Dates

```
'SELECT COLUMN_LABEL, ROLLUP_TIMESTAMP, AVERAGE
      FROM MGMT$METRIC_HOURLY
      WHERE TARGET_GUID = ??EMIP_BIND_TARGET_GUID??
```

```
AND METRIC_LABEL = ''Load''
AND KEY_VALUE = '' ''
AND ROLLUP_TIMESTAMP > ??EMIP_BIND_START_DATE??
AND ROLLUP_TIMESTAMP < ??EMIP_BIND_END_DATE??
ORDER BY ROLLUP_TIMESTAMP'
```

See the online help documentation for *Table from SQL* or *Chart from SQL* for detailed information.

6.5 Development Guidelines

Oracle recommends adhering to the following guidelines when defining the PL/SQL for a report definition file:

- **Use the PL/SQL API to create your report definition.**

  Report definitions should be created using the PL/SQL API rather than non-standard coding constructs. Using the PL/SQL API insulates your code from any Management Repository schema changes.

- **Every report definition added to a Management Plug-in must be defined as a SYSTEM report.**

  Each call to the `create_report_definition` method must pass `p_system_report => 1`. SYSTEM report definitions are handled differently in the Enterprise Manager console. For example, SYSTEM report definitions cannot be deleted or edited by administrators.

- **Specifying the report owner.**

  You must specify the value returned from `mgmt_user.get_repository_owner` as the owner for each report definition. Report definitions specifying any owner other than `mgmt_user.get_repository_owner` will not appear in the Reports subtab of the target home page. These report definitions will, however, appear in the Information Publisher Report Definitions page.

- **The component name must be set to the target type.**

  The component name must be set to the target type in order for Enterprise Manager to associate specific report definitions with a particular Management Plug-in. For example,

```
   BEGIN

-- specify the target type associated with this report
-- in this case the target type is 'oracle_orgchart'
   l_target_type := MGMT_IP_TARGET_TYPES();
   l_target_type.extend(1);
   l_target_type(1) := 'oracle_orgchart';
   l_report_title_nlsid := 'Org breakdown by title (oracle_orgchart)<>';
   l_report_owner := mgmt_user.get_repository_owner;

-- create a report definition for the report
   l_report_guid := mgmt_ip.create_report_definition (
     p_title_nlsid               => l_report_title_nlsid,
     p_description_nlsid         => 'Charts showing the breakdown by title',
     p_owner                     => l_report_owner,
     p_category_nlsid            => 'Test Org Chart Reports',
     p_sub_category_nlsid        => 'Interesting Org Data',
     p_late_binding_target_types => l_target_type,
     p_show_table_of_contents    => 0,
     p_system_report             => 1,
```

```
p_component_name              => l_target_type(1)
);
```

When Using *Chart from SQL* and *Table from SQL* elements

- If your element accepts a single non-aggregate target (only), which is the case for most Management Plug-in target types, you can take advantage of automatic time zone date adjustment built into the *Chart from SQL* and *Table from SQL* elements by setting the `oracle.sysman.eml.ip.render.elem.adjustTimes` parameter on your element to `'true'`. When this parameter is set, the start and end dates bound to your SQL query will be adjusted from the report time zone to the target time zone. Conversely, dates returned from the query will be adjusted from the target time zone to the report time zone.
- If your element accepts multiple targets or aggregate targets, you are responsible for handling time zone adjustment for your date values. You can obtain the report time zone from the `??EMIP_BIND_TIMEZONE_REGION??` bind variable. In order for the report viewer to understand the dates shown, dates displayed in a report must either conform to the report time zone or explicitly display the time zone associated with each date. The following examples illustrate common use cases.

Example 6–8 Adjusting a date returned in your select statement from the time zone of a given target to the report time zone.

```
select mgmt_view_util.adjust_tz(tbl.date, tgt.timezone_region,
??EMIP_BIND_TIMEZONE_REGION??)
from mgmt$target tgt, sometable tbl
where <your where clause here>
```

Example 6–9 Adjusting a report time period start and end dates used in the WHERE clause of your SELECT statement from the report time zone to your targets time zone

```
select <your selected columns here>
from mgmt$target tgt, sometable tbl
where
  tgt.target_guid = ??EMIP_BIND_TARGET_GUID?? and
  tbl.Mydate > MGMT_VIEW_UTIL.ADJUST_TZ(
??EMIP_BIND_START_DATE??,
??EMIP_BIND_TIMEZONE_REGION??,
tgt.TIMEZONE_REGION)
and
  tbl.Mydate < MGMT_VIEW_UTIL.ADJUST_TZ(
??EMIP_BIND_END_DATE??,
??EMIP_BIND_TIMEZONE_REGION??,
tgt.TIMEZONE_REGION)
```

Recommended Coding Practice

When calling the PL/SQL API methods, you should use named notation rather than positional notation. If you have => in your call, you are using named notation. For example,

```
l_report_guid := mgmt_ip.create_report_definition (
  p_title_nlsid               => 'Org breakdown by title (oracle_orgchart)',
  p_description_nlsid         => 'Charts showing the breakdown by title',
  p_owner                     => mgmt_user.get_repository_owner,
  p_category_nlsid            => 'Test Org Chart Reports',
  p_sub_category_nlsid        => 'Interesting Org Data',
  p_late_binding_target_types  => l_target_type,
  p_late_binding_multi_targets => 0,
```

```
p_show_table_of_contents     => 0,
p_system_report              => 1,
p_component_name             => 'oracle_orgchart'
);
```

Using named notation insulates your report definition from any code changes to future releases of the PL/SQL API and also helps make your code self-documenting.

7

Monitoring Using Web Services and JMX

You can extend Enterprise Manager Grid Control to monitor Web Services and JMX-instrumented applications for critical events, performance problems, error conditions, and statistics.

Enterprise Manager's ability to monitor WSDL and JMX-enabled targets enables you to consolidate monitoring and management operations. When added to the Enterprise Manager framework, Enterprise Manager functionality, such as notifications, jobs, and reporting, is automatically extended to these targets.

This chapter covers the following topics:

> **Note:** This chapter assumes knowledge of Management Plug-ins and the requisite target definition files. For information on Management Plug-in concepts, or developing and deploying Management Plug-ins, see Chapter 1, "Extending Monitoring" and Chapter 2, "Developing a Management Plug-in".

7.1 Overview

Using Enterprise Manager to monitor targets that expose a Web Services management interface, JMX-instrumented applications and servers, and standalone Java Virtual Machine (JVM) targets entails defining a new target type via Management Plug-ins.

Creating a new Management Plug-in consists of four basic steps:

1. Generate the target metadata and default collection files to be added to the Management Plug-in.

2. Create a Management Plug-in Archive containing the target definition files for one or more Management Plug-ins. A single archive may contain more than one Management Plug-in.
3. Import the Management Plug-in into Enterprise Manager.
4. Deploy the Management Plug-in to the appropriate Management Agent(s).

Procedural information for the monitoring targets can be found in the following sections:

- Section 7.2 discusses software components exposing an external interface that communicate across a network using a standard messaging protocol.
- Section 7.3 discusses J2EE applications running on an OC4J that are instrumented using JMX MBeans.
- Section 7.4 discusses standalone Java applications running on J2SE5.0 or higher that are instrumented using JMX MBeans.

Section 7.3 and Section 7.4 explain how to generate metadata and default collection files for your custom JMX-enabled application by guiding you through the MBeans for which you are interested in collecting data, and helping you define the MBeans as metrics in Enterprise Manager. Even if your standalone Java application is not instrumented through JMX, you can still monitor the JVMs it is running on by directly creating the built-in JVM target instances as defined in Section 7.8.3.

After the metadata and default collection files are created, you can follow the normal Management Plug-in mechanism to deploy your plug-in and create target instances of your Java application target type as discussed in Section 7.5 through Section 7.8.

7.2 Monitoring Using Web Services in Enterprise Manager

Web Services are loosely coupled software components that expose an external interface via the Web Service Definition Language (WSDL). These components communicate across a network using a standard messaging protocol called Simple Object Access Protocol (SOAP). The Management Agent's JMX/SOAP fetchlet supports SOAP communication.

Note: For more information about the Web Services standard, see the World Wide Web Consortium (W3C) website:

`HTTP://www.w3.org`

Prerequisites

- Enterprise Manager Grid Control Management Agent version 10.2.0.2 or greater installed on that host.
- Enterprise Manager Grid Control Management Server (OMS) version 10.2.0.2 or greater with which the Management Agent communicates.

7.2.1 Creating Metadata and Default Collection Files

Defining a target type to be monitored via a Web Services interface entails creating the requisite target definition files, which are required to collect metrics from resources that support the WSDL interface:

- Target Metadata

- Default Collection

Enterprise Manager provides an easy-to-use Web Services wsperfcli command-line tool that simplifies creating new Management Plug-ins by automatically generating these requisite files. Information retrieval is achieved via the SOAP/JMX fetchlet that is integrated with the Management Agent.

The command-line tool works by parsing a specified WSDL file for all operations, and enables you to select one or more operations to be invoked. If multiple port types are specified in the WSDL file, the tool prompts you to select one of them. Operations are listed along with their parameters. A Web Service operation can be one of four types:

- One Way
- Request Response
- Solicit Response
- Notification

The Request Response operation type is particularly useful: The selected operation could have primitive or complex parameters and results. The result of Web Service invocation is displayed in a table (the tool prompts you to provide labels for the table columns). You can also filter result attributes by specifying an Xpath expression (see the `columnOrder` property in the generated target metadata, Example 7–3). Filter attributes can be useful for complex return types from which only few attributes are interesting.

The Web Services command-line tool supports Web Services with the following binding and encoding styles:

- DOC/literal
- RPC/encoded

7.2.1.1 Web Services Command-line Tool Syntax

The Web Services command-line tool syntax is as follows:

```
wsperfcli [OPTIONS] < WSDL File | URL to WSDL>
```

The `wsperfcli` command accepts the following options:

- **-useSOAP11**Use SOAP version 1.1 instead of default 1.2
- **-useWSIF**Useful for WSDL with SOAP-type arrays
- **-usePROXY**Use proxy to connect to the Web Service
- **-useSSL**Use SSL to communicate with the Web Service

The command-line tool optionally accepts a WSDL file name or URL to locate the WSDL for a Web Service. For example, for a card-service Web Service, a WSDL URL would be as follows:

```
http://localhost:44861/card-service/card?WSDL
```

The command tool script requires access to the Enterprise Manager home directory (EM_HOME) to run. The tool defaults to ORACLE_HOME. The home directory setting can be set using the command-line argument `-DEM_HOME`. For example, `-DEM_HOME=/myEMHome`. Other optional arguments are `-DSOAP11` and `-DuseSSL`, which allow metadata to be generated to use an older version of the SOAP protocol and SSL for communication with the Web Service. The option `-useWSIF` is used for WSDL that use the old SOAP-style arrays (rpc/encoded Web Services).

The tool parses specified WSDL for all the port types and binding (supported protocols such as HTTP get/post, SOAP) to list all the operations. If there are multiple port types in WSDL, you will first be prompted to choose a port type.

7.2.1.2 Web Services Command-line Tool Security

The command-line tool does not invoke the Web Service; it generates metadata required by Enterprise Manager for target monitoring purposes via the WSDL file. When you run this tool, you only need read permission on the WSDL file or URL and permission to save generated files to the appropriate directory.

7.2.1.3 Generating the Files

Example 7–1 shows a sample WSDL file passed to the command-line tool to generate the target metadata and collection files.

Example 7–1 Sample WSDL File TargetWithWSMgmtInterface

```
<?xml version="1.0" encoding="UTF-8" ?>
<definitions
     name="TargetWithWSMgmtInterface"
     targetNamespace="http://tempuri.org"
     xmlns="http://schemas.xmlsoap.org/wsdl/"
     xmlns:tns="http://tempuri.org"
     xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
     xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
     xmlns:ns1="http://mypackage3/MyProdMgmtInterface.wsdl/types"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
     xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    >
    <types>
        <schema xmlns="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://tempuri.org" elementFormDefault="qualified"
             xmlns:tns="http://tempuri.org"
 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:soap11-enc="http://schemas.xmlsoap.org/soap/encoding/"/>
        <schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://mypackage3/MyProdMgmtInterface.wsdl/types"
             elementFormDefault="qualified"
xmlns:tns="http://mypackage3/MyProdMgmtInterface.wsdl/types"
             xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xmlns:soap11-enc="http://schemas.xmlsoap.org/soap/encoding/">
            <complexType name="ArrayOfstring">
                <complexContent>
                    <restriction base="soap11-enc:Array">
                        <attribute ref="soap11-enc:arrayType"
wsdl:arrayType="string[]"/>
                    </restriction>
                </complexContent>
            </complexType>
        </schema>
    </types>
    <message name="TargetWithWSMgmtInterfaceSEI1_getNumActiveThreads"/>
    <message name="TargetWithWSMgmtInterfaceSEI1_getUserSessionID">
        <part name="user" type="xsd:string"/>
    </message>
    <message name="TargetWithWSMgmtInterfaceSEI1_getUserSessionIDResponse">
        <part name="result" type="xsd:string"/>
```

```
    </message>
    <message name="TargetWithWSMgmtInterfaceSEI1_getNumActiveThreadsResponse">
        <part name="result" type="xsd:long"/>
    </message>
    <message name="TargetWithWSMgmtInterfaceSEI1_getActiveUsers"/>
    <message name="TargetWithWSMgmtInterfaceSEI1_getActiveUsersResponse">
        <part name="result" type="ns1:ArrayOfstring"/>
    </message>
    <portType name="ManagementInterfaceWS">
        <operation name="getActiveUsers">
            <input message="tns:TargetWithWSMgmtInterfaceSEI1_getActiveUsers"/>
            <output message="tns:TargetWithWSMgmtInterfaceSEI1_
getActiveUsersResponse"/>
        </operation>
        <operation name="getNumActiveThreads">
            <input message="tns:TargetWithWSMgmtInterfaceSEI1_
getNumActiveThreads"/>
            <output message="tns:TargetWithWSMgmtInterfaceSEI1_
getNumActiveThreadsResponse"/>
        </operation>
        <operation name="getUserSessionID" parameterOrder="user">
            <input message="tns:TargetWithWSMgmtInterfaceSEI1_getUserSessionID"/>
            <output message="tns:TargetWithWSMgmtInterfaceSEI1_
getUserSessionIDResponse"/>
        </operation>
    </portType>
    <binding name="TargetWithWSMgmtInterfaceSoapHttp"
type="tns:ManagementInterfaceWS">
        <soap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="getActiveUsers">
            <soap:operation soapAction="http://tempuri.org:getActiveUsers"/>
            <input>
                <soap:body use="encoded" namespace="http://tempuri.org"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                    parts=""/>
            </input>
            <output>
                <soap:body use="encoded" namespace="http://tempuri.org"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                    parts="result"/>
            </output>
        </operation>
        <operation name="getNumActiveThreads">
            <soap:operation soapAction="http://tempuri.org:getNumActiveThreads"/>
            <input>
                <soap:body use="encoded" namespace="http://tempuri.org"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                    parts=""/>
            </input>
            <output>
                <soap:body use="encoded" namespace="http://tempuri.org"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                    parts="result"/>
            </output>
        </operation>
        <operation name="getUserSessionID">
            <soap:operation soapAction="http://tempuri.org:getUserSessionID"/>
            <input>
                <soap:body use="encoded" namespace="http://tempuri.org"
```

```
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                    parts="user"/>
            </input>
            <output>
                <soap:body use="encoded" namespace="http://tempuri.org"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                    parts="result"/>
            </output>
        </operation>
    </binding>
    <service name="ManagementInterfaceWS">
        <port name="TargetWithWSMgmtInterfacePort"
binding="tns:TargetWithWSMgmtInterfaceSoapHttp">
            <soap:address
location="http://144.25.119.190:8989/WS-MyProdMgmtIntf-context-root/ManagementInte
rfaceWS"/>
        </port>
    </service>
</definitions>
```

Example 7–2 uses the WSDL file shown in Example 7–1. First, the tool parses the WSDL for all port types and bindings (supported protocols such as HTTP get/post or SOAP) to list all the operations. If there are multiple port types in the WSDL, the tool first prompts you to select a port type.

Example 7–2 Sample Web Services Command-Line Tool Session

```
All port types for specified WSDL:
0 TargetWithWSMgmtInterfacePort
All operations for specified port type:
0   long getNumActiveThreads()
1   string getUserSessionID(string user)
2   string [] getActiveUsers()
>> Enter the index of operation to select.: 0
```

Next, the tool prompts you to select attributes from the return type to add to the metric.

```
Return value(s) for operation:
0 /tns:TargetWithWSMgmtInterfaceSEI1_getNumActiveThreadsResponse/result <long>
>> Enter the name for this metric column: ActiveThreads
>> Enter the label for this metric : ActiveThreadsLabel
>> Is this column a key <y/n> [n]: y
>> Do you want to add another metric <y/n>? [n] :
>> Enter value(s) for operation arguments:
>> Is this metric for periodic collection <y/n>? [n] : y
>> Enter the frequency of collections in seconds:12
>> Do you want to pick another operation to add <y/n>? [n] :n
>> Enter the metadata file name (metadata/TargetWithWSMgmtInterface.xml).:
```

The command-line tool generates the metadata required to monitor the target type `TargetWithWSMgmtInterface` as shown in Example 7–3.

Example 7–3 TargetWithWSMgmtInterface Target Metadata File

```
<?xml version='1.0'?><!DOCTYPE TargetMetadata SYSTEM
"../dtds/TargetMetadata.dtd"><TargetMetadata META_VER="1.0"
TYPE="TargetWithWSMgmtInterface" CATEGORY_PROPERTIES="versionCategory">
<Display>
```

```
<Label NLSID="TargetWithWSMgmtInterface">TargetWithWSMgmtInterface</Label>
<ShortName NLSID="TargetWithWSMgmtInterface">TargetWithWSMgmtInterface</ShortName>
<Description
NLSID="TargetWithWSMgmtInterface">TargetWithWSMgmtInterface</Description>
</Display>
<Metric NAME="Threads" TYPE="TABLE">
<Display>
<Label NLSID="Threads">Threads</Label>
</Display>
<TableDescriptor>
<ColumnDescriptor NAME="activeThreads" TYPE="NUMBER">
<Display>
<Label NLSID="ActiveThreads">ActiveThreads</Label>
</Display>
</ColumnDescriptor>
</TableDescriptor>
<QueryDescriptor FETCHLET_ID="OJMX">
<Property NAME="machine" SCOPE="INSTANCE">HTTPMachine</Property>
<Property NAME="port" SCOPE="INSTANCE">HTTPPort</Property>
<Property NAME="metricType" SCOPE="GLOBAL" OPTIONAL="TRUE">GWS</Property>
<Property NAME="metric" SCOPE="GLOBAL">getNumActiveThreads</Property>
<Property NAME="requestBodyElement" SCOPE="GLOBAL"
OPTIONAL="TRUE">tns:getNumActiveThreads</Property>
<Property NAME="authuser" SCOPE="INSTANCE" OPTIONAL="TRUE">authUser</Property>
<Property NAME="authpwd" SCOPE="INSTANCE" OPTIONAL="TRUE">authPasswd</Property>
<Property NAME="documentType" SCOPE="GLOBAL"
OPTIONAL="TRUE">rpc/encoded</Property>
<Property NAME="soapVersion" SCOPE="GLOBAL" OPTIONAL="TRUE">SOAP1.1</Property>
<Property NAME="targetNamespace" SCOPE="GLOBAL"
OPTIONAL="TRUE"><![CDATA[<namespace prefix="tns" uri="http://tempuri.org"
/>]]></Property>
<Property NAME="columnOrder"
SCOPE="GLOBAL">/tns:getNumActiveThreadsResponse/result</Property>
<Property NAME="additionalNamespaces" SCOPE="GLOBAL"
OPTIONAL="TRUE"><![CDATA[<namespaces><namespace prefix="tns"
uri="http://tempuri.org" /><namespace prefix="mime"
uri="http://schemas.xmlsoap.org/wsdl/mime/" /><namespace prefix="soap12"
uri="http://schemas.xmlsoap.org/wsdl/soap12/" /><namespace prefix="ns1"
uri="http://mypackage3/MyProdMgmtInterface.wsdl/types" /><namespace prefix="xsd"
uri="http://www.w3.org/2001/XMLSchema" /><namespace prefix="soap"
uri="http://schemas.xmlsoap.org/wsdl/soap/" /></namespaces>]]></Property>
<Property NAME="delimiter" SCOPE="GLOBAL">|</Property>
<Property NAME="URI"
SCOPE="GLOBAL">/WS-MyProdMgmtIntf-context-root/ManagementInterfaceWS</Property>
<Property NAME="soapAction" SCOPE="GLOBAL"
OPTIONAL="TRUE">http://tempuri.org:getNumActiveThreads</Property>
<Property NAME="name" SCOPE="GLOBAL">getNumActiveThreads</Property>
<Property NAME="returnType" SCOPE="GLOBAL">long</Property>
<Property NAME="arguments"
SCOPE="GLOBAL"><![CDATA[<arguments></arguments>]]></Property>
</QueryDescriptor>
</Metric>
<InstanceProperties>
<InstanceProperty NAME="HTTPMachine"><Display>
<Label NLSID="dms_HTTPMachine_iprop">Machine name</Label>
<ShortName NLSID="dms_HTTPMachine_iprop">Machine name</ShortName>
</Display>
</InstanceProperty>
<InstanceProperty NAME="HTTPPort"><Display>
<Label NLSID="dms_HTTPPort_iprop">Port</Label>
```

```
<ShortName NLSID="dms_HTTPPort_iprop">Port</ShortName>
</Display>
</InstanceProperty>
<InstanceProperty NAME="version" OPTIONAL="TRUE"><Display>
<Label NLSID="oc4j_version_iprop">Version of TargetWithWSMgmtInterface</Label>
<ShortName NLSID="oc4j_version_iprop">Version of
TargetWithWSMgmtInterface</ShortName>
</Display>
</InstanceProperty>
<InstanceProperty NAME="authUser" OPTIONAL="TRUE"><Display>
<Label NLSID="dms_authUser_iprop">Username for basic authorization</Label>
<ShortName NLSID="dms_authUser_iprop">Username for basic authorization</ShortName>
</Display>
</InstanceProperty>
<InstanceProperty NAME="authPasswd" OPTIONAL="TRUE" CREDENTIAL="TRUE"><Display>
<Label NLSID="dms_authPasswd_iprop">Password for basic authorization</Label>
<ShortName NLSID="dms_authPasswd_iprop">Password for basic
authorization</ShortName>
</Display>
</InstanceProperty>
</InstanceProperties>
</TargetMetadata>
```

The command-line tool also generates the requisite collection file as shown in Example 7–4.

Example 7–4 TargetWithWSMgmtInterface Default Collection File

```
<!DOCTYPE TargetMetadata SYSTEM "../dtds/TargetCollection.dtd"><!-- This file is
generated by Collector at 2006-01-24 14:00:10 -->

<TargetCollection TYPE="TargetWithWSMgmtInterface"
NAME="TargetWithWSMgmtInterface" INCLUDE_DEFAULT="TRUE">
<CollectionItem NAME="Threads" UPLOAD="5">
<Schedule>
<IntervalSchedule INTERVAL="12" TIME_UNIT="Sec"/>
</Schedule>
</CollectionItem>

</TargetCollection>
```

After the tool generates the target metadata and collection files, you can create the Management Plug-in archive. See Section 7.5, "Creating a Management Plug-in Archive".

7.3 Monitoring JMX Applications Deployed on Oracle Application Servers

The Java Management Extensions (JMX) framework improves manageability of your JMX-instrumented applications by enabling you to see what is happening inside. You gain insight into your applications and infrastructure through modular plug-ins called Managed Beans (MBeans). MBeans integrate with your application, components (such as Enterprise Java-Beans), or other resources to expose attributes (values) and operations.

The OJMX fetchlet, supplied with 10.2 Management Agents, enables you to monitor key metrics in your JMX-instrumented applications deployed on Oracle Application Server 10.1.3 or above. The fetchlet extends monitoring capabilities via JMX to the J2EE 1.4-compliant Oracle containers for J2EE (OC4J) servers themselves.

Monitoring JMX-instrumented applications/servers with Enterprise Manager entails defining a new target type that Enterprise Manager can monitor via Management Plug-ins. As with the Web Services wsperfcli command-line tool, Enterprise Manager provides an `emjmxcli` command-line tool to automate the generation of the target metadata and collection files.

Prerequisites

- Oracle Application Server 10.1.3 instance running on a specific host with a JMX-enabled application deployed on it that needs to be monitored as a target in Enterprise Manager Grid Control.
- Enterprise Manager Grid Control Management Agent version 10.2.0.2 or greater installed on that host.
- Enterprise Manager Grid Control Management Server (OMS) version 10.2.0.2 or greater with which the Management Agent communicates.

Known Limitations

- Currently, the `emjmxcli` tool and OJMX fetchlet only allow you to browse and monitor MBeans (system and application-defined) that are available on the default MBeanserver on the target OC4J instance.
- The `emjmxcli` tool primarily handles attributes and parameter and return values for operations that are OpenTypes. Examples: SimpleTypes, CompositeTypes, TabularTypes, and arrays of SimpleTypes.

7.3.1 Creating Metadata and Default Collection Files

As with Web Services, the JMX command-line tool (`emjmxcli`) simplifies creating the requisite target definition files: metadata and the default collection file. The tool is an offline configuration utility that connects you to an MBeanServer and enables you to browse available MBeans. It can also append metrics to an existing set of files during a subsequent invocation of the tool.

During a command-line tool session, you select specific MBeans and then choose the desired attributes/statistical values or operations Enterprise Manager needs to retrieve or invoke periodically on these MBeans to collect these values. The tool helps define packaging for these collected values as one or more Enterprise Manager metrics (with columns), and also enables you to specify a metric collection interval.

7.3.1.1 JMX Command-line Tool Syntax

The JMX command-line tool syntax is as follows for a JMX-enabled target on an OC4J:

```
emjmxcli <TARGET_HOME>
                [  -h <hostname>
                   -p <port>
                   -u <username>
                   -c <credential/password>
                   -w <work directory>
                   -e <true/false>
                   [-m <MBeanName> | -d <jmx_domain> | -s <mBeanPattern>]
                ]
```

`<TARGET_HOME>` is an Oracle Home directory 10.1.3 or greater Oracle Application Server Container for J2EE (OC4J).

The `emjmxcli` command accepts the following options:

- **-h** Hostname of the OC4J. Default: "localhost"
- **-p** RMI/RMIS port of the OC4J. Default: "23791"
  From the `ORACLE_HOME/opmn/bin` directory of your Application Server 10.1.3.0 or later instance, run `opmnctl status -l` to determine the RMI port for the OC4J for which MBeans were deployed.
- **-u** Valid username for the OC4J. Default: "oc4jadmin"
- **-c** Password associated with the OC4J user specified by the `-u` option. Default: None. If you do not specify a password, you are prompted for the password.
- **-w** Directory where the metadata and default collection files created by the JMX command-line tool are placed. Default: Current directory. When invoking the command-line tool, you must have write permission on this directory to create subdirectories and add files. If the metadata and default collection files already exist within that directory, you have the option of appending to or overwriting the original files.
- **-e** Whether or not the RMIS connection to the OC4J is enabled (true or false). Default: false

You can also specify ONE of the following three parameters (`-m`, `-d` or `-s`) to retrieve a subset of MBeans available on the MBeanServer. By default, all MBeans on the MBeanServer are displayed for you to select from if none of these parameters are specified.

- **-m** MBean ObjectName of the required MBean that needs to be retrieved and examined. If this is an ObjectName pattern-matching multiple MBeans, you are shown a list of all MBeans that match the pattern, and you can select one at a time to work on.
- **-d** MBean domain of the required application whose MBeans need to be retrieved and examined. For example, you want to browse all MBeans for an application (myApp). MBeans for this application would be available in the JMX domain "myApp".
- **-s** MBean pattern-matching an existing set of MBeans from which the metrics are to be defined. The `-s` parameter allows bulk retrieval of JMX Attributes/Statistics from multiple MBeans of a similar type.

  If you specify the `-s` parameter, the resulting metrics created during this `emjmxcli` session appear as a table in the Enterprise Manager console with multiple rows — one row representing each MBean that matches the specified pattern, and with the MBean ObjectName as a key column. For example, if you specify `-s 'oc4j:j2eeType=Servlet,*'` the resulting metric will have multiple rows, one for each servlet that matches the ObjectName pattern. Besides the MBean ObjectName column, other columns would be the attributes or fields from the return object of the operation, selected during the `emjmxcli` session.

7.3.1.2 Generating the Files

To start the JMX command-line tool:

1. Go to the `$AGENT_HOME/bin` directory.
2. Run the following command:

   ```
   emjmxcli <Oracle Home of the target 10.1.3 or greater OC4J> [OPTIONS]
   ```

   Once invoked, the command-line interface automatically prompts you for the requisite information, as shown in Example 7–6. If you need to abort a JMX

command-line tool session, you can press Ctrl+C at any point to exit. Session information will not be saved.

Example 7–5 Sample EMJMXCLI Invocation

```
./emjmxcli /scratch/shiphomes/oc4j/1013_SOA_M1/ -h localhost -p 12404 -m
'oc4j:J2EEApplication=orabpel,name=\"ServerBean\",*'
```

Example 7–6 Sample EMJMXCLI Session

```
oracleHome=/ade/sparmesw_10202_ssm/oracle
targetHome=/scratch/shiphomes/oc4j/1013_SOA_M1/
The Port is 12404

Connecting to server: localhost:12404
Connecting as user: oc4jadmin
Enter the password:

Obtained 1 MBeans matching pattern
oc4j:J2EEApplication=orabpel,name="ServerBean",*.

Enter the target type for this metric: [myJ2EEApp] myBPELApp
```

This is the target type for the new J2EE application as it should show up within Enterprise Manager.

```
Enter the target version: [1.0]

Enter the target metadata file: [./metadata/myBPELApp.xml]
```

This is the location of the metadata file that emjmxcli generates. You must have write permission on the directories where the target metadata and default collection files are to be created.

```
Enter the default collections file: [./default_collection/myBPELApp.xml]
The file ./metadata/myBPELApp.xml already exists.
Do you want to overwrite the existing file, append to it, or quit <o/a/q>? [a] a
Appending to existing file: ./metadata/myBPELApp.xml.
The available targets are:
0: Identifies a deployed stateless session bean
        (oc4j:EJBModule="ejb_ob_
engine",J2EEApplication=orabpel,J2EEServer=standalone,j2eeType=StatelessSessionBea
n,name="ServerBean")

Enter the index of target/MBean you wish to monitor or press <Ctrl-C> to quit: 0
```

If multiple MBeans matched the -m <MBean pattern> specified when emjmxcli was invoked, all MBean ObjectNames matching the pattern are listed during this part of the command-line session, at which point you can select one among the list. You can choose another MBean from the above list after creating metrics for the first one without exiting this emjmxcli session.

If you want to append metrics from another MBean that does not match the above -m pattern, you must exit and start another emjmxcli session with the MBean ObjectName/Pattern of the latter MBean, and create metrics from this MBean which will be appended to the original target metadata and default collection files from the previous emjmxcli session. Using this method, you can append metrics created from multiple emjmxcli sessions to the same target metadata and default collection files, if necessary.

```
Following metric source types are available for selected target(s):
```

```
        0: JMX Attributes
        1: JMX Operations
        2: J2EE Statistics
Enter the index of your choice or press <Ctrl-C> to quit: 2

Statistics are:
        0: CreateCount
        1: ejbCreate()ClientActive
        2: ejbCreate()ClientTime
        3: ejbRemove()ClientActive
        4: ejbRemove()ClientTime
        5: MethodReadyCount
        6: RemoveCount
        7: setSessionContext(javax.ejb.SessionContext)ClientActive
        8: setSessionContext(javax.ejb.SessionContext)ClientTime
Select one or more items as comma-separated indices: 0,6

JavaBean is : CreateCount
        0: count
        1: description
        2: lastSampleTime
        3: name
        4: startTime
        5: unit
```

This indicates that the Statistic call CreateCount is not a simple data type, but has a JavaBean pattern with the above listed properties, of which some may interest you.

```
Select one or more items as comma-separated indices: 0
JavaBean is : RemoveCount
        0: count
        1: description
        2: lastSampleTime
        3: name
        4: startTime
        5: unit
Select one or more items as comma-separated indices: 0
Number of possible columns in the resultant metric are 2.

Enter the name for this metric column at index=0 : [countOfCreateCount]
createCount
```

You can specify any meaningful name here. If you do not specify a name, the JMX command-line tool generates a default name that may not be appropriate in all cases.

```
Is this column a KEY Column <y/n>? [n]
```

In situations where multiple rows can be returned, as might be the case when the Attribute or return value of the Operation is TabularData, you need to specify one or more of your chosen metrics as "Key" columns.

```
Is this column for SUMMARY_UI <y/n>? [n]
Enter the label for column: [createCount]
Enter the NLSID for column: [createCount]
Enter the UNIT for column "createCount": [count]
Do you want to create a threshold for this column <y/n>? [n] y
Creating threshold!!
Following operators are available for creating thresholds:
 0: GT
 1: EQ
 2: LT
```

```
 3: LE
 4: GE
 5: CONTAINS
 6: NE
 7: MATCH
```

If you want to create a threshold on this column, you can specify an operator and then a value that would trigger a CRITICAL or WARNING alert.

```
Enter the index of your choice or press <Ctrl-C> to quit: 0
Enter the CRITICAL threshold: [NotDefined] 100
Enter the WARNING threshold: [NotDefined] 85
Enter the number of occurrences that trigger threshold: [6] 3
```

This is the number of consecutive occurrences of above CRITICAL or WARNING values that would trigger an alert.

```
Enter the message to be used when threshold is triggered: [createCount is %value%
and has crossed warning (%warning_threshold%) or critical (%critical_threshold%)
threshold.]
Enter NLSID for the message used when threshold is triggered: [createCount_cond]
Enter the name for this metric column at index=1 : [countOfRemoveCount]
removeCount
Is this column a KEY Column <y/n>? [n]
Is this column for SUMMARY_UI <y/n>? [n]
Enter the label for column: [removeCount]
Enter the NLSID for column: [removeCount]
Enter the UNIT for column "removeCount": [count]
Do you want to create a threshold for this column <y/n>? [n]
Enter the name of this metric: ServerBeanStats
Enter the label for this metric: [ServerBeanStats]

Do you want periodic collection for this metric <y/n>? [n] y
```

If the metric does not have to be collected periodically, as would be the case with real-time-only metrics, you can specify "no".

```
Enter the collection interval in seconds: 60
Periodic collection interval is: 60 seconds.

Do you want to create another metric <y/n>? [n] y
The available targets are:
0: Identifies a deployed stateless session bean
        (oc4j:EJBModule="ejb_ob_
engine",J2EEApplication=orabpel,J2EEServer=standalone,j2eeType=StatelessSessionBea
n,name="ServerBean")

Enter the index of target/MBean you wish to monitor or press <Ctrl-C> to quit: 0
```

If multiple MBeans match the MBean pattern for the -m option (specified when emjmxcli was invoked) you can select a different MBean from the above list for the next iteration of this command-line session.

```
Following metric source types are available for selected target(s):
        0: JMX Attributes
        1: JMX Operations
        2: J2EE Statistics

Enter the index of your choice or press <Ctrl-C> to quit: 0

Attributes are:
```

```
        0: activeInstances       Return Value: int
        1: activeInstancesHighWaterMark  Return Value: int
        2: eventProvider         Return Value: boolean
        3: maxInstances  Return Value: int
        4: minInstances  Return Value: int
        5: ObjectName    Return Value: javax.management.ObjectName
        6: stateManageable       Return Value: boolean
        7: statisticsProvider    Return Value: boolean
        8: stats         Return Value: javax.management.j2ee.statistics.Stats
        9: transactionTimeout    Return Value: int
Select one or more items as comma-separated indices: 0,3,4

Number of possible columns in the resultant metric are 3.
Enter the name for this metric column at index=0 : [activeInstances]
Is this column a KEY Column <y/n>? [n]
Is this column for SUMMARY_UI <y/n>? [n]
Enter the label for column: [activeInstances]
Enter the NLSID for column: [activeInstances]
Enter the UNIT for column "activeInstances": [millisec, kb etc.. ]
Do you want to create a threshold for this column <y/n>? [n]
Enter the name for this metric column at index=1 : [maxInstances]
Is this column a KEY Column <y/n>? [n]
Is this column for SUMMARY_UI <y/n>? [n]
Enter the label for column: [maxInstances]
Enter the NLSID for column: [maxInstances]
Enter the UNIT for column "maxInstances": [millisec, kb etc.. ]
Do you want to create a threshold for this column <y/n>? [n]

Enter the name for this metric column at index=2 : [minInstances]
Is this column a KEY Column <y/n>? [n]
Is this column for SUMMARY_UI <y/n>? [n]
Enter the label for column: [minInstances]
Enter the NLSID for column: [minInstances]
Enter the UNIT for column "minInstances": [millisec, kb etc.. ]
Do you want to create a threshold for this column <y/n>? [n]

Enter the name of this metric: ServerBeanCount
Enter the label for this metric: [ServerBeanCount]

Do you want periodic collection for this metric <y/n>? [n] y
Enter the collection interval in seconds: 300
Periodic collection interval is: 300 seconds.

Do you want to create another metric <y/n>? [n] n
Written the metadata xml file: ./metadata/myBPELApp.xml.
Updated the default collection file for myBPELApp at location ./default_
collection/myBPELApp.xml.
Exiting...
```

After the JMX command-line tool generates the target metadata and collection files, you can create the Management Plug-in archive. See Section 7.5, "Creating a Management Plug-in Archive". A sample of each generated file from the command-line tool session above is shown in Example 7–7 and Example 7–8.

Example 7–7 Generated Target Metadata File

```
<!DOCTYPE TargetMetadata SYSTEM "../dtds/TargetMetadata.dtd">
<TargetMetadata META_VER="1.0" TYPE="myBPELApp" CATEGORY
_PROPERTIES="VersionCategory">
   <Display>
```

```
      <Label NLSID="myBPELAppNLSID">myBPELApp</Label>
      <ShortName NLSID="myBPELAppShortName">myBPELApp</ShortName>
      <Description NLSID="myBPELAppDescription">myBPELApp</Description>
   </Display>

   <Metric NAME="ServerBeanStats" TYPE="TABLE">
      <Display>
         <Label NLSID="ServerBeanStats">ServerBeanStats</Label>
      </Display>
      <TableDescriptor>
         <ColumnDescriptor NAME="createCount" TYPE="NUMBER">
            <Display>
               <Label NLSID="createCount">createCount</Label>
               <Unit NLSID="count">count</Unit>
            </Display>
         </ColumnDescriptor>
         <ColumnDescriptor NAME="removeCount" TYPE="NUMBER">
            <Display>
               <Label NLSID="removeCount">removeCount</Label>
               <Unit NLSID="count">count</Unit>
            </Display>
         </ColumnDescriptor>
      </TableDescriptor>
      <QueryDescriptor FETCHLET_ID="OJMX">
         <Property NAME="machine" SCOPE="INSTANCE">HTTPMachine</Property>
         <Property NAME="OracleHome" SCOPE="INSTANCE">OracleHome</Property>
         <Property NAME="oc4jInstanceName" SCOPE="INSTANCE"
OPTIONAL="TRUE">OC4JInstanceName</Property>
         <Property NAME="jvmId" SCOPE="INSTANCE" OPTIONAL="TRUE">JVMId</Property>
         <Property NAME="mgmtWebSite" SCOPE="INSTANCE"
OPTIONAL="TRUE">MgmtWebSite</Property>
         <Property NAME="authuser" SCOPE="INSTANCE"
OPTIONAL="TRUE">authUser</Property>
         <Property NAME="authpwd" SCOPE="INSTANCE"
OPTIONAL="TRUE">authPasswd</Property>
         <Property NAME="metric" SCOPE="GLOBAL">ServerBeanStats</Property>
         <Property NAME="delimiter" SCOPE="GLOBAL">|</Property>
         <Property NAME="name" SCOPE="GLOBAL">getStatistics</Property>
         <Property NAME="signature"
SCOPE="GLOBAL">objectName,statNames,languageCode,countryCode</Property>
<Property NAME="returnType" SCOPE="GLOBAL">arrayOfComplexObjectBean</Property>
         <Property NAME="dontAddDefaultRowKey" SCOPE="GLOBAL">true</Property>
         <Property NAME="columnOrder"
SCOPE="GLOBAL">/CreateCount/count,/RemoveCount/count</Property>
         <Property NAME="arguments" SCOPE="GLOBAL">
            <![CDATA[<arguments>
   <argument>
      <value>oc4j:EJBModule="ejb_ob_
engine",J2EEApplication=orabpel,J2EEServer=standalone,j2eeType=StatelessSessionBea
n,name="ServerBean"</value>
   </argument>
   <argument>
      <value>CreateCount</value>
      <value>RemoveCount</value>
   </argument>
   <argument>
      <value>en</value>
   </argument>
   <argument>
      <value>US</value>
```

```
   </argument>
</arguments>]]>
         </Property>
      </QueryDescriptor>
   </Metric>

<Metric NAME="ServerBeanCount" TYPE="TABLE">
      <Display>
         <Label NLSID="ServerBeanCount">ServerBeanCount</Label>
      </Display>
      <TableDescriptor>
         <ColumnDescriptor NAME="activeInstances" TYPE="NUMBER">
            <Display>
               <Label NLSID="activeInstances">activeInstances</Label>
            </Display>
         </ColumnDescriptor>
         <ColumnDescriptor NAME="maxInstances" TYPE="NUMBER">
            <Display>
               <Label NLSID="maxInstances">maxInstances</Label>
            </Display>
         </ColumnDescriptor>
         <ColumnDescriptor NAME="minInstances" TYPE="NUMBER">
            <Display>
               <Label NLSID="minInstances">minInstances</Label>
            </Display>
         </ColumnDescriptor>
      </TableDescriptor>
      <QueryDescriptor FETCHLET_ID="OJMX">
         <Property NAME="machine" SCOPE="INSTANCE">HTTPMachine</Property>
         <Property NAME="OracleHome" SCOPE="INSTANCE">OracleHome</Property>
         <Property NAME="oc4jInstanceName" SCOPE="INSTANCE"
OPTIONAL="TRUE">OC4JInstanceName</Property>
         <Property NAME="jvmId" SCOPE="INSTANCE" OPTIONAL="TRUE">JVMId</Property>
         <Property NAME="mgmtWebSite" SCOPE="INSTANCE"
OPTIONAL="TRUE">MgmtWebSite</Property>
         <Property NAME="authuser" SCOPE="INSTANCE"
OPTIONAL="TRUE">authUser</Property>
         <Property NAME="authpwd" SCOPE="INSTANCE"
OPTIONAL="TRUE">authPasswd</Property>
         <Property NAME="metric" SCOPE="GLOBAL">ServerBeanCount</Property>
         <Property NAME="delimiter" SCOPE="GLOBAL">|</Property>
         <Property NAME="name" SCOPE="GLOBAL">getAttributes</Property>
         <Property NAME="signature"
SCOPE="GLOBAL">objectName,attributeNames,languageCode,countryCode</Property>
         <Property NAME="returnType"
SCOPE="GLOBAL">arrayOfComplexObjectBean</Property>
         <Property NAME="dontAddDefaultRowKey" SCOPE="GLOBAL">true</Property>
         <Property NAME="columnOrder"
SCOPE="GLOBAL">/activeInstances,/maxInstances,/minInstances</Property>
<Property NAME="arguments" SCOPE="GLOBAL">
            <![CDATA[<arguments>
   <argument>      <value>oc4j:EJBModule="ejb_ob_
engine",J2EEApplication=orabpel,J2EEServer=standalone,j2eeType=StatelessSessionBea
n,name="ServerBean"</value>
   </argument>
   <argument>
      <value>activeInstances</value>
      <value>maxInstances</value>
      <value>minInstances</value>
   </argument>
```

```
   <argument>
      <value>en</value>
   </argument>
   <argument>
      <value>US</value>
   </argument>
</arguments>]]>
         </Property>
      </QueryDescriptor>
   </Metric>

<Metric NAME="Response" TYPE="TABLE">
      <Display>
         <Label NLSID="Response">Response</Label>
      </Display>
      <TableDescriptor>
         <ColumnDescriptor NAME="Status" TYPE="NUMBER">
            <Display>
               <Label NLSID="Status">Status</Label>
            </Display>
         </ColumnDescriptor>
      </TableDescriptor>
      <QueryDescriptor FETCHLET_ID="OJMX">
         <Property NAME="machine" SCOPE="INSTANCE">HTTPMachine</Property>
         <Property NAME="OracleHome" SCOPE="INSTANCE">OracleHome</Property>
         <Property NAME="oc4jInstanceName" SCOPE="INSTANCE"
OPTIONAL="TRUE">OC4JInstanceName</Property>
         <Property NAME="jvmId" SCOPE="INSTANCE" OPTIONAL="TRUE">JVMId</Property>
         <Property NAME="mgmtWebSite" SCOPE="INSTANCE"
OPTIONAL="TRUE">MgmtWebSite</Property>
         <Property NAME="authuser" SCOPE="INSTANCE"
OPTIONAL="TRUE">authUser</Property>
         <Property NAME="authpwd" SCOPE="INSTANCE"
OPTIONAL="TRUE">authPasswd</Property>
         <Property NAME="metric" SCOPE="GLOBAL">Response</Property>
         <Property NAME="delimiter" SCOPE="GLOBAL">|</Property>
         <Property NAME="name" SCOPE="GLOBAL">getAttributes</Property>
         <Property NAME="signature"
SCOPE="GLOBAL">objectName,attributeNames,languageCode,countryCode</Property>
         <Property NAME="returnType"
SCOPE="GLOBAL">arrayOfComplexObjectBean</Property>
         <Property NAME="dontAddDefaultRowKey" SCOPE="GLOBAL">true</Property>
         <Property NAME="columnOrder" SCOPE="GLOBAL">/state</Property>
         <Property NAME="arguments" SCOPE="GLOBAL">
            <![CDATA[<arguments>
   <argument>

<value>oc4j:J2EEServer=standalone,j2eeType=J2EEApplication,name=orabpel</value>
   </argument>
   <argument>
      <value>state</value>
   </argument>
   <argument>
      <value>en</value>
   </argument>
   <argument>
      <value>US</value>
   </argument>
</arguments>]]>
         </Property>
```

```
      </QueryDescriptor>
   </Metric>

   <InstanceProperties>
      <InstanceProperty NAME="HTTPMachine">
         <Display>
            <Label NLSID="dms_HTTPMachine_iprop">Machine name</Label>
         </Display>
      </InstanceProperty>
      <InstanceProperty NAME="OracleHome">
         <Display>
            <Label NLSID="dms_OracleHome_iprop">Oracle home path</Label>
         </Display>
      </InstanceProperty>
<InstanceProperty NAME="OC4JInstanceName" OPTIONAL="TRUE"><Display><Label
NLSID="OC4JInstanceNameiprop">OC4JInstanceName</Label></Display>home</InstanceProp
erty>
      <InstanceProperty NAME="JVMId" OPTIONAL="TRUE"><Display><Label NLSID="JVMId_
iprop">JVMId</Label></Display>1</InstanceProperty>
      <InstanceProperty NAME="MgmtWebSite" OPTIONAL="TRUE"><Display><Label
NLSID="MgmtWebSite_
iprop">MgmtWebSite</Label></Display>default-web-site</InstanceProperty>
      <InstanceProperty NAME="URI" OPTIONAL="TRUE"><Display><Label
NLSID="URI">URI</Label></Display>/JMXSoapAdapter/JMXSoapAdapter</InstanceProperty>
      <InstanceProperty NAME="authUser" OPTIONAL="TRUE">
         <Display>
            <Label NLSID="dms_authUser_iprop">Username for Basic
authorization</Label>
         </Display>
      </InstanceProperty>
      <InstanceProperty NAME="authPasswd" OPTIONAL="TRUE" CREDENTIAL="TRUE">
         <Display>
            <Label NLSID="dms_authPasswd_iprop">Password for Basic
authorization</Label>
         </Display>
      </InstanceProperty>
      <InstanceProperty NAME="Version" OPTIONAL="TRUE"><Display><Label
NLSID="oc4j_version_iprop">Version of
myBPELApp</Label></Display>1.0</InstanceProperty>
   </InstanceProperties>
</TargetMetadata>
```

Example 7–8 Generated Metric Collection File

```
<!DOCTYPE TargetCollection SYSTEM "../dtds/TargetCollection.dtd">
<!-- This file is generated by Collector at 2006-04-28 12:11:55 -->

<TargetCollection TYPE="myBPELApp" INCLUDE_DEFAULT="TRUE">
        <CollectionItem NAME="ServerBeanStats" UPLOAD="YES">
                <Schedule>
                        <IntervalSchedule INTERVAL="60" TIME_UNIT="Sec"/>
                </Schedule>
                <MetricColl NAME="ServerBeanStats">
                      <Condition COLUMN_NAME="createCount" CRITICAL="100"
WARNING="85" OPERATOR="GT" OCCURRENCES="3" MESSAGE="createCount is %value% and has
crossed warning (%warning_threshold%) or critical (%critical_threshold%)
threshold." MESSAGE_NLSID="createCount_cond"/>
                </MetricColl>
        </CollectionItem>
        <CollectionItem NAME="ServerBeanCount" UPLOAD="YES">
```

```
                <Schedule>
                        <IntervalSchedule INTERVAL="300" TIME_UNIT="Sec"/>
                </Schedule>
                <MetricColl NAME="ServerBeanCount">
                </MetricColl>
        </CollectionItem>

        <CollectionItem NAME="Response" UPLOAD="YES">
                <Schedule>
                        <IntervalSchedule INTERVAL="30" TIME_UNIT="Sec"/>
                </Schedule>
                <MetricColl NAME="Response">
                        <Condition COLUMN_NAME="Status" CRITICAL="1"
WARNING="NotDefined" OPERATOR="NE" OCCURRENCES="2" MESSAGE="Status is %value% and
has crossed warning (%warning_threshold%) or critical (%critical_threshold%)
threshold." MESSAGE_NLSID="Status_cond"/>
                </MetricColl>
        </CollectionItem>
</TargetCollection>
```

7.3.2 Displaying Target Status Information

For the status information of your targets to appear correctly within the Enterprise Manager console, you must define a metric, called Response, that has a column, named Status, with a critical threshold set. The status of target instances of this type appears in the console as "Up" (available) if the metric value is below the critical threshold. When the threshold is exceeded, the target status appears as "Down" in the console.

You can create the Response metric in another `emjmxcli` session (append the metric to the metadata and collection files created in an earlier session). Example 7–9 illustrates adding a Response metric to previously generated metadata and collection files from a new command-line session.

Example 7–9 Adding a Response Metric

```
./emjmxcli /scratch/shiphomes
//oc4j/1013_PRODUCTION/ -p 12403 -c welcome1 -m 'oc4j:j2eeType=J2EEApplication,n
ame=orabpel,*'

oracleHome=/ade/sparmesw_10202_ssm/oracle
targetHome=/scratch/shiphomes//oc4j/1013_PRODUCTION/
The Port is 12403

Connecting to server: localhost:12403
Connecting as user: oc4jadmin

Obtained 1 MBeans matching pattern oc4j:j2eeType=J2EEApplication,name=orabpel,*.

Enter the target type for this metric: [myJ2EEApp] myBPELApp

Enter the target version: [1.0]

Enter the target metadata file: [./metadata/myBPELApp.xml]

Enter the default collections file: [./default_collection/myBPELApp.xml]
The file ./metadata/myBPELApp.xml already exists.

Do you want to overwrite the existing file, append to it, or quit <o/a/q>? [a] a
Appending to existing file: ./metadata/myBPELApp.xml.
```

```
The available targets are:
0: Identifies a J2EE application EAR that has been deployed
        (oc4j:J2EEServer=standalone,j2eeType=J2EEApplication,name=orabpel)

Enter the index of target/MBean you wish to monitor or press <Ctrl-C> to quit: 0
Following metric source types are available for selected target(s):
        0: JMX Attributes
        1: JMX Operations

Enter the index of your choice or press <Ctrl-C> to quit: 0

Attributes are:
        0: allAccessibleGroups   Return Value: java.util.Set
        1: allAccessibleUsers    Return Value: java.util.Set
        2: applicationRootDirectoryPath  Return Value: java.lang.String
        3: archivePath   Return Value: java.lang.String
        4: childApplicationNames         Return Value: [Ljava.lang.String;
        5: childApplications     Return Value: [Ljavax.management.ObjectName;
        6: dataSourcesDescriptor         Return Value: java.lang.String
        7: deploymentDescriptor  Return Value: java.lang.String
        8: ejbClassLoaderPath    Return Value: java.lang.String
        9: eventProvider         Return Value: boolean
        10: groups       Return Value: java.util.Set
        11: iiopStubs    Return Value: [B
        12: metricRulesDescriptor        Return Value: java.lang.String
        13: Modules      Return Value: [Ljavax.management.ObjectName;
        14: objectName   Return Value: java.lang.String
        15: ohsRouting   Return Value: boolean
        16: parentApplication    Return Value: javax.management.ObjectName
        17: parentApplicationName        Return Value: java.lang.String
        18: properties   Return Value: java.util.Properties
        19: proprietaryDeploymentDescriptor      Return Value: java.lang.String
        20: proxyInterfaceSQLObjects     Return Value: [Ljava.lang.String;
        21: routingId    Return Value: java.lang.String
        22: Server       Return Value: javax.management.ObjectName
        23: sharedLibraryImports   Return Value:
[Loracle.oc4j.admin.management.shared.SharedLibraryImport;
        24: startTime    Return Value: long
        25: state        Return Value: int
        26: stateManageable      Return Value: boolean
        27: statisticsProvider   Return Value: boolean
        28: syntheticWebModules  Return Value:
oracle.oc4j.admin.management.shared.WebModule
        29: users        Return Value: java.util.Set
        30: webSite      Return Value: java.lang.String
        31: webSiteBindings      Return Value: java.util.Map
Select one or more items as comma-separated indices: 25

Number of possible columns in the resultant metric are 1.

Enter the name for this metric column at index=0 : [state] Status
Is this column a KEY Column <y/n>? [n]
Is this column for SUMMARY_UI <y/n>? [n]
Enter the label for column: [Status]
Enter the NLSID for column: [Status]
Enter the UNIT for column "Status": [millisec, kb etc.. ]
Do you want to create a threshold for this column <y/n>? [n] y
Creating threshold!!
Following operators are available for creating thresholds:
 0: GT
```

```
 1: EQ
 2: LT
 3: LE
 4: GE
 5: CONTAINS
 6: NE
 7: MATCH
Enter the index of your choice or press <Ctrl-C> to quit: 6
Enter the CRITICAL threshold: [NotDefined] 1
Enter the WARNING threshold: [NotDefined]
Enter the number of occurrences that trigger threshold: [6] 2
Enter the message to be used when threshold is triggered: [Status is %value% and
has crossed warning (%warning_threshold%) or critical (%critical_threshold%)
threshold.]
Enter NLSID for the message used when threshold is triggered: [Status_cond]

Enter the name of this metric: Response
Enter the label for this metric: [Response]

Do you want periodic collection for this metric <y/n>? [n] y
Enter the collection interval in seconds: 30
Periodic collection interval is: 30 seconds.

Do you want to create another metric <y/n>? [n] n
Written the metadata xml file: ./metadata/myBPELApp.xml.
Updated the default collection file for myBPELApp at location ./default_collecti
on/myBPELApp.xml.
Exiting...

Please note that the Response metric collected in this emjmxcli session would be
appended to the metadata and default_collection file created in an earlier session
of the tool. (User can chose to overwrite the earlier file as well if they
specific  the "o" option to the following prompt)

Do you want to overwrite the existing file, append to it, or quit <o/a/q>? [a] a
```

7.4 Monitoring a Standalone JMX-instrumented Java Application or Java Virtual Machine (JVM) Target

> **Note:** If your Java application is not JMX-instrumented, but you want to monitor the J2SE 5.0 JVM on which it is running, go directly to Section 7.8.3, "Configuring a Standalone Java Application or JVM Target" to create target instances of type JVM. This enables you to monitor these JVMs in Enterprise Manager Grid Control, preferably from an Enterprise Manager Agent installed on the same host as your JVM. The pre-requisites and known limitations discussed below still apply, however.

Enterprise Manager provides an out-of-box JVM target type. This enables you to add and configure metrics from standalone J2SE1.5 JVMs that are enabled for remote management in Enterprise Manager Grid Control version 10.2.0.3 or greater.

If your standalone Java application exposes data through JMX MBeans as for a J2EE application deployed on an Oracle Container for J2EE, you can use the JMX command-line tool to define such an application as an Enterprise Manager target type and generate a metadata and default collection file for this target type. You can

monitor your standalone application targets from an Enterprise Manager Agent, preferably installed on the same host as your JVM. Multiple JVMs running on that host can be monitored by the same Enterprise Manager Agent.

You can collect metrics from user-defined MBeans on a standalone (J2SE1.5-based) JVM and place them into Enterprise Manager Grid Control using the JMX fetchlet. The fetchlet is designed for a standalone Sun J2SE1.5-based (or higher) JVM containing user-defined MBeans that use JMX OpenTypes as arguments and return values.

Prerequisites

- Java virtual machine J2SE 1.5 or higher instance running on a specific host. This JVM could be running a JMX-enabled application that exposes metrics via MBeans that need to be monitored as a target in Enterprise Manager Grid Control. If the application does not expose MBeans, the JVM itself could be monitored using the built-in JVM target type provided in Enterprise Manager Grid Control. See Section 7.8.3, "Configuring a Standalone Java Application or JVM Target" for more information.
- JMX agent enabled for local access. Set this system property when you start the JVM or Java application:

  `com.sun.management.jmxremote`
- Monitoring and management from remote systems enabled. Set this system property when you start the JVM:

  `com.sun.management.jmxremot.port=portNum`

  For additional information about enabling the JVM for remote management, see the following document:

  `http://java.sun.com/j2se/1.5.0/docs/guide/management/agent.html#remote`
- Enterprise `Manager` Grid Control Management Agent version 10.2.0.3 or greater installed on that host.
- Enterprise Manager Grid Control Management Server (OMS) version 10.2.0.3 or greater with which the Management Agent communicates.

Known Limitations

- Currently, the `emjmxcli` tool only allows you to browse and monitor MBeans (platform and application-defined) that are available on the default platform MBeanserver on the target JVM instance. The tool does not support monitoring a custom MBeanServer on the target JVM instance.
- The `emjmxcli` tool primarily handles attributes as well as parameter and return values for operations that are OpenTypes, such as SimpleTypes, CompositeTypes, TabularTypes, and arrays of SimpleTypes.

7.4.1 Generating Metadata and Default Collection Files

As with Web Services and the J2EE application on OC4J, the command-line tool (`emjmxcli`) simplifies creating the requisite target definition files: metadata and the default collection file for a standalone JMX-instrumented Java application. The tool is an offline configuration utility that connects you to an MBeanServer on a J2SE1.5 or higher JVM and enables you to browse available MBeans. It can also append metrics to an existing set of files during a subsequent invocation of the tool.

During a command-line tool session, you select specific MBeans and then choose the desired attributes/statistical values or operations Enterprise Manager needs to retrieve or invoke periodically on these MBeans to collect these values. The tool helps define packaging for these collected values as one or more Enterprise Manager metrics (with columns), and also enables you to specify a metric collection interval.

7.4.1.1 JMX Command-line Tool Syntax

The JMX command-line tool syntax is as follows:

```
emjmxcli -t JVM
                [  -l <JMXServiceURL>
                   -h <hostname>
                   -p <port>
                   -u <username>
                   -c <credential/password>
                   -w <work directory>
                   -e <true/false>
                   [-m <MBeanName> | -d <jmx_domain> | -s <mBeanPattern>]
                ]
```

The `emjmxcli` command accepts the following options:

- **-t JVM** Indicates that the MBeanServer is on a standalone JVM
- **-l** JMXServiceURL of the target JVM
- **-h** Hostname of the JVM. Default: "localhost" if the -l option is not specified
- **-p** RMI/RMIS port of the JVM. Default: "23791" if the -l option is not specified. From the `ORACLE_HOME/opmn/bin` directory of your Application Server 10.1.3.0 or later instance, run `opmnctl status -l` to determine the RMI port for the OC4J for which MBeans were deployed.
- **-u** Valid username for the JVM. Default: None
- **-c** Password for the above user. Default: None. The password is only used to retrieve data and is not stored anywhere.
- **-w** Work directory where the metadata and default collection files are created. Default: Current directory. When invoking the command-line tool, you must have write permission on this directory to create subdirectories and add files. If the metadata and default collection files already exist within that directory, you have the option of appending to or overwriting the original files.
- **-e** True for enabling the SSL connection to the JVM. Default: false

You can also specify ONE of the following three parameters (`-m`, `-d` or `-s`) to retrieve a subset of MBeans available on the MBeanServer. By default, all MBeans on the MBeanServer are displayed for you to select from if none of these parameters are specified.

- **-m** MBean ObjectName of the required MBean that needs to be retrieved and examined. If this is an ObjectName pattern-matching multiple MBeans, you are shown a list of all MBeans that match the pattern, and you can select one at a time to work on.
- **-d** MBean domain of the required application whose MBeans need to be retrieved and examined. For example, you want to browse all MBeans for an application (myApp). MBeans for this application would be available in the JMX domain "myApp".

- **-s** MBean pattern matching an set of similar MBeans from which the metrics are to be defined. The `-s` parameter allows bulk retrieval of JMX Attributes/Statistics from multiple MBeans of a similar type.

  If you specify the `-s` parameter, the resulting metrics created during this `emjmxcli` session appear as a table in the Enterprise Manager console with multiple rows — one row representing each MBean that matches the specified pattern, and with the MBean ObjectName as a key column. For example, if you specify `-s 'oc4j:j2eeType=Servlet,*'` the resulting metric will have multiple rows, one for each servlet that matches the ObjectName pattern. Besides the MBean ObjectName column, other columns would be the attributes or fields from the return object of the operation, selected during the `emjmxcli` session.

7.4.1.2 Generating the Files

The following steps explain how to prepare for and then use the JMX command-line tool to generate the files.

1. Bring up the standalone JVM instance with the MBeans. The following example shows an invocation of the JVM:

   ```
   JDK15/bin/java -Dcom.sun.management.jmxremote
   -Dcom.sun.management.jmxremote.port=6789
   -Dcom.sun.management.jmxremote
   -Dcom.sun.management.jmxremote.authenticate=false
   -Dcom.sun.management.jmxremote.ssl=false MyJMXEnabledApp $*
   ```

   The `emjmxcli` fetchlet connects to the port number above as a JSR-160 client.

2. Go to the `$ORACLE_HOME/bin` directory of the 10.2.0.3 or higher version of the Enterprise Manager Agent.

3. Set the environment variable as follows:

   ```
   setenv USER_JARS /myAppHome/<myJar1.jar>;/myAppHome/<myJar2.jar>
   ```

   This step is needed if custom classes are being returned in attributes and/or operations in any of the MBeans registered with the target MBeanServer. The Enterprise Manager Agent (fetchlet) can only effectively monitor attributes and/or operations that return JMX OpenTypes, but it could also handle Java Bean properties (through getters and setters) on any custom classes.

   > **Note:** If the application-defined MBeans are returning custom classes, you need to also set up the corresponding user jar file in the `CLASSPATH` of the Enterprise Manager Agent monitoring this application. To do this, manually insert the location of this jar into the `$ORACLE_HOME/sysman/config/classpath.lst` file on the Enterprise Manager Agent, then restart the Agent.

4. Run the following command:

   ```
   ./emjmxcli –t JVM –h localhost –p 6789 u <user> -c <password>
   ```

   where:

   - **-t** JVM indicates that the MBeanServer is running on a standard JVM
   - **-h** Hostname where the JVM is running
   - **-p** Port number that enables the JVM for JSR-160 remote access

You can also specify an `-l <JMXServiceURL>` option instead of `-h <host>` and `-p <port>` options.

You can invoke `emjmxcli` with a `-w <work directory>` option to create the metadata and default collection files in the specified work directory. If you do not specify `-w` when you start `emjmxcli`, it defaults to the current directory, which is the directory where you start `emjmxcli`.

Once invoked, the command-line interface automatically prompts you for the requisite information, as shown in Example 7–10. For most of the prompts, you can just press enter to use defaults. If you need to abort a JMX command-line tool session, you can press Ctrl+C at any point to exit. Session information will not be saved.

When the session concludes after you exit, the result will be a `myJ2EEApp.xml` file (or whatever target type you specified) as `metadata/myJ2EEApp.xml`, and a `default_collection/myJ2EEApp.xml` file if you specified periodic collection.

Sample EMJMXCLI Invocations

The following sample enables you to browse all MBeans on a remote MBeanServer:

```
./emjmxcli -t JVM -p 6789 (the host defaults to "localhost")
```

The following sample invokes the command-line interface and filters MBeans based on the MBeanPattern specified as the argument for the `-m` option:

```
./emjmxcli -t JVM -p 6789 -m "java.lang:*"
```

Example 7–10 Sample EMJMXCLI Session

```
oracleHome=/ade/sparmesw_emas_ml/oracle
userJars=

Connecting to server: localhost:6789
Connecting without authentication. For specifying username and password use
the
-u and -c options.

Obtained 14 MBeans matching pattern java.lang:*.

Enter the target type for this metric: [myJ2EEApp] myJavaApp

Enter the target version: [1.0]

Enter the target metadata file: [./metadata/myJavaApp.xml]

Enter the default collections file: [./default_collection/myJavaApp.xml]

Enter a label for this target type: [myJavaApp]

Enter a description for this target type: [myJavaApp]
The available targets are:
0: sun.management.CompilationImpl
        (java.lang:type=Compilation)
1: sun.management.MemoryManagerImpl
        (java.lang:name=CodeCacheManager,type=MemoryManager)

2: sun.management.GarbageCollectorImpl
        (java.lang:name=Copy,type=GarbageCollector)

3: sun.management.MemoryPoolImpl
        (java.lang:name=Eden Space,type=MemoryPool)
```

```
4: sun.management.RuntimeImpl
        (java.lang:type=Runtime)

5: sun.management.ClassLoadingImpl
        (java.lang:type=ClassLoading)

6: sun.management.MemoryPoolImpl
        (java.lang:name=Survivor Space,type=MemoryPool)

7: sun.management.ThreadImpl
        (java.lang:type=Threading)

8: sun.management.GarbageCollectorImpl
        (java.lang:name=MarkSweepCompact,type=GarbageCollector)

9: com.sun.management.UnixOperatingSystem
        (java.lang:type=OperatingSystem)

10: sun.management.MemoryImpl
        (java.lang:type=Memory)

11: sun.management.MemoryPoolImpl
        (java.lang:name=Code Cache,type=MemoryPool)

12: sun.management.MemoryPoolImpl
        (java.lang:name=Tenured Gen,type=MemoryPool)

13: sun.management.MemoryPoolImpl
        (java.lang:name=Perm Gen,type=MemoryPool)

Enter the index of target/MBean you wish to monitor or press <Ctrl-C> to quit: 4

Following metric source types are available for selected target(s):
        0: JMX Attributes

Enter the index of your choice or press <Ctrl-C> to quit: 0

Attributes are:
        0: BootClassPath         Return Value: java.lang.String
        1: BootClassPathSupported        Return Value: boolean
        2: ClassPath     Return Value: java.lang.String
        3: InputArguments        Return Value: [Ljava.lang.String;
        4: LibraryPath   Return Value: java.lang.String
        5: ManagementSpecVersion         Return Value: java.lang.String
        6: Name  Return Value: java.lang.String
        7: SpecName      Return Value: java.lang.String
        8: SpecVendor    Return Value: java.lang.String
        9: SpecVersion   Return Value: java.lang.String
        10: StartTime    Return Value: long
        11: SystemProperties     Return Value:
javax.management.openmbean.TabularData
        12: Uptime       Return Value: long
        13: VmName       Return Value: java.lang.String
        14: VmVendor     Return Value: java.lang.String
        15: VmVersion    Return Value: java.lang.String
Select one or more items as comma-separated indices: 6,7,8

Number of possible columns in the resultant metric are 3.
```

```
Enter the name for this metric column at index=0 : [Name]
Is this column a KEY Column <y/n>? [n] y
Is this column for SUMMARY_UI <y/n>? [n]
Enter the label for column: [Name]
Enter the NLSID for column: [Name]
Enter the UNIT for column "Name": [millisec, kb etc.. ]

Enter the name for this metric column at index=1 : [SpecName]
Is this column a KEY Column <y/n>? [n]
Is this column for SUMMARY_UI <y/n>? [n]
Enter the label for column: [SpecName]
Enter the NLSID for column: [SpecName]
Enter the UNIT for column "SpecName": [millisec, kb etc.. ]
Do you want to create a threshold for this column <y/n>? [n]

Enter the name for this metric column at index=2 : [SpecVendor]
Is this column a KEY Column <y/n>? [n]
Is this column for SUMMARY_UI <y/n>? [n]
Enter the label for column: [SpecVendor]
Enter the NLSID for column: [SpecVendor]
Enter the UNIT for column "SpecVendor": [millisec, kb etc.. ]
Do you want to create a threshold for this column <y/n>? [n]

Enter the name of this metric: RuntimeMetric
Enter the label for this metric: [RuntimeMetric]

Do you want periodic collection for this metric <y/n>? [n] y
Enter the collection interval in seconds: 300
Periodic collection interval is: 300 seconds.

Do you want to create another metric <y/n>? [n]
Written the metadata xml file: ./metadata/myJavaApp.xml.
Creating new file: ./default_collection/myJavaApp.xml.
Updated the default collection file for myJavaApp at location
./default_collection/myJavaApp.xml.
Exiting...
```

7.4.2 Using the Metadata and Default Collection Files

Look at the `<currentDir>/metadata` and `currentDir/default_collection` directories to see the `myTarget.xml` files (for the target type you specified earlier).

You can use these files as follows:

- Convert the files to a Management Plug-in Archive (MPA) and push them from OMS to the Agent and target instances created from OMS. See Section 7.5, "Creating a Management Plug-in Archive" and Section 7.8.3, "Configuring a Standalone Java Application or JVM Target".
- Edit the files, extract the metric definitions and `QueryDescriptors`, move them to other metadata and default collection files, and post-process them by creating `ExecutionDescriptors` as needed.

If you want the status information of your targets to appear correctly in the Enterprise Manager console, you need to define a Response metric. See Section 7.3.2, "Displaying Target Status Information" for more information.

7.5 Creating a Management Plug-in Archive

> **Note:** The Management Plug-in creation process applies to the Web Service, JMX-instrumented, and standalone JVM target types. Section 7.4 through Section 7.7 use Web Services as an illustrative example.

Using the Enterprise Manager Command-Line Interface (EM CLI), you create a Management Plug-in Archive, as shown in the following example.

```
>./emcli add_mp_to_mpa -mpa=TargetWithWSMgmtInterface.jar -mp_version="1.0"
-ttd="{META_FILE} -dc="${DC_FILE}"
```

`META_FILE` and `DC_FILE` represent the target metadata and default collection files that the Web Services Command-Line tool generates. For more information about the EM CLI, see *Oracle Enterprise Manager Command Line Interface*.

After archive jar file is created, you can upload the archive to Enterprise Manager and import the Management Plug-in. For more information about Management Plug-in Archives, see Chapter 2, "Developing a Management Plug-in".

7.6 Importing a Management Plug-in

The next step is to import the Management Plug-in from the archive file. You perform this operation from the Enterprise Manager console.

1. From the Enterprise Manager console, click **Setup**.
2. Click **Management Plug-ins** from the left navigation bar.
3. Click **Import**.
4. From the Import page, specify the Management Plug-in Archive file and click **List Archive**. All Management Plug-ins contained within the archive are displayed.
5. Select the Web service Management Plug-in from the list and click **OK**. Enterprise Manager returns you to the Management Plug-ins page with your Web service-specific plug-in added to the list.

Figure 7–1 Importing the TargetWithWSMgmtInterface Management Plug-in

Graphic shows the Management Plug-in page from the Enterprise Manager console with the petstoreApp Management Plug-in added.

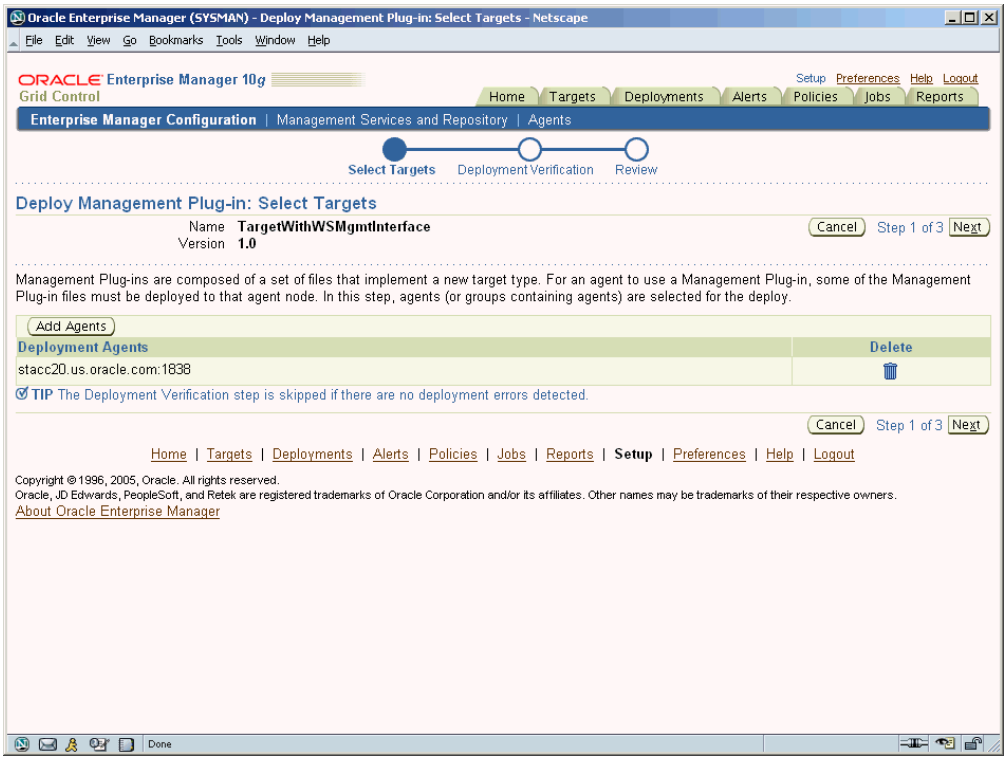
****

7.7 Deploying a Management Plug-in to the Management Agent

After the Management Plug-in has been imported into Enterprise Manager, you can deploy the plug-in to any number of Management Agents. Deploying the Management Plug-in to an Agent allows that Agent to monitor targets exposing a Web Service management interface.

1. From the Management Plug-in main page, click the **Deploy** icon for the Management Plug-in you want to deploy. The Deploy Management Plug-in: Select Targets page appears, as shown in the following figure.

Figure 7–2 Deploying the Management Plug-in to Management Agents



Graphic shows the Management Plug-in being deployed to a Management Agent.

****

2. Click **Add Agents**. The Search and Select: Agents page appears.
3. Choose the desired Management Agent. Use the appropriate search parameters to find the desired Agent.
4. Click **Select**. Enterprise Manager returns you to the Deploy Management Plug-in: Select Targets page. The selected Agent(s) appears in the deployment list.

   If deployment issues are detected for one or more the selected Agents, the Deployment Verification page appears. Only Agents for which an issue is detected appear in the table on this page. See online help for more information.
5. Click **Next** to go to the Review page. The Management Plug-in name and version are shown. This page contains important information pertaining to the deployment process. Ensure that you understand the information before proceeding.
6. Click **Finish**.

7.8 Adding a Target Instance

Now that the Management Plug-in has been deployed to the appropriate Agent(s), you are ready to begin monitoring target types defined by your Management Plug-in. Because of the different target type properties, separate procedures for adding all target types are covered.

7.8.1 Adding a Web Services Target Instance

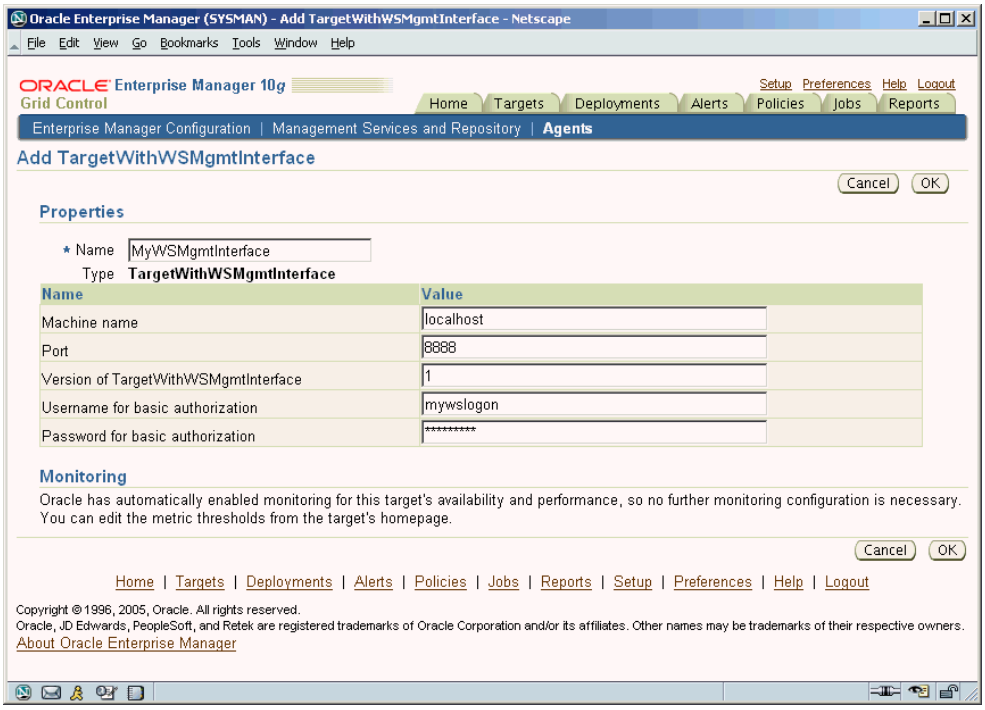
Now that the Management Plug-in has been deployed to the appropriate Agent(s), you can begin monitoring targets that expose a Web Service management interface, as defined by your Management Plug-in.

1. From the Monitored Targets section of the Management Agent home page, choose the desired target type from the **Add** drop-down menu and click **Go**. The Add target page appears.

   Note that **Username/Password for basic authorization** are the login credentials used to access the Web Service.

2. Enter the requisite target properties and click **OK**, as shown in Figure 7–3. The newly added target appears in the Agent's Monitored Targets list.

Figure 7–3 Adding a Target Instance



Graphic illustrates adding the myPetstoreApp instance to Enterprise Manager.

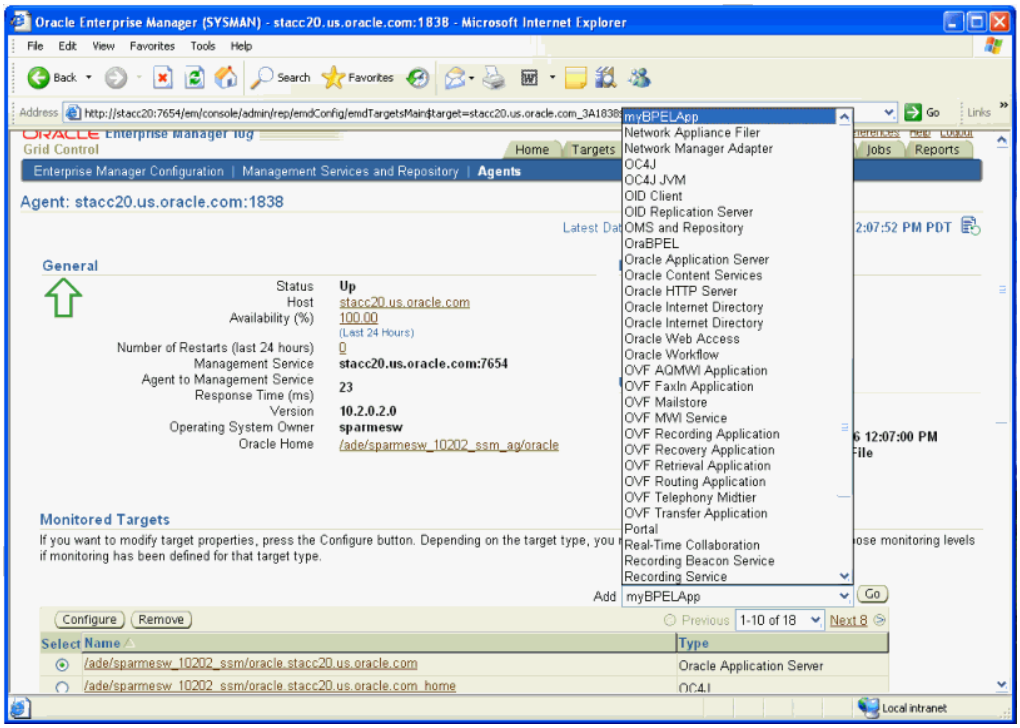
****

7.8.2 Adding a JMX-instrumented J2EE Target Instance

If you deployed a Management Plug-in that defines a JMX-instrumented target type, you can begin configuring your JMX-enabled J2EE application targets so that metrics for these targets can be collected in Enterprise Manager Grid Control.

1. From the Monitored Targets section of the Management Agent home page, choose the desired target type from the **Add** drop-down menu and click **Go**. The Add target page appears.

Figure 7–4 Selecting a JMX Target Type

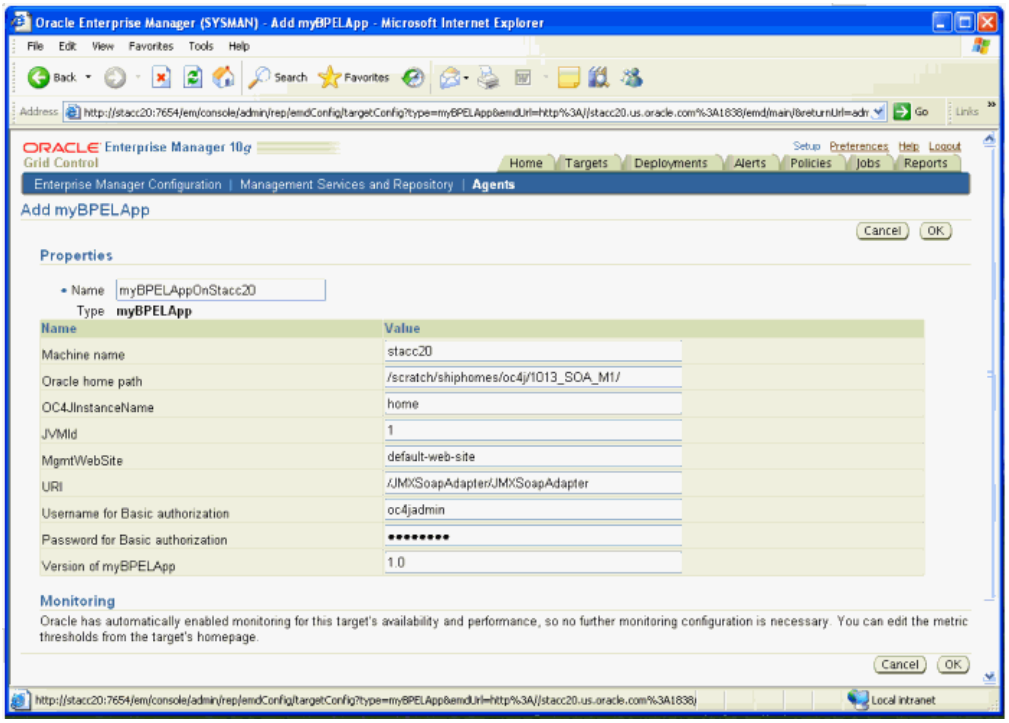


Graphic shows the Management Agent home page with the target type drop-down menu displayed.

****

2. Enter the requisite target properties, as shown in Figure 7–5, then click **OK**. The newly added target appears in the Agent's Monitored Targets list.

Figure 7–5 Specifying JMX Target Properties



Graphic shows the target properties page for a JMX-instrumented target.

****

Table 7–1 provides definitions for the instance properties.

Table 7–1 JMX Target Properties

Property	Definition
Name	Unique name for this target instance.
MachineName	Hostname/IpAddress of the machine running the 10.1.3 version or higher of the Oracle Application Server.
Oracle Home Path	Absolute path to the Oracle_Home of the application server.
OC4JinstanceName	Instance name of the OC4J to which this JMX-enabled j2ee application is deployed. Default is the "home" instance.
JVMId	If this OC4J instance is running multiple JVMs, the index (starts with 1) of the JVM where this metric needs to be collected from (if there are multiple JVMs, a target instance may be added for each JVM instance). **Note:** You can create a "Group" from all these target instances running on multiple JVMs and see rolled-up values across all the JVMs, dashboards, and other targets where aggregate monitoring is advantageous.
MgmtWebSite	The http website on which the JMXSoapAdapter is deployed. This is default-web-site by default.
URI	WebService end-point for the JMXSoapAdapter (by default this is /JMXSoapAdapter/JMXSoapAdapter).

Table 7–1 (Cont.) JMX Target Properties

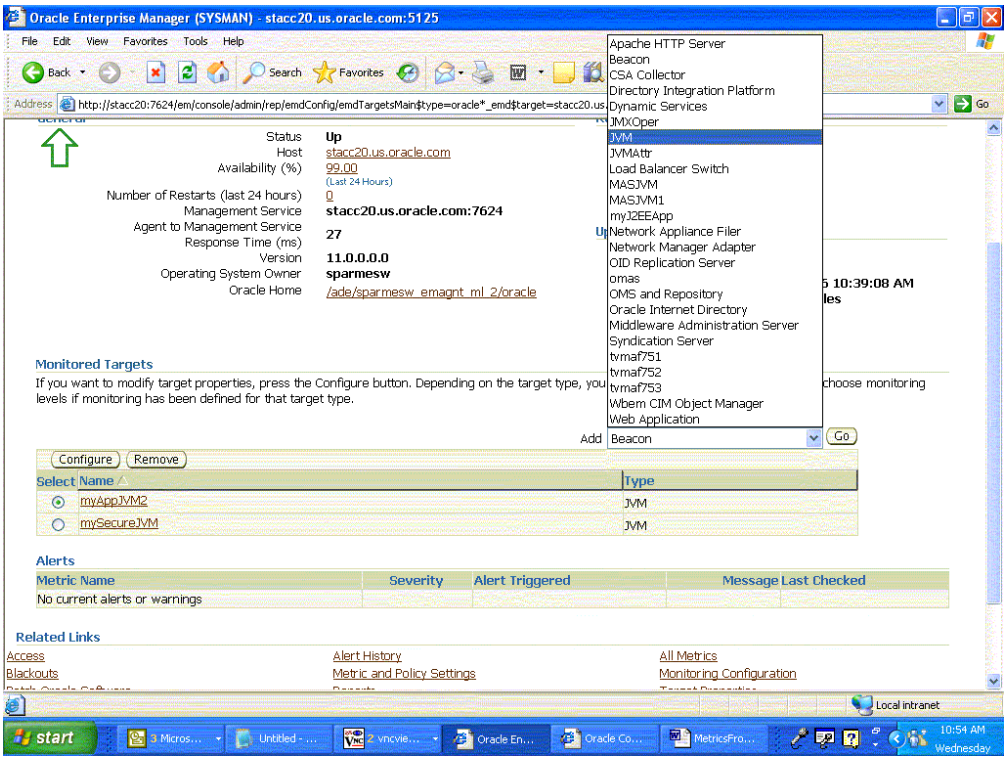
Property	Definition
User Name for Basic Authorization	oc4jadmin user name.
Password for Basic Authorization	Password for the above user.
Version of TargetType	Default to the value from the metadata file created with `emjmxcli`.

7.8.3 Configuring a Standalone Java Application or JVM Target

If you deployed a Management Plug-in that defines your standalone Java application or you want to use the built-in JVM target type, you can begin configuring your JVM or JMX-enabled Java application targets so that metrics for these targets can be collected in Enterprise Manager Grid Control.

1. On the system running the JVM, install an Enterprise Manager Grid Control Agent for version 10.2.0.3 or greater. Although recommended, this is not absolutely necessary for JVM and standalone Java application targets.
2. From the Enterprise Manager console, navigate to the home page of the Enterprise Manager Agent.
3. In the Monitored Targets section, select the JVM target (or your standalone Java application target type for which you deployed the MPA earlier) from the drop-down list as shown in Figure 7–6.

Figure 7–6 Selecting the JVM Target

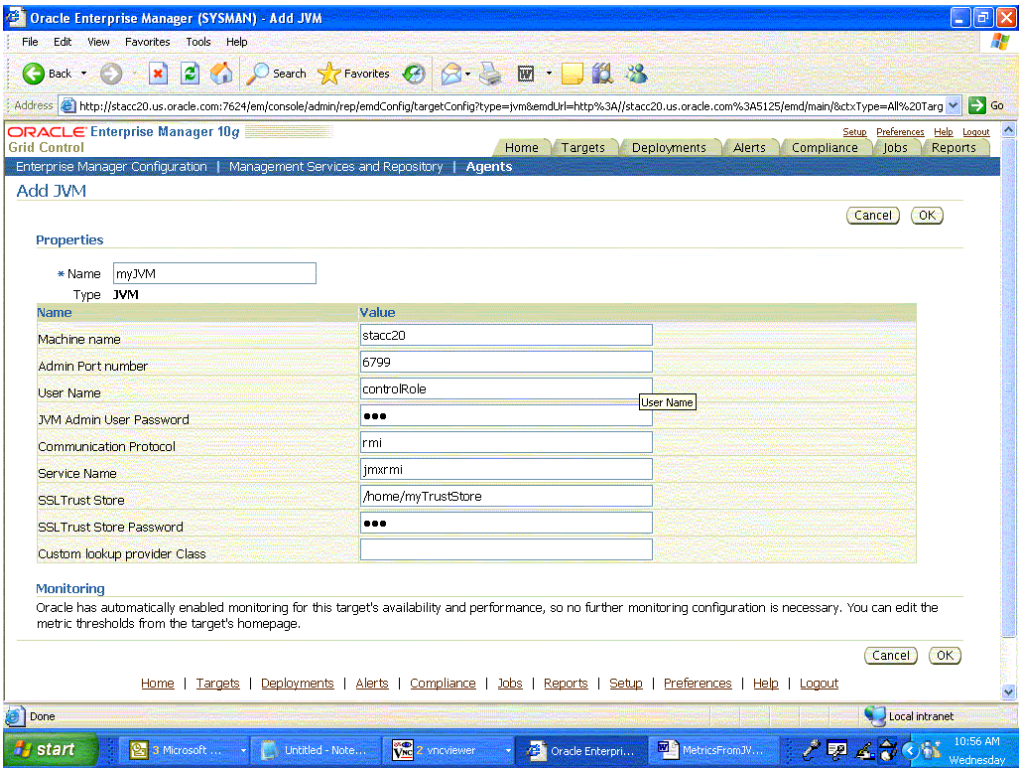


Graphic shows the Management Plug-in page from the Enterprise Manager console with the petstoreApp Management Plug-in added.

****

4. Enter the instance properties for this JVM or Java application instance that the Enterprise Manager Agent needs to monitor, as shown in Figure 7–7, then click **OK**.

Figure 7–7 Specifying JVM Instance Properties



Graphic shows the Management Plug-in page from the Enterprise Manager console with the petstoreApp Management Plug-in added.

****

Table 7–2 provides definitions for the instance properties.

Table 7–2 JVM Instance Properties

Property	Definition
Name	Target name for this JVM instance.
MachineName	Host name where this JVM is running.
Admin Port Number	Port number a JSR-160 client can use (such as jconsole when using the "remote" option) to connect to the JVM. (This is the port specified for the `-Dcom.sun.management.jmxremote.port` property when the JVM is started up to enable remote management.)
User Name	Required if JVM started with: `Dcom.sun.management.jmxremote.authenticate=true` with a password and access file.
JVM Admin User Password	See User Name above.

Table 7–2 (Cont.) JVM Instance Properties

Property	Definition
Communication Protocol	Establishes a connection to the MBeanServer on the target JVM. This corresponds to the properties of the JMX ServiceURL needed to establish the JMX connection to the target MBeanServer. The default of rmi should be retained.
Service Name	Establishes a connection to the MBeanServer on the target JVM. This corresponds to the properties of the JMX ServiceURL needed to establish the JMX connection to the target MBeanServer. The default of jmxrmi should be kept.
SSL Trust Store	Location of the SSL Trust Store, which is needed if the target JVM has SSL enabled with `-Dcom.sun.management.jmxremote.ssl=true` on its startup.
SSL Trust Store Password	Password needed to access the SSL Trust Store path.
Custom Lookup Provider Class	Full package name of a user-implemented Java lookup class that can be integrated into the Enterprise Manager client and be used to perform a custom lookup of the MBeanServer through LDAP or other lookup protocols. For information about providing a custom lookup service, see Section 7.8.4.

5. Navigate to the All Metrics page of the added JVM (Java application) target to see the metrics collected from the JVM (Java application) to Enterprise Manager Grid Control, as shown in Figure 7–8. These metrics are exposed by the platform MBeans, which is available on JDK1.5 or above, or from application-defined MBeans for your Java application.

Figure 7–8 Metrics Collected from the JVM

Graphic shows the Management Plug-in page from the Enterprise Manager console with the petstoreApp Management Plug-in added.

****

Figure 7–9 shows details (columns) for these metrics.

Figure 7–9 Details of Collected Metrics

Graphic shows the Management Plug-in page from the Enterprise Manager console with the petstoreApp Management Plug-in added.

****

7.8.4 Using a Custom Look-up Service to Obtain MBean Server Details

In some situations, you may not be able to statically provide the host:port information of an MBeanServer/JVM when you configure a JVM or Java application target in Enterprise Manager. Instead, you could use the machine name and port properties of the target instance as information for a look-up service, and apply your implementation of a custom look-up class to return the appropriate MBeanServer properties to the EM JMX fetchlet, which would then proceed to collect the metric.

The look-up service can override the value of any configuration parameter for a particular target instance. For example, assume you specify `port=8000` (lookup service's port number) and a custom look-up provider class as `com.mycompany.lookup` when adding the JVM/Java application target in the `targets.xml` file in Enterprise Manager Grid Control. The `resolveProperties` method of your custom lookup class (`com.mycompany.lookup`) is called with all the properties defined when your target was added in Enterprise Manager Grid Control (that is, with the value of `port` as `8000:petstore`). Your code can now parse the incoming port property and connect to port 8000 to look up `petstore` and decide that the MBeanServer of interest is on port 9000, and can thereby return port=9000 (actual port needed to connect to the

MBeanServer). The Enterprise Manager Agent JMX fetchlet will now use port 9000 as the port number for the JMX connection.

7.8.4.1 Writing the Java Class

You can use the look-up service by writing a Java class that implements the following interface:

`oracle.sysman.emd.fetchlets.JMX.generic.JMXLookUp`

This interface contains one method:

```
public Properties resolveProperties(Properties p)
```

This method should connect to the look-up service and return the values for configuration parameters the fetchlet requires as a java.util.Properties object with each parameter name and its corresponding value as a key-value pair. The input to this method is also a java.util.Properties object with the properties MachineName, Port, Username, and Password passed on by default to the resolveProperties method of the custom look-up implementation object.

7.8.4.2 Passing Additional Information

If you need to pass additional information to the look-up service, you could, for example, append the information in one of the above properties (MachineName, Port, Username, and Password) and custom-process these in their look-up implementation class. However, when the method in the look-up class returns the Properties object, the method needs to replace the above properties with the corresponding properties of the target MBean Server it looked up.

For example, If you need to look up an MBeanServer "petstore1" on a look-up service on host myHost1, you could define the MachineName as myHost1:petstore1 and handle the custom processing of the MachineName property in your custom processing code. The look-up code would have to pass the correct value of '`myHost1`' as the value of `MachineName` in the Properties object that the `resolveProperties` function returns.

Alternatively, you could manually create additional properties in the `QueryDescriptor` of each metric definition in the target metadata file. By specifying look-up properties by means of a special property called `lookupProperties`, you can provide a comma-separated list of additional properties to be passed to the look-up class. These additional properties can either be instance properties for the new target type (`SCOPE=INSTANCE`) or static values (`SCOPE=GLOBAL`).

Example 7–11 shows a portion of a `QueryDescriptor` with additional static look-up properties (Property NAME="lookupProperties").

Example 7–11 QueryDescriptor Example for Manually Creating Additional Properties

```
  <QueryDescriptor FETCHLET_ID="JMX">
         <Property NAME="MachineName" SCOPE="INSTANCE">MachineName</Property>
         <Property NAME="Port" SCOPE="INSTANCE">Port</Property>
         <Property NAME="UserName" SCOPE="INSTANCE"
OPTIONAL="TRUE">UserName</Property>
         <Property NAME="password" SCOPE="INSTANCE"
OPTIONAL="TRUE">password</Property>
         <Property NAME="protocol" SCOPE="INSTANCE"
OPTIONAL="TRUE">protocol</Property>
         <Property NAME="service" SCOPE="INSTANCE"
OPTIONAL="TRUE">service</Property>
         <Property NAME="SSLTrustStore" SCOPE="INSTANCE"
```

```
OPTIONAL="TRUE">SSLTrustStore</Property>
         <Property NAME="SSLTrustStorePassword" SCOPE="INSTANCE"
OPTIONAL="TRUE">SSLTrustStorePassword</Property>
         <Property NAME="lookupClass" SCOPE="INSTANCE"
OPTIONAL="TRUE">com.mycompany.JMXLookup</Property>
         <Property NAME="lookupP1" SCOPE="GLOBAL" OPTIONAL="TRUE">false</Property>
         <Property NAME="lookupP2" SCOPE="GLOBAL"
OPTIONAL="TRUE">myMBeanServerName</Property>
<Property NAME="lookupProperties" SCOPE="INSTANCE" OPTIONAL="TRUE">lookupP1,
lookupP2</Property>
         <Property NAME="nestedView" SCOPE="GLOBAL"
OPTIONAL="TRUE">false</Property>
         <Property NAME="metric" SCOPE="GLOBAL">java.lang:type=Runtime</Property>
         <Property NAME="identityCol" SCOPE="GLOBAL"
OPTIONAL="TRUE">none</Property>
         <Property NAME="columnOrder"
SCOPE="GLOBAL">VmName;VmVendor;VmVersion;Uptime</Property>
      </QueryDescriptor>
```

This example indicates that two additional properties, `lookupP1` and `lookupP2`, are passed to the look-up class `com.mycompany.JMXLookup` as input to the method `resolveProperties`. The function `resolveProperties` can perform its look-up processing, but needs to return appropriate values for the properties MachineName, Port, Service, and Protocol. This enables the JMX fetchlet to establish the JSR-160 (remote) JMX connection to the MBeanServer.

Example 7–12 performs the look-up and gets values for the configuration parameters:

Example 7–12 Look-up Class Code Example

```
public class MylookupImplementationClass implements JMXLookUp{
public java.util.Properties resolveProperties(java.util.Properties input){
/*  your custom code here that does the lookup and gets values for the
configuration parameters. The return Properties object should have the keys
corresponding to the configuration parameters shown in Table 7-3. If any of the
keys are missing in the return object, it defaults to whatever is available in the
metadata file */
}
}
```

Table 7–3 shows the properties the fetchlet uses in making the JMX connection that the lookup implementation class could override (by returning these properties in the Properties object returned in the resolveProperties call in the example above). These refer to properties overridden by the Lookup class.

Table 7–3 Properties the Fetchlet Uses

Property	Default	Description
MachineName	localhost	MBean server host machine name.
Port	8888	Port on which the MBean server is listening for connections.
Username	null	User name if required for a connection.
Password	null	Password if required for a connection.
Protocol	rmi	Protocol used for the connection.
Service	jmxrmi	Service used for the connection.

Table 7–3 (Cont.) Properties the Fetchlet Uses

Property	Default	Description
SSLTrustStore	null	Path to the SSLTrustStore.
SSLTrustStorePassword	null	Password needed to access the SSLTrustStore path.

7.8.4.3 Compiling the Look-up Class

To compile the `JMXLookUp` implementation class, put `$ORACLE_HOME/lib/emagentRT.jar` in the classpath when compiling your lookup class. The `JMXLookUp` implementation class/jar must eventually be available in the Enterprise Manager Agent's classpath so that it can be used at run time.

To achieve this, do the following after you build the custom look-up class:

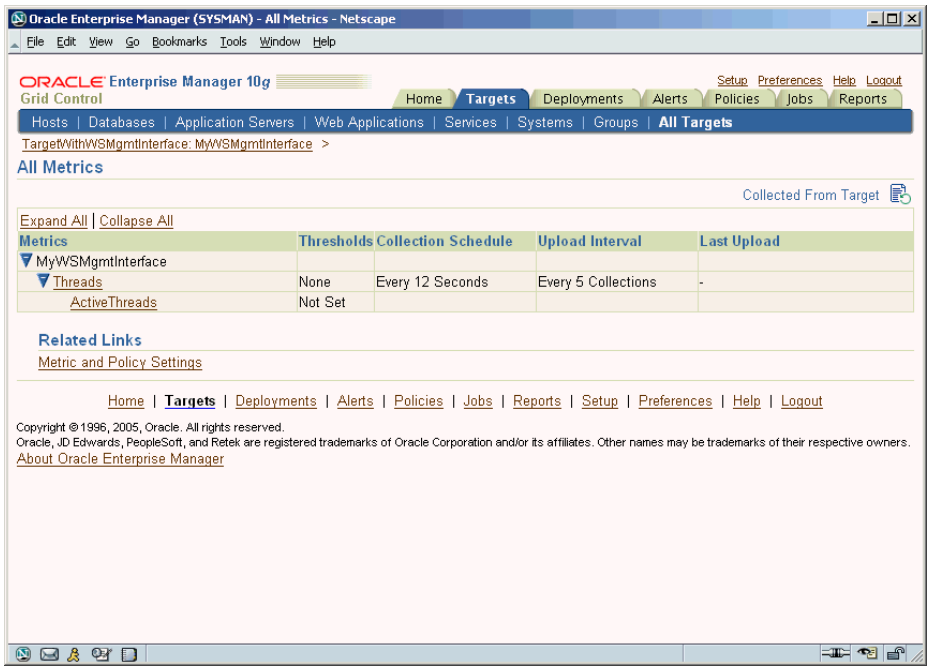
1. Put the custom look-up class in a jar file.
2. Put the location of this jar file in `emagent/sysman/config/classpath.list` or `/emagent/sysman/config/emd.properties` under the `CLASSPATH` section.
3. Restart the Enterprise Manager Agent.

7.9 Viewing Monitored Metrics

With a target instance added to the Agent for monitoring, you can now view metrics defined for your target type.

1. From the Agent home page, click on the target instance you added in the previous step in the **Monitored Targets** list. Enterprise Manager takes you to that target's home page.
2. In the **Related Links** section, click **All Metrics**. The All Metrics page appears for the monitored target. An expandable tree list for each metric enables you to drill down to view specific metric parameters, as shown in Figure 7–10.

Figure 7–10 Viewing Metrics



Graphic shows metrics collected for the myPetstoreApp target instance.

****

8

Defining Policies

Though Enterprise Manager provides a number of out-of-box policies, you may need to define additional policies to meet your monitoring needs. Enterprise Manager provides this capability in the form of the MGMT_USER_DEFINED_POLICY package.

Using the MGMT_USER_DEFINED_POLICY package, you can create and delete policies, as well as add policies to, and remove policies from, targets.

See Also:

- *Oracle Database Concepts* for information about packages, SQL, and PL/SQL
- *Oracle Enterprise Manager Concepts* for information on policies

This chapter includes the following:

Note: This chapter assumes that you are familiar with PL/SQL packages and Enterprise Manager policies.

8.1 Overview

A policy defines the desired behavior of systems and is associated with one or more targets or groups. A policy rule is a conditional expression that test values from a target against a condition, for example, verifying that database profile limits are set as expected.

A policy tests data retrieved from a query performed against the Oracle Management Repository. A policy is said to be compliant if it is determined that the managed targets do, in fact, meet the desired state; that is, the test of the policy failed to identify any violations. Otherwise, a policy is said to be non-compliant when it has one or more policy violations.

As an extension of the out-of-box policies, Enterprise Manager provides the MGMT_USER_DEFINED_POLICY package which supports user-defined policies in the context of subprograms.

To use these subprograms, use SQL*Plus to connect to the Management Repository database as the Management Repository owner. The default Management Repository owner is SYSMAN.

8.1.1 MGMT_USER_DEFINED_POLICY Package

The following subprograms make up the MGMT_USER_DEFINED_POLICY package:

Subprogram	Description
CREATE_POLICY	Creates a user-defined policy.
DELETE_POLICY	Deletes a user-defined policy.
ADD_POLICY_TO_TARGET	Adds a policy to an existing target.
REMOVE_POLICY_FROM_TARGET	Removes a policy from an existing target.

Descriptions of the constants and data types used in the subprograms follow.

8.1.1.1 Constants

The following constants are defined by the MGMT_USER_DEFINED_POLICY package and can be used when calling the supplied subprograms. See Table 8–1.

Table 8–1 Constants Used in the MGMT_USER_DEFINED_POLICY Package

Constant	Description
Categories	*Different types of policies*
G_CATEGORY_FAULT	Breakdown in a component or occurrence of an error that indicates some component or user is unable to successfully complete processing, for example, database down
G_CATEGORY_WORKLOAD_VOL	Workload on a system induced in proportion to the users or batch jobs running against the system, for example, number of user calls
G_CATEGORY_WORKLOAD_TYPE	Type of workload on a system independent of demand, for example, CPU usage
G_CATEGORY_PERFORMANCE	Performance of a system, for example, response time
G_CATEGORY_CAPACITY	Usage of a fixed resource, for example, tablespace usage
G_CATEGORY_CONFIGURATION	Configuration of a target against recommended "best practice" configurations, for example, insufficient number of control files
G_CATEGORY_SECURITY	Security settings and issues, for example, open ports
G_CATEGORY_STORAGE	Storage, for example, permanent and temporary tablespaces
G_CATEGORY_UNCLASSIFIED	Default category to be used if a policy does not fall into a particular category
Severity Levels for Violation	*Seriousness of violation*
G_SEVERITY_INFORMATIONAL	Provides facts about the violation.

Table 8–1 (Cont.) Constants Used in the MGMT_USER_DEFINED_POLICY Package

Constant	Description
G_SEVERITY_WARNING	Forewarns of serious consequences if the violation is not dealt with in a timely manner.
G_SEVERITY_CRITICAL	Requires immediate attention to the violation.
Parameter Data Types	*Data types for policy-related parameters*
G_PARAM_TYPE_NUMBER	Number
G_PARAM_TYPE_STRING	String
Target Types	*Entities monitored by Enterprise Manager*
G_HOST_TARGET_TYPE	Host
G_DATABASE_TARGET_TYPE	Oracle Database Instance
G_LISTENER_TARGET_TYPE	Oracle Listener
G_CLUSTER_TARGET_TYPE	Group of independent servers
G_RAC_DATABASE_TARGET_TYPE	Real Application Cluster (RAC) Database Instance
G_REDUNDANCY_GROUP_TARGET_TYPE	Group containing members of the same type that function collectively as a unit
G_COMPOSITE_TARGET_TYPE	Number of targets grouped together for a purpose, for example, a business function
G_HOST_GROUP_TARGET_TYPE	Group consisting of many hosts
G_DATABASE_GROUP_TARGET_TYPE	Group consisting of many Oracle database instances
G_IAS_TARGET_TYPE	Oracle Application Server
G_WEBSITE_TARGET_TYPE	Web Application
G_FORMSAPP_TARGET_TYPE	Oracle Forms
G_HTTP_SERVER_TARGET_TYPE	Oracle HTTP Server
G_WEBCACHE_TARGET_TYPE	OracleAS Web Cache
G_OC4J_TARGET_TYPE	Oracle Application Server Containers for J2EE
G_BC4J_TARGET_TYPE	ADF Business Components for Java
G_LDAP_TARGET_TYPE	Lightweight Directory Access Protocol (LDAP)
G_PORTAL_TARGET_TYPE	OracleAS Portal
G_APPLICATION_TARGET_TYPE	Oracle Application
G_APPS_SYSTEM_TARGET_TYPE	Oracle Application System
G_ASM_TARGET_TYPE	Automatic Storage Management
Condition Operators	*Used when manipulating thresholds which are boundary values against which monitored metric values are compared*
G_THRESHOLD_EQ	Equal to (=)
G_THRESHOLD_LT	Less than (<)
G_THRESHOLD_GT	Greater than (>)
G_THRESHOLD_LE	Less than or equal to (< =)
G_THRESHOLD_GE	Greater than or equal to (> =)

Table 8–1 (Cont.) Constants Used in the MGMT_USER_DEFINED_POLICY Package

Constant	Description
G_THRESHOLD_NE	Not equal to (≠)
G_THRESHOLD_CONTAINS	Contains at least
G_THRESHOLD_MATCH	Exact match

8.1.1.2 Data Types

The MGMT_USER_DEFINED_POLICY subprograms use the data types described in Table 8–2.

Table 8–2 Data Types Used in the MGMT_USER_DEFINED_POLICY Package

Type	Description
UDP_PARAMETERS	Represents a collection of parameters used in the user-defined policy (UDP).
UDP_PARAMETER	Represents a single parameter.

When creating a user-defined policy, parameter information is passed using the UDP_PARAMETERS object.

8.1.1.2.1 UDP_PARAMETERS and UDP_PARAMETER Types Use these PL/SQL types to represent the list of parameters used by a user-defined policy.

Syntax

```
TYPE UDP_PARAMETER IS RECORD
(
     param_name          VARCHAR2(64),
     param_type          NUMBER(1),
     threshold_value     VARCHARS(4000)
);

TYPE UDP_PARAMETERS IS TABLE of UDP_PARAMETER;
```

Parameters

Parameter	Description
param_name	Name of the parameter to be created
param_type	Type of parameter Example: G_PARAM_TYPE_NUMBER G_PARAM_TYPE_STRING
threshold_value	Default value of parameter

8.2 Creating a User-Defined Policy

As part of creating a user-defined policy, you specify the SQL to use to extract the information to be tested from the Management Repository. Enterprise Manager provides management views with which you can safely extract data from the Management Repository without reading from the base tables.

By using Management Repository views, you:

- Protect your queries from changes to the Management Repository schema that may occur in future releases
- Ensure that your policy definition remains functional

A complete listing of the Management Repository views can be found in Chapter 9, "Management Repository Views".

CREATE_POLICY Procedure

The CREATE_POLICY procedure creates a user-defined policy. As part of a policy, one must identify the:

- Name of the policy
- Data that is to be evaluated
- Test, optionally parameterized, that is used to test the current state of the data
- What, if any, default parameter values should be substituted into the test

Once the script is run, the policy is automatically stored in the Policy Library and is available for viewing.

Tests Supported by CREATE_POLICY Procedure

The following types of tests are supported in the CREATE_POLICY procedure:

- Threshold condition or simple condition

  The value from a single selected column is tested using a specified test operator

- SQL expression

  A SQL expression that can contain references to literals, selected columns and policy parameters, where the latter two are named as bind variables. In addition, any built-in SQL functions can be used. For example, the following uses the SQL function (length), as well as the selected column (PASSWORD), and policy parameter (MIN_PWD_LENGTH):

  ```
  length(:PASSWORD) <= :MIN_PWD_LENGTH
  ```

The following syntax describes the procedure used to create a policy. If the policy is not created, an error message is raised using the RAISE_APPLICATION_ERROR procedure. The error number and message can be trapped like any Oracle error for processing.

Syntax

Threshold or simple condition

```
PROCEDURE create_policy
(
    p_policy_name           IN VARCHAR2,
    p_target_type           IN VARCHAR2,
    p_sql_text              IN VARCHAR2,
    p_column_name           IN VARCHAR2,
    p_test_operator         IN VARCHAR2,
    p_threshold_value       IN VARCHAR2,
    p_threshold_data_type   IN NUMBER       DEFAULT G_PARAM_TYPE_NUMBER,
    p_num_keys              IN NUMBER       DEFAULT 1,
    p_description           IN VARCHAR2     DEFAULT ' ',
    p_impact                IN VARCHAR2     DEFAULT ' ',
    p_recommendation        IN VARCHAR2     DEFAULT ' ',
    p_severity_level        IN NUMBER       DEFAULT G_SEVERITY_INFORMATIONAL,
```

```
    p_category              IN VARCHAR2     DEFAULT G_CATEGORY_UNCLASSIFIED,
    p_url_link              IN VARCHAR2     DEFAULT NULL,
    p_violation_message     IN VARCHAR2     DEFAULT NULL,
    p_clear_message         IN VARCHAR2     DEFAULT NULL,
    p_eval_interval         IN NUMBER       DEFAULT 24
);
```

SQL expression

```
PROCEDURE create_policy
(
    p_policy_name           IN VARCHAR2,
    p_target_type           IN VARCHAR2,
    p_sql_text              IN VARCHAR2,
    p_test                  IN VARCHAR2,
    p_parameters            IN UDP_PARAMETERS  DEFAULT NULL,
    p_num_keys              IN NUMBER          DEFAULT 1,
    p_description           IN VARCHAR2        DEFAULT ' ',
    p_impact                IN VARCHAR2        DEFAULT ' ',
    p_recommendation        IN VARCHAR2        DEFAULT ' ',
    p_severity_level        IN NUMBER          DEFAULT G_SEVERITY_INFORMATIONAL,
    p_category              IN VARCHAR2        DEFAULT G_CATEGORY_UNCLASSIFIED,
    p_url_link              IN VARCHAR2        DEFAULT NULL,
    p_violation_message     IN VARCHAR2        DEFAULT NULL,
    p_clear_message         IN VARCHAR2        DEFAULT NULL,
    p_eval_interval         IN NUMBER          DEFAULT 24
);
```

Parameters

Parameter	Description
p_policy_name	Name of the policy to be created
p_target_type	Type of target to which this policy is applicable Example: G_HOST_TARGET_TYPE (Host) G_DATABASE_TARGET_TYPE (Oracle Database Instance)
p_sql_text	SQL used to retrieve data to be tested from the Management Repository The specified SQL needs to satisfy the following requirements: ■ The first *n* selected columns, where *n* equals p_num_keys, are columns that are used to uniquely identify a row. The SQL query must not return more than one row with the same key. Furthermore, one of these key columns must be a column called TARGET_GUID which contains the target guid of the target and to which the policy violation should be associated. ■ The remaining selected columns, of which there must be at least one, return the values that are then tested by the policy's test.
p_column_name	Name of the column from the select list against which to compare the p_threshold_value

Parameter	Description
p_test_operator	Type of comparison to be performed. Valid values are: G_THRESHOLD_EQ G_THRESHOLD_NE G_THRESHOLD_LT G_THRESHOLD_LE G_THRESHOLD_GT G_THRESHOLD_GE G_THRESHOLD_CONTAINS G_THRESHOLD_MATCH
p_threshold_value	Value against which comparison is performed
p_threshold_data_type	Data type of the threshold value. Valid values are: G_PARAM_TYPE_NUMBER G_PARAM_TYPE_STRING
p_test	Test to apply to the rows returned by p_sql_text for identifying violations of the policy. The test can be any valid SQL expression. It can also reference columns in the select list from p_sql_text and/or parameters specified in p_parameters. To reference columns in the select column list or a parameter, prefix the name with a colon (:).
p_parameters	Tuples containing a list of parameters and the default value to be used when evaluating the policy
p_num_keys	Number of columns in the select list that are key columns; that is, they uniquely identify a row returned by p_sql_text
p_description	Contains descriptive text for the policy.
p_impact	Provides text that states why this policy is important.
p_recommendation	Contains information regarding how to bring a target back into compliance with the policy.
p_severity_level	Severity level for a violation. Valid values include: G_SEVERITY_INFORMATIONAL G_SEVERITY_WARNING G_SEVERITY_CRITICAL
p_category	Policy category. Valid values are: G_CATEGORY_FAULT G_CATEGORY_WORKLOAD_VOL G_CATEGORY_WORKLOAD_TYPE G_CATEGORY_PERFORMANCE G_CATEGORY_CAPACITY G_CATEGORY_CONFIGURATION G_CATEGORY_SECURITY G_CATEGORY_STORAGE G_CATEGORY_UNCLASSIFIED
p_url_link	URL to be used for additional detailed information regarding this policy
p_violation_message	Message recorded along with the violation. Used for notifications, such as e-mails and paging, that happen as a result of a detection of a new violation. The message can reference columns in the select list from p_sql_text and/or parameters specified in p_parameters. To reference columns in the select column list or a parameter, enclose the name with percent signs (%).

Parameter	Description
p_clear_message	Message recorded with the clearing of a violation. Used for notifications, such as e-mails and paging, that happen as a result of a detection of a new violation. The message can reference columns in the select list from p_sql_text and/or parameters specified in p_ parameters. To reference columns in the select column list or a parameter, enclose the name with percent signs (%).
p_eval_interval	Evaluation interval expression in number of hours

Examples

The following examples depict how to create a policy.

Example 8–1 Sufficient Control Files Threshold

```
DECLARE
  l_sql         VARCHAR2(2000);
BEGIN
  l_sql := 'SELECT target_guid, control_file_count ' ||
              'FROM ' ||
                '(SELECT target_guid, COUNT(file_name) control_file_count ' ||
                'FROM mgmt$db_controlfiles ' ||
                'GROUP BY target_guid)';
  MGMT_USER_DEFINED_POLICY.CREATE_POLICY
  (
    p_policy_name          => 'Insufficient Control Files',
    p_target_type          => mgmt_user_defined_policy.G_DATABASE_TARGET_TYPE,
    p_sql_text             => l_sql,
    p_column_name          => 'control_file_count',
    p_test_operator        => mgmt_user_defined_policy.G_THRESHOLD_LT,
    p_threshold_value      => 2,
    p_threshold_data_type  => mgmt_user_defined_policy.G_PARAM_TYPE_NUMBER,
    p_num_keys             => 1,       -- target_guid is key column
    p_description          => 'Ensures sufficient control files',
    p_impact         => 'The control file is one of the most important files '||
                        'in an Oracle database. It maintains many physical ' ||
                        'characteristics and important recovery information ' ||
                        'about the database. If you lose the only copy of the ' ||
                        'control file due to a media error, there will be ' ||
                        'unnecessary down time and other risks.'
    p_recommendation       => 'Use at least two control files that are ' ||
                              'multiplexed on different disks.',
    p_severity_level       => mgmt_user_defined_policy.G_SEVERITY_CRITICAL,
    p_category             => mgmt_user_defined_policy.G_CATEGORY_CONFIGURATION,
    p_violation_message    => 'Insufficient control files: %control_file_count%',
    p_clear_message        => 'Sufficient control files'
 );
COMMIT;
END;
/
```

Example 8–2 Sufficient Control Files Using UDP Parameters

```
DECLARE
  l_sql         VARCHAR2(2000);
  l_test        VARCHAR2(2000);
  l_parameters  mgmt_user_defined_policy.udp_parameters;
BEGIN
  l_sql := 'SELECT target_guid, control_file_count ' ||
```

```
          'FROM ' ||
            '(SELECT target_guid, COUNT(file_name) control_file_count ' ||
            'FROM mgmt$db_controlfiles ' ||
            'GROUP BY target_guid)';

  l_test := ':control_file_count < :min_control_file_count';

  l_parameters := mgmt_user_defined_policy.udp_parameters();
  l_parameters.extend(1);
  l_parameters(1).param_name := 'min_control_file_count';
  l_parameters(1).param_type := mgmt_user_defined_policy.G_PARAM_TYPE_NUMBER;
  l_parameters(1).threshold_value :=2;

  MGMT_USER_DEFINED_POLICY.CREATE_POLICY
  (
    p_policy_name    => 'Insufficient Control Files',
    p_target_type    => mgmt_user_defined_policy.G_DATABASE_TARGET_TYPE,
    p_sql_text       => l_sql,
    p_test           => l_test,
    p_parameters     => l_parameters,
    p_num_keys       => 1,   -- target_guid is key column
    p_description    => 'Ensures sufficient control files',
    p_impact         => 'The control file is one of the most important files ' ||
                        'in an Oracle database. It maintains many physical ' ||
                        'characteristics and important recovery information ' ||
                        'about the database. If you lose the only copy of the ' ||
                        'control file due to a media error, there will be ' ||
                        'unnecessary down time and other risks.',
    p_recommendation      => 'Use at least two control files that are ' ||
                             'multiplexed on different disks.',
    p_severity_level      => mgmt_user_defined_policy.G_SEVERITY_CRITICAL,
    p_category            => mgmt_user_defined_policy.G_CATEGORY_CONFIGURATION,
    p_violation_message   => 'Insufficient control files: %control_file_count%',
    p_clear_message       => 'Sufficient control files'
  );
  COMMIT;
END;
/
```

Example 8–3 Data Dictionary Protected SQL Expression

```
DECLARE
  l_sql   VARCHAR2(2000);
  l_test  VARCHAR2(2000);
BEGIN
  l_sql := 'SELECT target_guid, name, value ' ||
              'FROM mgmt$db_init_params ' ||
              'WHERE name =''07_DICTIONARY_ACCESSIBILITY''';
  l_test := 'UPPER(NVL(:value, ''TRUE'')) = ''TRUE''';
 MGMT_USER_DEFINED_POLICY.CREATE_POLICY
 (
    p_policy_name      => 'Data Dictionary Protected',
    p_target_type      => mgmt_user_defined_policy.G_DATABASE_TARGET_TYPE,
    p_sql_text         => l_sql,
    p_test             => l_test,
    p_num_keys         => 1,      -- target_guid is key column
    p_description      => 'Ensures data dictionary protection is enabled',
    p_impact           => 'Setting the 07_DICTIONARY_ACCESSIBILITY to TRUE ' ||
                          'allows users with ANY system privileges to access ' ||
                          'the data dictionary.  As a result, these user ' ||
```

```
                            'accounts can be exploited to gain unauthorized ' ||
                            'access to data. Instead the data dictionary should ' ||
                            'be protected such that only those authorized users ' ||
                            'making DBA-privileged connections can use the ANY ' ||
                            'system privilege to access the data dictionary.',
    p_recommendation        => 'Set O7_DICTIONARY_ACCESSIBILITY to TRUE',
    p_severity_level        => mgmt_user_defined_policy.G_SEVERITY_CRITICAL,
    p_category              => mgmt_user_defined_policy.G_CATEGORY_SECURITY,
    p_violation_message     => 'Data dictionary is not protected',
    p_clear_message         => 'Data dictionary is protected',
    p_eval_interval         => 12
 );
 COMMIT;
END;
/
```

8.3 Adding a User-Defined Policy to Existing Targets

There are a number of ways to add a user-defined policy to existing targets. These methods include:

- Using the Metric and Policy Settings UI
- Using a PL/SQL Procedure

8.3.1 Using the Metric and Policy Settings UI

To quickly add a user-defined policy to existing targets, use the Metric and Policy Settings UI pages.

1. On the Grid Control home page, click the **Targets** tab.
2. On the resulting page, click the target type of the target to which you want to add the user-defined policy.
3. Select the name of the target to which you want to add the user-defined policy.
4. On the resulting Home page for the target, scroll to the Related Links section and click **Metric and Policy Settings**.
5. Choose the **Policies** subtab and click **Add Policies**.
6. Select the policy rule from the Policy Library you want to add and click **Continue**.
7. On the confirmation page, click **Yes** to add the policy rule to the target. In addition, ensure that you click **OK** on the Policies page, otherwise the change will not take affect.

8.3.2 Using a PL/SQL Procedure

The following procedure associates an existing user-defined policy with a target that already exists in the Management Repository. If the association is not created, an error message is raised using the RAISE_APPLICATION_ERROR procedure. The error number and message can be trapped like any Oracle error for processing.

Syntax

```
PROCEDURE add_policy_to_target
(
    p_policy_name       IN VARCHAR2,
    p_target_type       IN VARCHAR2,
    p_target_name       IN VARCHAR2
```

```
);
```

Parameters

Parameter	Description
p_policy_name	Name of the policy to be added
p_target_type	Type of target to which this policy is applicable Examples: G_HOST_TARGET_TYPE (Host) G_DATABASE_TARGET_TYPE (Oracle Database Instance)
p_target_name	Name of the target

Examples

The following examples depict how to add policies to targets.

Example 8–4 Add a Policy to One Target

```
BEGIN
  MGMT_USER_DEFINED_POLICY.ADD_POLICY_TO_TARGET
  (
    p_policy_name     => 'Insufficient Control Files',
    p_target_type     => mgmt_user_defined_policy.G_DATABASE_TARGET_TYPE,
    p_target_name     => 'Finance'
  );
COMMIT;
END;
/
```

Example 8–5 Add a Policy to All Existing Targets

```
DECLARE
  l_target_names   MGMT_MEDIUM_STRING_ARRAY;
BEGIN
 SELECT target_name BULK COLLECT INTO l_target_names
 FROM mgmt$target
 WHERE target_type = mgmt_user_defined_policy.G_DATABASE _TARGET_TYPE;

 IF (l_target_names IS NOT NULL) and (l_target_names.COUNT > 0)
 THEN
   FOR i in 1..l_target_names.COUNT LOOP
     mgmt_user_defined_policy.add_policy_to_target
      (
        p_policy_name  => 'Insufficient Control Files',
        p_target_type  => mgmt_user_defined_policy.G_DATABASE_TARGET_TYPE,
        p_target_name  => l_target_names(i)
      );
   END LOOP;
   COMMIT;
 END IF;

END;
/
```

8.3.3 Using Monitoring Templates

Another way to add a user-defined policy to existing targets is to:

1. Create a monitoring template.
2. Add the user-defined policy to the monitoring template.
3. Apply the monitoring template to the target.

8.4 Deleting a User-Defined Policy

This procedure deletes an existing user-defined policy. As part of deleting a user-defined policy:

- All associations of the policy to a target are automatically removed.
- All existing evaluations results for the policy are deleted.
- All references of this policy in existing monitoring templates are removed.
- The policy is deleted from the Policy Library.

If the policy is not deleted, an error message is raised using the RAISE_APPLICATION_ERROR procedure. The error number and message can be trapped like any Oracle error for processing.

The Management Repository owner has the permission to delete the policy.

Syntax

```
PROCEDURE delete_policy
(
    p_policy_name           IN VARCHAR2,
    p_target_type           IN VARCHAR2
);
```

Parameters

Parameter	Description
p_policy_name	Name of the policy to be deleted
p_target_type	Type of target to which this policy is applicable Examples: G_HOST_TARGET_TYPE (Host) G_DATABASE_TARGET_TYPE (Oracle Database Instance)

Example

The following example depicts how to delete a policy.

Example 8–6 Deleting a User-Defined Policy

```
BEGIN
   MGMT_USER_DEFINED_POLICY.DELETE_POLICY
   (
       p_policy_name  => 'Insufficient Control Files',
       p_target_type  => mgmt_user_defined_policy.G_DATABASE_TARGET_TYPE
    );
COMMIT;
END;
/
```

8.5 Removing a User-Defined Policy from Existing Targets

There are two ways to remove a user-defined policy from existing targets:

- Using the Metric and Policy Settings UI
- Using a PL/SQL Procedure

8.5.1 Using the Metric and Policy Settings UI

To quickly remove a user-defined policy from existing targets, use the Metric and Policy Settings UI pages.

1. On the Grid Control home page, click the **Targets** tab.
2. On the resulting page, click the target type of the target from which you want to remove the user-defined policy.
3. Select the name of the target from which you want to remove the user-defined policy.
4. On the resulting Home page for the target, scroll to the Related Links section and click **Metric and Policy Settings**.
5. Choose the **Policies** subtab and click **Remove**.
6. Select the policy rule you want to remove and click **Continue**.
7. On the confirmation page, click **Yes** to remove the policy rule from the target. In addition, ensure that you click **OK** on the Policies page, otherwise the change will not take affect.

8.5.2 Using a PL/SQL Procedure

The following procedure is used to remove an existing user-defined policy from a target that exists in the Management Repository. If the association is not removed, an error message is raised using the RAISE_APPLICATION_ERROR procedure. The error number and message can be trapped like any Oracle error for processing.

Syntax

```
PROCEDURE remove_policy_from_target
(
    p_policy_name           IN VARCHAR2,
    p_target_type           IN VARCHAR2,
    p_target_name           IN VARCHAR2
);
```

Parameters

Parameter	Description
p_policy_name	Name of the policy to be removed
p_target_type	Type of target to which this policy is applicable Example: G_HOST_TARGET_TYPE (Host) G_DATABASE_TARGET_TYPE (Oracle Database Instance)
p_target_name	Name of the target

Example

The following example depicts how to remove a user-defined policy from existing targets.

Example 8–7 Remove a Policy from an Existing Target

```
BEGIN
  MGMT_USER_DEFINED_POLICY.REMOVE_POLICY_FROM_TARGET
  (
    p_policy_name        => 'Insufficient Control Files',
    p_target_type        => mgmt_user_defined_policy.G_DATABASE_TARGET_TYPE,
    p_target_name        => 'Finance'
  );
 COMMIT;
END;
/
```

Part II

Reference

This section of the guide provides reference information to be used while developing Management Plug-ins.

Part II contains the following chapters:

9

Management Repository Views

Enterprise Manager repository views are used to access information in the Management Repository for further processing and/or presentation.

> **Important:** Views supplied with Enterprise Manager 10g release 10.2 are provisional. Provisional views, although fully supported for the current release of Enterprise Manager 10g, may change in subsequent releases. Backward compatibility is not guaranteed for provisional views.

This chapter covers the following:

9.1 Overview

The Enterprise Manager Management Repository views provide access to target, metric, and monitoring information stored in the Management Repository. Accessing the repository will allow you to perform the following:

- Obtain pertinent application-specific information at the right level of granularity and density for a wider variety of users: IT staff, executives, developers.
- Send alerts for metric threshold violations.
- Perform historical analysis or additional computation on stored data.
- Seamless integration of Enterprise Manager alerts with user ticketing systems, such as iSupport and Remedy.

The Management Repository is the comprehensive source for all the management information for Enterprise Manager, with the key to extensibility being the repository's open schema. This open architecture allows users to customize how the information in the repository is used if Enterprise Manager's standard configuration does not meet their requirements. To facilitate easy access to information stored in the repository, Enterprise Manager supplies a comprehensive set of views rather than forcing the user to access repository base tables directly. Views buffer custom applications from any underlying changes to the repository schema and ensures up-stream applications will not break when the repository schema changes via patching or new releases.

Using Repository Views

Because the views are simple queries to a database, users can imbed these queries within any application code used to return information for further processing and/or display in the Enterprise Manager Grid Control console.

As shown in Example 9–1, "View Usage", the Java code uses Enterprise Manager views to query the Management Repository rather than accessing the repository tables directly. For each of four time windows, there are four SQL statements with questions marks ('?') as placeholders for the parameters.

Example 9–1 View Usage

```
public static final String hour_stmt =
"SELECT collection_timestamp, value "+
"FROM mgmt$metric_details " +
"WHERE target_type = ? and target_name = ? and metric_name = ? and metric_column=
? " +
"and collection_timestamp > sysdate - 1/24 " +
"ORDER BY collection_timestamp ";

public static final String day_stmt =
"SELECT rollup_timestamp, average "+
"FROM mgmt$metric_hourly " +
"WHERE target_type = ? and target_name = ? and metric_name = ? and metric_column=
? " +
     "and rollup_timestamp > sysdate - 1 " +
     "ORDER BY rollup_timestamp";

public static final String week_stmt =
"SELECT rollup_timestamp, average "+
"FROM mgmt$metric_daily " +
"WHERE target_type = ? and target_name = ? and metric_name = ? and metric_column=
? " +
"and rollup_timestamp > sysdate - 7 " +
"ORDER BY rollup_timestamp";

public static final String month_stmt =
"SELECT rollup_timestamp, average "+
"FROM mgmt$metric_daily " +
"WHERE target_type = ? and target_name = ? and metric_name = ? and metric_column=
```

```
? " +
"and rollup_timestamp > sysdate - 31 " +
"ORDER BY rollup_timestamp";
```

9.2 Monitoring Views

9.2.1 MGMT$METRIC_ERROR_HISTORY

MGMT$METRIC_ERROR_HISTORY displays the history of metric collection errors that have occurred in last 3 months.

Table 9–1 MGMT$METRIC_ERROR_HISTORY

Column	Datatype	Description
TARGET_NAME	VARCHAR2(256)	The name of the target for which the metric collection error occurred.
TARGET_TYPE	VARCHAR2(64)	Target type defines the set of metrics that are applicable for the target.
TARGET_GUID	RAW(16)	Target GUID of the target for which the metric collection error occurred.
METRIC_NAME	VARCHAR2(64)	Name of the metric for which the error occurred.
METRIC_LABEL	VARCHAR2(64)	User display name of the metric for which the error occurred.
COLL_NAME	VARCHAR2(256)	Name of the collection collecting the metric for which the error occurred.
COLLECTION_TIMESTAMP	DATE	Time when the collection error occurred.
ERROR_TYPE	NUMBER	Type of metric error. 0=Error 1=Warning
ERROR_MESSAGE	VARCHAR2(4000)	Text of the error message.

Usage Notes

List the history of metric errors for a given target.

List the history of metric errors for a given target and metric.

Queries using this view would use index if target_name and target_type were specified.

9.2.2 MGMT$METRIC_ERROR_CURRENT

MGMT$METRIC_ERROR_CURRENT displays the metric collections that are currently generating errors.

Table 9–2 MGMT$METRIC_ERROR_CURRENT

Column	Datatype	Description
TARGET_NAME	VARCHAR2(256)	Name of the target for which the metric collection error occurred.
TARGET_TYPE	VARCHAR2(64)	Target type defines the set of metrics that are applicable for the target.

Table 9–2 (Cont.) MGMT$METRIC_ERROR_CURRENT

Column	Datatype	Description
TARGET_GUID	RAW(16)	GUID of the target for which the metric collection error occurred.
METRIC_NAME	VARCHAR2(64)	Name of the metric for which the error occurred
METRIC_LABEL	VARCHAR2(64)	User display name of the metric for which the error occurred.
COLL_NAME	VARCHAR2(256)	Name of the collection collecting the metric for which the error occurred
COLLECTION_TIMESTAMP	DATE	Time when the collection error occurred.
ERROR_TYPE	NUMBER	Type of metric error. 0=Error 1=Warning
ERROR_MESSAGE	VARCHAR2(4000)	Text of the error message.

Usage Notes

List the current metric errors for a given target.

List the current metric errors for a given target and metric.

Queries using this view would use index if target_name and target_type were specified.

9.2.3 MGMT$TARGET_COMPONENTS

MGMT$TARGET_COMPONENTS displays information about the software components that are associated with a managed target as well as details about where the software components have been installed.

Table 9–3 MGMT$TARGET_COMPONENTS

Column	Datatype	Description
TARGET_NAME	VARCHAR2(256)	Name of the target where the metric was collected. The target name uniquely identifies a managed target within the Management Repository. The target name typically contains the name of the managed entity that was provided by the system or database administrator.
TARGET_TYPE	VARCHAR2(64)	Target type defines the set of metrics that are applicable for the target.
TARGET_GUID	RAW(16)	The unique global identifier (GUID) for the target.
HOST_NAME	VARCHAR2(64)	The name of the host where the software homes are located.
HOME_NAME	VARCHAR2(64)	The name for the location where the software is installed. This might be a user-supplied name or a name supplied by the Oracle Universal Installer.
HOME_TYPE	VARCHAR2(64)	Three categories of homes are available: · ORACLE_HOME · APPL_TOP · INDEPENDENT · UNKNOWN

Table 9–3 (Cont.) MGMT$TARGET_COMPONENTS

Column	Datatype	Description
HOME_LOCATION	VARCHAR2(128)	The file system path to the root of the home.
COMPONENT_NAME	VARCHAR2(128)	The name of the software component.
COMPONENT_ EXTERNAL_ NAME	VARCHAR2(128)	The external name of the target, as specified during installation.
COMPONENT_VERSION	VARCHAR2(64)	The current version of the software component. If the software component has been patched, this version number may be different than the base version number.
COMPONENT_BASE_ VERSION	VARCHAR2(64)	Base version of the software component.
SNAPSHOT_GUID	RAW(16)	GUID of the snapshot.

Usage Notes

This view can be used to identify the software components that are part of a specific managed target.

Access to this view will use an index if the query references the target name and target type, the home location, or the host name.

9.2.4 MGMT$BLACKOUT_HISTORY

MGMT$BLACKOUT_HISTORY displays a historical log of changes in the blackout state for a managed target. In addition, the view can be used to generate a list of targets that were in a blackout period for a specific period of time.

Table 9–4 MGMT$BLACKOUT_HISTORY

Column	Datatype	Description
BLACKOUT_NAME	VARCHAR(64)	The name of the blackout.
CREATED_BY	VARCHAR(256)	The Enterprise Manager administrator who created the blackout.
BLACKOUT_GUID	RAW(16)	The unique global identifier (GUID) for the blackout.
START_TIME	DATE	Start of the blackout period for the managed target.
END_TIME	DATE	End of the blackout period for the managed target. If the target is currently in a blackout period, the END_ TIMESTAMP date will be NULL.
TARGET_NAME	VARCHAR2(64)	Name of the target where the metric was collected. The target name uniquely identifies a managed target within the Management Repository. The target name typically contains the name of the managed entity that was provided by the system or database administrator.
TARGET_TYPE	VARCHAR2(64)	Types of targets may include databases, hosts, web servers, applications, or Application Servers. The definer of the collection definition at the Management Agent defines the target type. The target type defines the set of metrics that are collected for a managed target within the Management Repository.

Table 9–4 (Cont.) MGMT$BLACKOUT_HISTORY

Column	Datatype	Description
STATUS	NUMBER(2)	Current status of the blackout. Status Values: 0: Scheduled 1: Start Processing 2: Start Partial 4: Started 5: Stop Pending 6: Stop Failed 7: Stop Partial 8: Edit Failed 9: Edit Partial 10: Stopped 11: Ended 12: Partial Blackout 13: Modify Pending

Usage Notes

Queries using this view will use an index if they reference the target_name, target_type, start_timestamp, or end_timestamp.

Typically, blackout history information retrieved using this view will be ordered by target_name, target_type, and start_timestamp.

9.2.5 MGMT$BLACKOUTS

MGMT$BLACKOUTS displays all blackout definition information along with current schedules.

Table 9–5 MGMT$BLACKOUTS

Column	Datatype	Description
BLACKOUT_NAME	VARCHAR(64)	The name of the blackout.
BLACKOUT_GUID	RAW(16)	The unique global identifier (GUID) of the blackout.
REASON	VARCHAR(64)	Purpose of the blackout. Reasons are chosen from a predefined list by the report owner.
DESCRIPTION	VARCHAR2(2000)	Detailed information about the blackout.

Table 9–5 (Cont.) MGMT$BLACKOUTS

Column	Datatype	Description
STATUS	NUMBER(2)	Current status of the blackout. Status Values: 0: Scheduled 1: Start Processing 2: Start Partial 4: Started 5: Stop Pending 6: Stop Failed 7: Stop Partial 8: Edit Failed 9: Edit Partial 10: Stopped 11: Ended 12: Partial Blackout 13: Modify Pending
CREATED_BY	VARCHAR(256)	Administrator who created the blackout. CREATED_BY returns SYSTEM as the blackout owner if the blackout was created using the Enterprise Manager Command Line Interface.
LAST_START_TIME	DATE	Last time the blackout successfully started.
LAST_END_TIME	DATE	Last time the blackout successfully ended.
SCHEDULED_TIME	NUMBER	Possible values are: 0 - Immediate schedule 1 - Run once at specified time 2 - Run on interval 3 - Run daily 4 - Run on specified days of the week 5 - Run on specified days of the month 6 - Run on specified days of the year
SCHEDULE_START_TIME	DATE	Time the blackout is scheduled to start.
SCHEDULE_END_TIME	DATE	Time the blackout is scheduled to end.
DURATION	NUMBER	Duration of the blackout in minutes.

9.2.6 MGMT$ALERT_ANNOTATIONS

MGMT$ALERT_ANNOTATIONS displays a summary of alert annotations.

Table 9–6 MGMT$

Column	Datatype	Description
TARGET_NAME	VARCHAR2(256)	Name of the target where the metric was collected. The target name uniquely identifies a managed target within the Management Repository. The target name typically contains the name of the managed entity that was provided by the system or database administrator
TARGET_TYPE	VARCHAR2(64)	Target type defines the set of metrics that are applicable for the target.
TARGET_GUID	RAW(16)	The unique global identifier (GUID) for the target.

Table 9–6 (Cont.) MGMT$

Column	Datatype	Description
METRIC_NAME	VARCHAR2(64)	Name of the metric.
METRIC_COLUMN	VARCHAR2(64)	If the metric defined at collection time is a TABLE type, the name of the column in the table that is represented by this metric is written to this field. The metric column is part of the primary key that uniquely identifies the value of this table that this metric represents.
METRIC_LABEL	VARCHAR2(64)	Display name for the metric.
COLUMN_LABEL	VARCHAR2(64)	Display name for the column of a TABLE metric.
KEY_VALUE	VARCHAR2(256)	Key value of the alert violation. For composite keys, this is the first part of the key.
KEY_VALUE2	VARCHAR2(256)	For composite keys, this is the second part of the key.
KEY_VALUE3	VARCHAR2(256)	For composite keys, this is the third part of the key.
KEY_VALUE4	VARCHAR2(256)	For composite keys, this is the fourth part of the key.
KEY_VALUE5	VARCHAR2(256)	For composite keys, this is the fourth part of the key.
MESSAGE	VARCHAR2(4000)	The metric severity message.
ALERT_STATE	VARCHAR2(20)	A user readable description of the internal alert code that is sent from the Management Agent to identify the state of the alert condition. An alert record is transferred to the repository from the Management Agent each time the metric threshold is crossed in either direction, or if the Management Agent is restarted. The value of this column will contain one of the following strings: · Clear · Warning · Critical
COLLECTION_ TIMESTAMP	DATE	Date and time when the alert condition was detected by the Management Agent.
ANNOTATION_MESSAGE	VARCHAR2(4000)	Annotation text.
ANNOTATION_TIMESTAMP	DATE	Time the annotation was created.
ANNOTATED_BY	VARCHAR2(64)	Enterprise Manager administrator who authored the the annotation.
SOURCE_OBJ_GUID	RAW(16)	Unique global identifier (GUID) of the metric severity associated with this annotation.

9.2.7 MGMT$ALERT_NOTIF_LOG

MGMT$ALERT_NOTIF_LOG displays a summary of alert notifications.

Table 9–7 MGMT$MGMT$ALERT_NOTIF_LOG

Column	Datatype	Description
TARGET_NAME	VARCHAR2(256)	Name of the target where the metric was collected. The target name uniquely identifies a managed target within the Management Repository. The target name typically contains the name of the managed entity that was provided by the system or database administrator.
TARGET_TYPE	VARCHAR2(64)	The target type defines the set of metrics that are applicable for the target.
TARGET_GUID	RAW(16)	The unique global identifier (GUID) for the target.
METRIC_NAME	VARCHAR2(64)	The name of the metric being defined.
METRIC_COLUMN	VARCHAR2(64)	For table metrics, the metric column contains the name of the column in the table that is being defined. If the metric that is being defined is not a table metric, the value in this column is a single space. For example, if the table describing the MGMT$TARGET_TYPE view was being defined as a table metric, Column Name, Data Type, and Description would be metric columns.
METRIC_LABEL	VARCHAR2(64)	A intuitive display name for the metric that is being defined.
COLUMN_LABEL	VARCHAR2(64)	For table metrics, the column label contains a user understandable display name for the metric column.
KEY_VALUE	VARCHAR2(256)	For table metrics, this column contains the value of the key column for the row in the table whose thresholds are being defined. If the thresholds are not for a table metric, or the thresholds apply for all rows in the metric column, then the value in this column will contain a single space.
KEY_VALUE2	VARCHAR2(256)	For composite keys, this is the second part of the key.
KEY_VALUE3	VARCHAR2(256)	For composite keys, this is the third part of the key.
KEY_VALUE4	VARCHAR2(256)	For composite keys, this is the fourth part of the key.
KEY_VALUE5	VARCHAR2(256)	For composite keys, this is the fifth part of the key.
MESSAGE	VARCHAR2(4000)	An optional message that is generated when the alert is created that provides additional information about the alert condition.
ALERT_STATE	VARCHAR2(20)	A user readable description of the internal alert code that is sent from the Management Agent to identify the state of the alert condition. A alert record is transferred to the repository from the Management Agent each time the metric threshold is crossed in either direction, or if the Management Agent is restarted. The value of this column will contain one of the following strings: · Clear · Warning · Critical

Table 9–7 (Cont.) MGMT$MGMT$ALERT_NOTIF_LOG

Column	Datatype	Description
COLLECTION_ TIMESTAMP	DATE	Date and time when the alert condition was detected by the Management Agent.
DELIVERY_MESSAGE	VARCHAR2(1024)	Message indicating the success or failure of the notification delivery.
DELIVERY_TIMESTAMP	DATE	Time at which the log message was created.
SOURCE_OBJ_GUID	RAW(16)	Unique global identifer (GUID) of the metric severity associated with this delivery message.

9.2.8 MGMT$TARGET_METRIC_COLLECTIONS

MGMT$TARGET_METRIC_COLLECTIONS displays information about the metric collections.

Table 9–8 MGMT$TARGET_METRIC_COLLECTIONS

Column	Datatype	Description
TARGET_NAME	VARCHAR2(64)	Name of the target where the metric was collected. The target name uniquely identifies a managed target within the Management Repository. The target name typically contains the name of the managed entity that was provided by the system or database administrator.
TARGET_TYPE	VARCHAR2(64)	The target type defines the set of metrics that are applicable for the target.
TARGET_GUID	RAW(16)	Unique global identifier (GUID) for the target.
METRIC_NAME	VARCHAR2(64)	Name of the metric being defined.
METRIC_COLUMN	VARCHAR2(64)	For table metrics, the metric column contains the name of the column in the table that is being defined. If the metric that is being defined is not a table metric, the value in this column is a single space. For example, if the table describing the MGMT$TARGET_TYPE view was being defined as a table metric, Column Name, Data Type, and Description would be metric columns.
METRIC_GUID	RAW(16)	Unique global identifier (GUID) for the metric. This ID can be used to associate metric information with metric data information during reporting.
COLLECTION_NAME	VARCHAR2(64)	The name of the collection.
IS_ENABLED	NUMBER(1)	Indicates whether the collection is currently enabled. 0=not enabled 1=enabled.
IS_REPOSITORY	NUMBER(1)	Indicates whether this is a repository-side collection. A repository-side collection has a PL/SQL evaluation procedure that is responsible for calculating the metric values.

Table 9–8 (Cont.) MGMT$TARGET_METRIC_COLLECTIONS

Column	Datatype	Description
FREQUENCY_CODE	NUMBER	The metric collection frequency type. Possible values are: 1: One Time 2: Interval 3: Daily 4: Weekly 5: Monthly 6: Yearly 7: On Demand
COLLECTION_FREQUENCY	VARCHAR2	Frequency of the metric collection. Value displayed is dependent on the frequency code: [2] For One Time, the start date-time is stored in DD-MON-YY HH24:MI format. [2] For Interval type, the frequency in minutes is stored. [2] For Daily/Weekly/Monthly/Yearly types, the hour and minute of collection is stored in HH24:MI format. [2] For On-Demand type, On-Demand is stored.
UPLOAD_POLICY	NUMBER	The frequency with which the metric data is uploaded/stored.

Usage Notes

List the metric collections for a given target.

9.2.9 MGMT$TARGET_METRIC_SETTINGS

MGMT$TARGET_METRIC_SETTINGS displays information about the current metric setting stored for all targets in the Management Repository. This view provides information for both Agent-side and repository-side metrics.

Table 9–9 MGMT$TARGET_METRIC_SETTINGS

Column	Datatype	Description
TARGET_NAME	VARCHAR2(64)	Name of the target where the metric was collected. The target name uniquely identifies a managed target within the Management Repository. The target name typically contains the name of the managed entity that was provided by the system or database administrator.
TARGET_TYPE	VARCHAR2(64)	The target type defines the set of metrics that are applicable for the target.
TARGET_GUID	RAW(16)	The unique global identifier (GUID) for the target.
METRIC_NAME	VARCHAR2(64)	Name of the metric being defined.

Table 9–9 (Cont.) MGMT$TARGET_METRIC_SETTINGS

Column	Datatype	Description
METRIC_COLUMN	VARCHAR2(64)	For table metrics, the metric column contains the name of the column in the table that is being defined. If the metric that is being defined is not a table metric, the value in this column is a single space. For example, if the table describing the MGMT$TARGET_TYPE view was being defined as a table metric, Column Name, Data Type, and Description would be metric columns.
METRIC_GUID	RAW(16)	The unique global identifier for the metric. This ID can be used to associate metric information with metric data information during reporting.
COLLECTION_NAME	VARCHAR2(64)	The name of the collection.
CATEGORY	VARCHAR2(64)	The name of the category the metric. Refer to MGMT$METRIC_CATEGORIES for the list of all metric categories.
KEY_VALUE	VARCHAR2(256)	The key value of the metric setting. For composite keys, this is the first part of the key. If the thresholds are not for a table metric, or the thresholds apply for all rows in the metric column, then the value in this column will contain a single space.
KEY_VALUE2	VARCHAR2(256)	For composite keys, this is the second part of the key.
KEY_VALUE3	VARCHAR2(256)	For composite keys, this is the third part of the key.
KEY_VALUE4	VARCHAR2(256)	For composite keys, this is the fourth part of the key.
KEY_VALUE5	VARCHAR2(256)	For composite keys, this is the fifth part of the key.
KEY_OPERATOR	NUMBER	Specifies whether the key_value columns have any SQL wildcards. For single key column metrics, the value is 1 if the key_value has wildcard characters, 0 otherwise. For metrics with multiple keys, a list of operators for all key columns will be stored here. For example, a metric with 3 keys (k1, k2, k3) where K1 and K2 use wildcards and K3 uses exact match, then 011 is stored in this column.
HAS_ACTIVE_BASELINE	NUMBER(1)	The is a flag that specifies that the metric rows with this key_value has an active baseline and any user updates to thresholds/parameter values should be ignored.
PREVENT_OVERRIDE	NUMBER(1)	The is a flag that specifies that the metric rows with this key_value has a template override flag. Once the template override flag is ON, any template application will not update the threshold/parameter values.

Table 9–9 (Cont.) MGMT$TARGET_METRIC_SETTINGS

Column	Datatype	Description
WARNING_OPERATOR	NUMBER	Defines the warning threshold condition to be applied 0 - GT 1 - EQ 2 - LT 3 - LE 4 - GE 5 - CONTAINS 6 - NE 7 - MATCH : regular expression
WARNING_THRESHOLD	VARCHAR2(256)	The warning threshold value.
CRITICAL_OPERATOR	NUMBER	Defines the critical threshold condition to be applied. 0 - GT 1 - EQ 2 - LT 3 - LE 4 - GE 5 - CONTAINS 6 - NE 7 - MATCH : regular expression
CRITICAL_THRESHOLD	VARCHAR2(256)	The critical threshold value.
OCCURRENCE_COUNT	NUMBER	The number of times the test has to trigger to raise a violation.
WARNING_ACTION_TYPE	VARCHAR2(32)	Specifies the job type of the warning corrective action when WARNING_ACTION_TYPE is "Corrective-Action".
WARNING_ACTION_JOB_OWNER	VARCHAR2(256)	Specifies the job owner of the warning corrective action when WARNING_ACTION_TYPE is "Corrective-Action"
WARNING_ACTION_JOB_NAME	VARCHAR2(64)	Specifies the job name of the warning corrective action when WARNING_ACTION_TYPE is "Corrective-Action"
CRITICAL_ACTION_TYPE	VARCHAR2(17)	The critical corrective action type configured. Possible values are : No-Action : when no action is configured Corrective-Action : when a repository side corrective action is configured Agent-Fixit Job : when an Agent side fix-it job is configured.
CRITICAL_ACTION_JOB_TYPE	VARCHAR2(32)	Specifies the job type of the critical corrective action when WARNING_ACTION_TYPE is "Corrective-Action".
CRITICAL_ACTION_JOB_OWNER	VARCHAR2(256)	Specifies the job owner of the critical corrective action when WARNING_ACTION_TYPE is "Corrective-Action".
CRITICAL_ACTION_JOB_NAME	VARCHAR2(64)	Specifies the job name of the critical corrective action when WARNING_ACTION_TYPE is "Corrective-Action".

Usage Notes

List all the metric setting for a given target.

List the metric settings for a given target and metric.

List the corrective actions assigned for a given target-metric.

9.2.10 MGMT$AVAILABILITY_CURRENT

MGMT$AVAILABILITY_CURRENT displays information about the most recent target availability information stored in the Management Repository.

Table 9–10 MGMT$AVAILABILITY_CURRENT

Column	Datatype	Description
TARGET_NAME	VARCHAR2(64)	Name of the target where the metric was collected. The target name uniquely identifies a managed target within the Management Repository. The target name typically contains the name of the managed entity that was provided by the system or database administrator.
TARGET_TYPE	VARCHAR2(64)	The target type defines the set of metrics that are applicable for the target.
START_TIMESTAMP	DATE	The time when the target availability status change was first detected.
AVAILABILITY_STATUS	VARCHAR(15)	Current target availability status. This column contains one of the following values: ² Target Down ² Target Up ² Metric Error ² Agent Down ² Unreachable ² Blackout ² Pending/Unknown

Usage Notes

Get the current availability status of a given target.

9.2.11 MGMT$AVAILABILITY_HISTORY

MGMT$AVAILABILITY_HISTORY displays detailed historical information about changes in the availability status for a target over time.

Table 9–11 MGMT$AVAILABILITY_HISTORY

Column	Datatype	Description
TARGET_NAME	VARCHAR2(64)	Name of the target where the metric was collected. The target name uniquely identifies a managed target within the Management Repository. The target name typically contains the name of the managed entity that was provided by the system or database administrator.
TARGET_TYPE	VARCHAR2(64)	The target type defines the set of metrics that are applicable for the target.
TARGET_GUID	RAW(16)	The unique global identifier for the target.

Table 9–11 (Cont.) MGMT$AVAILABILITY_HISTORY

Column	Datatype	Description
START_TIMESTAMP	DATE	The time when the target availability status change was first detected.
END_TIMESTAMP	DATE	The time when the target availability status change was last detected.
AVAILABILITY_STATUS	VARCHAR2(10)	Target availability status. This column will contain one of the following values: ² Target Down ² Target Up ² Metric Error ² Agent Down ² Unreachable ² Blackout ² Pending/Unknown

Usage Notes

Access to this view will use an index if the query references the member TARGET_NAME, TARGET_TYPE and the START_TIMESTAMP.

9.2.12 MGMT$ALERT_CURRENT

MGMT$ALERT_CURRENT displays current information for any alerts that are logged in the Management Repository that are in a non-clear state. Only the most recent open alert in a non-clear status for a given metric will be displayed through this view.

Table 9–12 MGMT$

Column	Datatype	Description
TARGET_NAME	VARCHAR2(64)	Name of the target where the metric was collected. The target name uniquely identifies a managed target within the Management Repository. The target name typically contains the name of the managed entity that was provided by the system or database administrator.
TARGET_TYPE	VARCHAR2(64)	The target type defines the set of metrics that are applicable for the target.
TARGET_GUID	RAW(16)	The unique global identifier for the target.
VIOLATION_GUID	RAW(16)	Unique identifier for the alert.
METRIC_NAME	VARCHAR2(64)	Name of the metric being defined.
METRIC_COLUMN	VARCHAR2(64)	For table metrics, the metric column contains the name of the column in the table that is being defined. If the metric that is being defined is not a table metric, the value in this column is a single space. For example, if this table describing the MGMT$TARGET_TYPE view was being defined as a table metric, Column Name, Data Type, and Description would be metric columns.
METRIC_LABEL	VARCHAR2(64)	An intuitive display name for the metric that is being defined.
COLUMN_LABEL	VARCHAR2(64)	For table metrics, the column label contains an intuitive display name for the metric column.

Table 9–12 (Cont.) MGMT$

Column	Datatype	Description
KEY_VALUE	VARCHAR2(256)	The key value for which the alert has been recorded. For composite keys, this is the first part of the key.
KEY_VALUE2	VARCHAR2(256)	For composite keys, this is the second part of the key for which the alert has been recorded
KEY_VALUE3	VARCHAR2(256)	For composite keys, this is the third part of the key for which the alert has been recorded
KEY_VALUE4	VARCHAR2(256)	For composite keys, this is the fourth part of the key for which the alert has been recorded
KEY_VALUE5	VARCHAR2(256)	For composite keys, this is the fifth part of the key for which the alert has been recorded
COLLECTION_ TIMESTAMP	DATE	The date-time when the alert condition was detected by the Management Agent.
ALERT_STATE	VARCHAR2(20)	A user readable description of the internal alert code that is sent from the Management Agent to identify the state of the alert condition. A alert record is transferred to the repository from the Management Agent each time the metric threshold is crossed in either direction, or if the Management Agent is restarted. The value of this column will contain one of the following strings: · Warning · Critical If the metric's alert condition goes into a clear state, it will no longer be visible from this view.
VIOLATION_TYPE	VARCHAR2(19)	A user readable description of the type of violation. Possible values are : Threshold Violation, when the alert is triggered based on a metric threshold Availability, when the alert is triggered for an availability metric Policy Violation, when the alert is triggered based on a policy violation.
MESSAGE	VARCHAR2(1024)	An optional message that is generated when the alert is created that provides additional information about the alert condition.
MESSAGE_NLSID	VARCHAR2(64)	The NLSID of the alert message
MESSAGE_PARAMS	VARCHAR2(4000)	Contains the URL encoded parameters separated by "&" to be used to format the alert message.
ACTION_MESSAGE	VARCHAR2(4000)	Suggested action message in english for this alert
ACTION_MESSAGE_NLSID	VARCHAR2(64)	The NLS ID of the action message.
ACTION_MESSAGE_PARAMS	VARCHAR2(4000)	Contains the URL encoded parameters for translating action message
TYPE_DISPLAY_NAME	VARCHAR2(128)	The display name of the target type

Usage Notes

List the current alerts that are in a non-clear state for a metric, set of metrics, or for a managed target. If the user is only interested in non-clear alerts, counts or selects, using this view provide better performance than using the MGMT$ALERT_DETAILS view.

Access to this view will use an index if the query references the member target name, target type, metric name, and metric column or a subset of these columns if they are included as listed above from left to right.

9.2.13 MGMT$ALERT_HISTORY

MGMT$ALERT_HISTORY displays historical information for any alerts that are logged in the Management Repository.

Table 9–13 MGMT$ALERT_HISTORY

Column	Datatype	Description
TARGET_NAME	VARCHAR2(64)	Name of the target where the metric was collected. The target name uniquely identifies a managed target within the Management Repository. The target name typically contains the name of the managed entity that was provided by the system or database administrator.
TARGET_TYPE	VARCHAR2(64)	The target type defines the set of metrics that are applicable for the target.
TARGET_GUID	RAW(16)	The unique global identifier for the target.
VIOLATION_GUID	RAW(16)	Unique identifier for the alert.
METRIC_NAME	VARCHAR2(64)	Name of the metric being defined.
METRIC_COLUMN	VARCHAR2(64)	For table metrics, the metric column contains the name of the column in the table that is being defined. If the metric that is being defined is not a table metric, the value in this column is a single space. For example, if this table describing the MGMT$TARGET_TYPE view was being defined as a table metric, Column Name, Data Type, and Description would be metric columns.
METRIC_LABEL	VARCHAR2(64)	An intuitive display name for the metric that is being defined.
COLUMN_LABEL	VARCHAR2(64)	For table metrics, the column label contains an intuitive display name for the metric column.
KEY_VALUE	VARCHAR2(256)	The key value for which the alert has been recorded. For composite keys, this is the first part of the key.
KEY_VALUE2	VARCHAR2(256)	For composite keys, this is the second part of the key for which the alert has been recorded
KEY_VALUE3	VARCHAR2(256)	For composite keys, this is the third part of the key for which the alert has been recorded
KEY_VALUE4	VARCHAR2(256)	For composite keys, this is the fourth part of the key for which the alert has been recorded
KEY_VALUE5	VARCHAR2(256)	For composite keys, this is the fifth part of the key for which the alert has been recorded
COLLECTION_ TIMESTAMP	DATE	The date-time when the alert condition was detected by the Management Agent.

Table 9–13 (Cont.) MGMT$ALERT_HISTORY

Column	Datatype	Description
ALERT_STATE	VARCHAR2(20)	A user readable description of the internal alert code that is sent from the Management Agent to identify the state of the alert condition. A alert record is transferred to the repository from the Management Agent each time the metric threshold is crossed in either direction, or if the Management Agent is restarted. The value of this column will contain one of the following strings: · Warning · Critical If the metric's alert condition goes into a clear state, it will no longer be visible from this view.
ALERT_DURATION	NUMBER	The time, in hours, from when the alert condition was first detected until it was cleared.
MESSAGE	VARCHAR2(1024)	An optional message that is generated when the alert is created that provides additional information about the alert condition.
MESSAGE_NLSID	VARCHAR2(64)	The NLSID of the alert message
MESSAGE_PARAMS	VARCHAR2(4000)	Contains the URL encoded parameters separated by "&" to be used to format the alert message.
ACTION_MESSAGE	VARCHAR2(4000)	Suggested action message in english for this alert
ACTION_MESSAGE_NLSID	VARCHAR2(64)	The NLS ID of the action message.
ACTION_MESSAGE_PARAMS	VARCHAR2(4000)	Contains the URL encoded parameters for translating action message
VIOLATION_TYPE	VARCHAR2(19)	An intuitive description of the type of violation. Possible values are : Threshold Violation: When the alert is triggered based on a metric threshold Availability: When the alert is triggered for an availability metric Policy Violation: When the alert is triggered based on a policy violation.
TYPE_DISPLAY_NAME	VARCHAR2(128)	The display name of the target type

Usage Notes

List the historical details of alerts for a managed target or a specific metric ordered by time.

Determine the notification status for a specific alert or set of alerts.

Generate statistical reports on alert frequency using criteria such as by target, by metric, by groups of metrics.

Custom implementation of event filtering or causal analysis.

Custom aggregation (grouping) of alert information into categories.

Custom alert status queries or aggregations for specific time periods. i.e. What was my most problematic metric for target x last week? How many times did alert y occur between times w and x?

Generating time based histograms of alerts.

The view does not force any specific order on the rows returned. The assumption is that if the user wants to order the alert records that are returned in a specific way, they will add an order by clause to their select query.

Access to this view will use an index if the query references the member target name, target type, metric name, and metric column or a subset of these columns if they are included as listed above from left to right.

Queries that use the collection_timestamp or that query using the alert_state will also be indexed.

9.2.14 MGMT$METRIC_DETAILS

MGMT$METRIC_DETAILS displays a rolling 25 hour window of individual metric samples. These are the metric values for the most recent sample that has been loaded into the Management Repository plus any earlier samples that have not been aggregated into hourly statistics.

Table 9–14 MGMT$METRIC_DETAILS

Column	Datatype	Description
TARGET_NAME	VARCHAR2(64)	Name of the target where the metric was collected. The target name uniquely identifies a managed target within the Management Repository. The target name typically contains the name of the managed entity that was provided by the system or database administrator.
TARGET_TYPE	VARCHAR2(64)	The target type defines the set of metrics that are applicable for the target.
TARGET_GUID	RAW(16)	The unique global identifier for the target.
METRIC_NAME	VARCHAR2(64)	Name of the metric being defined.
METRIC_TYPE	VARCHAR2(20)	A DECODE of the internal numeric type of the metric that is being defined. This column will contain one of the following values: Number String Table Raw External
METRIC_COLUMN	VARCHAR2(64)	For table metrics, the metric column contains the name of the column in the table that is being defined. If the metric that is being defined is not a table metric, the value in this column is a single space. For example, if this table describing the MGMT$TARGET_TYPE view was being defined as a table metric, Column Name, Data Type, and Description would be metric columns.
METRIC_LABEL	VARCHAR2(64)	An intuitive display name for the metric that is being defined.
COLUMN_LABEL	VARCHAR2(64)	For table metrics, the column label contains an intuitive display name for the metric column.
COLLECTION_ TIMESTAMP	DATE	The date-time when the alert condition was detected by the Management Agent.

Table 9–14 (Cont.) MGMT$METRIC_DETAILS

Column	Datatype	Description
VALUE	VARCHAR2(4000)	Since current metric values can be a numeric or a string type, this column returns the value of the metric as a string. If the user of the view is restricting the query to numeric metric values, they can use the TO_NUMBER SQL function to return the values in numeric form.
KEY_VALUE	VARCHAR2(256)	The key value for which the alert has been recorded. For composite keys, this is the first part of the key.
KEY_VALUE2	VARCHAR2(256)	For composite keys, this is the second part of the key for which the alert has been recorded
KEY_VALUE3	VARCHAR2(256)	For composite keys, this is the third part of the key for which the alert has been recorded
KEY_VALUE4	VARCHAR2(256)	For composite keys, this is the fourth part of the key for which the alert has been recorded
KEY_VALUE5	VARCHAR2(256)	For composite keys, this is the fifth part of the key for which the alert has been recorded

Usage Notes

Show the individual values for a metric over time.

Identify time periods when abnormal samples for metric were collected.

Calculate the correlation coefficient between two or more metrics.

Provide metric values that are associated with an alert.

Queries using this view will use an index if the queries use the target name, the target type, metric name. metric column, and key value, or if they are based upon the collection_timestamp.

9.2.15 MGMT$METRIC_CURRENT

MGMT$METRIC_CURRENT displays information on the most recent metric values that have been loaded into the Management Repository.

Table 9–15 MGMT$METRIC_CURRENT

Column	Datatype	Description
TARGET_NAME	VARCHAR2(64)	Name of the target where the metric was collected. The target name uniquely identifies a managed target within the Management Repository. The target name typically contains the name of the managed entity that was provided by the system or database administrator.
TARGET_TYPE	VARCHAR2(64)	The target type defines the set of metrics that are applicable for the target.
TARGET_GUID	RAW(16)	The unique global identifier for the target.
METRIC_NAME	VARCHAR2(64)	Name of the metric being defined.

Table 9–15 (Cont.) MGMT$METRIC_CURRENT

Column	Datatype	Description
METRIC_TYPE	VARCHAR2(20)	A DECODE of the internal numeric type of the metric that is being defined. This column will contain one of the following values: Number String Table Raw External
METRIC_COLUMN	VARCHAR2(64)	For table metrics, the metric column contains the name of the column in the table that is being defined. If the metric that is being defined is not a table metric, the value in this column is a single space. For example, if this table describing the MGMT$TARGET_TYPE view was being defined as a table metric, Column Name, Data Type, and Description would be metric columns.
METRIC_LABEL	VARCHAR2(64)	An intuitive display name for the metric that is being defined.
COLUMN_LABEL	VARCHAR2(64)	For table metrics, the column label contains an intuitive display name for the metric column.
COLLECTION_ TIMESTAMP	DATE	The date-time when the alert condition was detected by the Management Agent.
VALUE	VARCHAR2(4000)	Since current metric values can be a numeric or a string type, this column returns the value of the metric as a string. If the user of the view is restricting the query to numeric metric values, they can use the TO_NUMBER SQL function to return the values in numeric form.
KEY_VALUE	VARCHAR2(256)	The key value for which the alert has been recorded. For composite keys, this is the first part of the key.
KEY_VALUE2	VARCHAR2(256)	For composite keys, this is the second part of the key for which the alert has been recorded
KEY_VALUE3	VARCHAR2(256)	For composite keys, this is the third part of the key for which the alert has been recorded
KEY_VALUE4	VARCHAR2(256)	For composite keys, this is the fourth part of the key for which the alert has been recorded
KEY_VALUE5	VARCHAR2(256)	For composite keys, this is the fifth part of the key for which the alert has been recorded

Usage Notes

Retrieve the most recent value for a metric that is stored in the Management Repository.

Retrieve the latest metrics for a target or metric for a specific time period.

Queries using this view will use an index if the queries use target name, the target type, metric name. metric column, and key value, or if they are based upon the collection_timestamp.

9.2.16 MGMT$METRIC_HOURLY

MGMT$METRIC_HOURLY displays metric statistics information that have been aggregated from the individual metric samples into hourly time periods. For example, if a metric is collected every 15 minutes, the 1 hour rollup would aggregate the 4 samples into a single hourly value by averaging the 4 individual samples together. The current hour of statistics may not be immediately available from this view. The timeliness of the information provided from this view is dependent on when the query against the view was executed and when the hourly rollup table was last refreshed.

Table 9–16 MGMT$METRIC_HOURLY

Column	Datatype	Description
TARGET_NAME	VARCHAR2(64)	Name of the target where the metric was collected. The target name uniquely identifies a managed target within the Management Repository. The target name typically contains the name of the managed entity that was provided by the system or database administrator.
TARGET_TYPE	VARCHAR2(64)	The target type defines the set of metrics that are applicable for the target.
TARGET_GUID	RAW(16)	The unique global identifier for the target.
METRIC_NAME	VARCHAR2(64)	Name of the metric being defined.
METRIC_COLUMN	VARCHAR2(64)	For table metrics, the metric column contains the name of the column in the table that is being defined. If the metric that is being defined is not a table metric, the value in this column is a single space. For example, if this table describing the MGMT$TARGET_TYPE view was being defined as a table metric, Column Name, Data Type, and Description would be metric columns.
METRIC_LABEL	VARCHAR2(64)	An intuitive display name for the metric that is being defined.
COLUMN_LABEL	VARCHAR2(64)	For table metrics, the column label contains an intuitive display name for the metric column.
KEY_VALUE	VARCHAR2(256)	The key value for which the alert has been recorded. For composite keys, this is the first part of the key.
KEY_VALUE2	VARCHAR2(256)	For composite keys, this is the second part of the key for which the alert has been recorded
KEY_VALUE3	VARCHAR2(256)	For composite keys, this is the third part of the key for which the alert has been recorded
KEY_VALUE4	VARCHAR2(256)	For composite keys, this is the fourth part of the key for which the alert has been recorded
KEY_VALUE5	VARCHAR2(256)	For composite keys, this is the fifth part of the key for which the alert has been recorded
ROLLUP_TIMESTAMP	DATE	The rollup timestamp identifies the start of the rollup period. For the one-hour rollups, samples that fall within the hourly boundaries from minute 00 through minute 59 inclusive will be combined. For example, samples from 12:00 AM through 12:59 AM would be combined into a single aggregated record with a rollup timestamp of "date" 12:00 AM.
SAMPLE_COUNT	NUMBER	The number of non-NULL samples for the metric that were aggregated.

Table 9–16 (Cont.) MGMT$METRIC_HOURLY

Column	Datatype	Description
AVERAGE	NUMBER	The average of the metric values for the samples that have been included in the rollup period.
MINIMUM	NUMBER	The minimum value for the metric for the samples that have been included in the rollup period.
MAXIMUM	NUMBER	The maximum value for the metric for samples that have been included in the rollup period.
STANDARD_DEVIATION	NUMBER	The standard deviation for the metric values that have been included in the rollup period.

Usage Notes

This view provides the best level of granularity to show changes in a metric's value over the course of a day.

Identify hourly time periods when a metric or sets of metrics are maximized.

Understand how the variability of a metric over a one hour time period.

Identify the values of the collected metrics for a target when a particular hour has been identified as problematic.

Queries using this view will use an index if the queries use the target_name, the metric_name, or if they are based upon the rollup_timestamp.

9.2.17 MGMT$METRIC_DAILY

MGMT$METRIC_DAILY displays metric statistics that have been aggregated from the samples collected over the previous twenty-four hour time period. The timeliness of the information provided from this view is dependent on when the query against the view was executed and when the hourly rollup table was last refreshed.

Table 9–17 MGMT$METRIC_DAILY

Column	Datatype	Description
TARGET_NAME	VARCHAR2(64)	Name of the target where the metric was collected. The target name uniquely identifies a managed target within the Management Repository. The target name typically contains the name of the managed entity that was provided by the system or database administrator.
TARGET_TYPE	VARCHAR2(64)	The target type defines the set of metrics that are applicable for the target.
TARGET_GUID	RAW(16)	The unique global identifier for the target.
METRIC_NAME	VARCHAR2(64)	Name of the metric being defined.
METRIC_COLUMN	VARCHAR2(64)	For table metrics, the metric column contains the name of the column in the table that is being defined. If the metric that is being defined is not a table metric, the value in this column is a single space. For example, if this table describing the MGMT$TARGET_TYPE view was being defined as a table metric, Column Name, Data Type, and Description would be metric columns.
METRIC_LABEL	VARCHAR2(64)	An intuitive display name for the metric that is being defined.

Table 9–17 (Cont.) MGMT$METRIC_DAILY

Column	Datatype	Description
COLUMN_LABEL	VARCHAR2(64)	For table metrics, the column label contains an intuitive display name for the metric column.
KEY_VALUE	VARCHAR2(256)	The key value for which the alert has been recorded. For composite keys, this is the first part of the key.
KEY_VALUE2	VARCHAR2(256)	For composite keys, this is the second part of the key for which the alert has been recorded
KEY_VALUE3	VARCHAR2(256)	For composite keys, this is the third part of the key for which the alert has been recorded
KEY_VALUE4	VARCHAR2(256)	For composite keys, this is the fourth part of the key for which the alert has been recorded
KEY_VALUE5	VARCHAR2(256)	For composite keys, this is the fifth part of the key for which the alert has been recorded
ROLLUP_TIMESTAMP	DATE	The rollup timestamp identifies the start of the rollup period. For the one-hour rollups, samples that fall within the hourly boundaries from minute 00 through minute 59 inclusive will be combined. For example, samples from 12:00 AM through 12:59 AM would be combined into a single aggregated record with a rollup timestamp of "date" 12:00 AM.
SAMPLE_COUNT	NUMBER	The number of non-NULL samples for the metric that were aggregated.
AVERAGE	NUMBER	The average of the metric values for the samples that have been included in the rollup period.
MINIMUM	NUMBER	The minimum value for the metric for the samples that have been included in the rollup period.
MAXIMUM	NUMBER	The maximum value for the metric for samples that have been included in the rollup period.
STANDARD_DEVIATION	NUMBER	The standard deviation for the metric values that have been included in the rollup period.

Usage Notes

View provides the best granularity to show changes in a metric's value over the course of a week or month.

Understand trends in metric values.

Queries using this view will use an index if the queries use the target_name, the metric_name, or if they are based upon the rollup_timestamp.

9.3 Inventory Views

9.3.1 MGMT$TARGET

MGMT$TARGET displays information about the managed targets that are known to the Management Repository. These targets may or may not be actively monitored.

Table 9–18 MGMT$TARGET

Column	Datatype	Description
TARGET_NAME	VARCHAR2(64)	Name of the target where the metric was collected. The target name uniquely identifies a managed target within the Management Repository. The target name typically contains the name of the managed entity that was provided by the system or database administrator.
TARGET_TYPE	VARCHAR2(64)	The type of the target. Types of targets may include databases, hosts, web servers, applications, or Application Servers. The definer of the collection definition at the Management Agent defines the target type. The target type defines the set of metrics that are collected for a managed target within the Management Repository.
TARGET_GUID	RAW(16)	The unique global identifier for the target.
TYPE_VERSION	VARCHAR2(8)	The target type meta version of the metadata set. Metadata versions may be updated when applying patches or upon new releases of Enterprise Manager Grid Control.
TYPE_QUALIFIER1-5	VARCHAR2(64)	Up to five qualifiers can be used to distinguish different metric definitions based on different system configurations. Example qualifier entries may include operating system version, database version, or RAC configuration.
EMD_URL	VARCHAR2(2000)	The URL address of the Management Agent that is managing the target.
TIMEZONE_REGION	VARCHAR2(64)	The timezone region in which the target operates.
DISPLAY_NAME	VARCHAR2(64)	User-friendly name for the target.
HOST_NAME	VARCHAR2(128)	Name of the host where the target is running. For composite targets or targets that span a host, this column will be NULL.
LAST_METRIC_LOAD_TIME	TIMESTAMP	Timestamp when information for this target was last loaded into the Management Repository. If metrics have not been loaded into the Management Repository for the target, this column will be NULL.
TYPE_DISPLAY_NAME	VARCHAR2(128)	User-friendly name of the target type.

Usage Notes

Display a list of the targets known to the Management Repository.

Display administration and monitoring information in the context of a managed target.

Order the targets by last load time for customers to get a sense on how recent the information is for a target in the Management Repository. To access this information in an ordered way, customers should use the appropriate ORDER BY clause with the view.

Access to this view will use an index if the query references the target name and target type.

There is an implicit assumption that customers will not use this view to identify the targets that are owned by a Management Agent or the targets that reside on a specific host.

9.3.2 MGMT$TARGET_TYPE

MGMT$TARGET_TYPE displays metric descriptions for a given target name and target type. This information is available for the metrics for the managed targets that have been loaded into the Management Repository. Metrics are specific to the target type.

Table 9–19 MGMT$TARGET_TYPE

Column	Datatype	Description
TARGET_NAME	VARCHAR2(64)	Name of the target where the metric was collected. The target name uniquely identifies a managed target within the Management Repository. The target name typically contains the name of the managed entity that was provided by the system or database administrator.
TARGET_TYPE	VARCHAR2(64)	The target type defines the set of metrics that are applicable for the target.
TARGET_GUID	RAW(16)	The unique global identifier for the target.
TYPE_VERSION	VARCHAR2(8)	The target type meta version of the metadata set. Metadata versions may be updated when applying patches or upon new releases of Enterprise Manager Grid Control.
TYPE_QUALIFIER1-5	VARCHAR2(64)	Up to five qualifiers can be used to distinguish different metric definitions based on different system configurations. Example qualifier entries may include operating system version, database version, or RAC configuration.
METRIC_NAME	VARCHAR2(64)	The name of the metric that is being defined.
METRIC_COLUMN	VARCHAR2(64)	For table metrics, the metric column contains the name of the column in the table that is being defined. If the metric that is being defined is not a table metric, the value in this column is a single space. For example, if a table describing the MGMT$TARGET_TYPE view is to be defined as a table metric, Column Name, Data Type, and Description would be metric columns.
KEY_COLUMN	VARCHAR2(64)	For table metrics, the key column contains the name of the column in the table that represents the primary key. Values in this column must uniquely identify rows in the table. If the metric that is being defined is not a table metric, the value in this column is a single space; For example, the Column Name would be the key column if this table describing the MGMT$TARGET_TYPE view was being defined as a table metric.
METRIC_TYPE	VARCHAR2(20)	A DECODE of the internal numeric type of the metric that is being defined. This column will contain one of the following values: Number String Table Raw External Repository Metric
METRIC_LABEL	VARCHAR2(64)	A intuitive display name for the metric that is being defined.

Table 9–19 (Cont.) MGMT$TARGET_TYPE

Column	Datatype	Description
COLUMN_LABEL	VARCHAR2(64)	For table metrics, the column label contains a user understandable display name for the metric column.
DESCRIPTION	VARCHAR2(128)	A description of the metric that is being defined.
DESCRIPTION_NLSID	VARCHAR2(64)	The NLSid of the description of the metric
UNIT	VARCHAR2(32)	The unit of the metric that is being defined.
UNIT_NLSID	VARCHAR2(64)	The NLSid of the unit of the metric being defined.
SHORT_NAME	VARCHAR2(40)	This is a shortened version of the metric display name for the "dense" UI concept
SHORT_NAME_NLSID	VARCHAR2(64)	The NLSid of the short name of the metric being defined.

Usage Notes

List the set of metrics that have been defined for a target type.

Display intuitive metric names and associated attributes such as unit in a general way during portal, web application, or custom 4GL report generation.

Access to this view will use an index if the query references the metric name, metric column. The query should also qualify the target name and target type in order to restrict the amount of information returned.

9.3.3 MGMT$TARGET_TYPE_DEF

MGMT$TARGET_TYPE_DEF displays definition information for a target type.

Table 9–20 MGMT$TARGET_TYPE_DEF

Column	Datatype	Description
TARGET_TYPE	VARCHAR2(64)	The target type defines the set of metrics that are applicable for the target.
TYPE_DISPLAY_NAME	VARCHAR2(128)	User-friendly name of the target type.
TARGET_TYPE_GUID	RAW(16)	The unique global identifier (GUID) of the target type .
MAX_TYPE_META_VER	VARCHAR2(8)	The maximum version of the target type stored in the Management Repository.

9.3.4 MGMT$TARGET_ASSOCIATIONS

MGMT$TARGET_ASSOCIATIONS displays the various associations between targets. This view can be used to find all types of associations for a given target.

Table 9–21 MGMT$TARGET_ASSOCIATIONS

Column	Datatype	Description
ASSOC_DEF_NAME	VARCHAR2(64)	Name of the association definition.
SOURCE_TARGET_NAME	VARCHAR2(256)	Target name of the target to which the association is being defined.
SOURCE_TARGET_TYPE	VARCHAR2(64)	The target type of the target for which the association is being defined. “ANY” can be used to specify that any target type be used.

Table 9–21 (Cont.) MGMT$TARGET_ASSOCIATIONS

Column	Datatype	Description
ASSOC_TARGET_NAME	VARCHAR2(256)	Target Name of the target which is being associated with the source target.
ASSOC_TARGET_TYPE	VARCHAR2(64)	The target type of the associated target. "ANY" can be used to specify that any target type be used.
SCOPE_TARGET_NAME	VARCHAR2(256)	The target under whose scope the association is valid. This applies to non-global associations only. For example: A database may be part of a composite target only for a particular service.
SCOPE_TARGET_TYPE	VARCHAR2(64)	The target type for which the association is valid. This applies to non-global associations only.
ASSOCIATION_TYPE	VARCHAR2(64)	The type of association.

Usage Notes

Can be used to list the associations defined for a specific target.

Queries using this view will use an index if either (source_target_name, source_target_type) or (assoc_target_name, assoc_target_type) is specified.

9.3.5 MGMT$TARGET_MEMBERS

MGMT$TARGET_MEMBERS displays the list of direct members for a target.

Table 9–22 MGMT$TARGET_MEMBERS

Column	Datatype	Description
AGGREGATE_TARGET_NAME	VARCHAR2(256)	Target name of the aggregate target.
AGGREGATE_TARGET_TYPE	VARCHAR2(64)	Target type of the aggregate target.
AGGREGATE_TARGET_GUID	RAW(16)	Target GUID of the aggregate target.
MEMBER_TARGET_NAME	VARCHAR2(256)	Target Name of the member target.
MEMBER_TARGET_TYPE	VARCHAR2(64)	Target type of the member target.
MEMBER_TARGET_GUID	RAW(16)	Target GUID of the member target.

Usage Notes

Find the members for a aggregate target.

Find the aggregate targets for which a given target is a direct member.

Queries, which specify values for (AGGREGATE_TARGET_NAME, AGGREGATE_TARGET_TYPE) or (MEMBER_TARGET_NAME, MEMBER_TARGET_TYPE) will use index.

Joins using AGGREGATE_TARGET_GUID and MEMBER_TARGET_GUID will be efficient.

9.3.6 MGMT$TARGET_FLAT_MEMBERS

MGMT$TARGET_FLAT_MEMBERS displays the list of all direct and indirect members of the target.

Table 9–23 MGMT$TARGET_FLAT_MEMBERS

Column	Datatype	Description
AGGREGATE_TARGET_NAME	VARCHAR2(256)	The target name of the aggregate target
AGGREGATE_TARGET_TYPE	VARCHAR2(64)	The target type of the aggregate target
AGGREGATE_TARGET_GUID	RAW(16)	Target GUID of the aggregate target
MEMBER_TARGET_NAME	VARCHAR2(256)	Target Name of the member target
MEMBER_TARGET_TYPE	VARCHAR2(64)	Target type of the member target
MEMBER_TARGET_GUID	RAW(16)	Target GUID of the member target

Usage Notes

Find the members for a aggregate target.

Find the aggregate targets for which a given target is a member either directly or indirectly.

Queries, which specify values for (AGGREGATE_TARGET_NAME, AGGREGATE_TARGET_TYPE) or (MEMBER_TARGET_NAME, MEMBER_TARGET_TYPE) will use index.

Joins using AGGREGATE_TARGET_GUID and MEMBER_TARGET_GUID will be the most efficient on this view.

9.3.7 MGMT$TARGET_TYPE_PROPERTIES

MGMT$TARGET_TYPE_PROPERTIES displays the default list of properties that are applicable to the target based on the target type to which the target belongs.

Table 9–24 MGMT$TARGET_TYPE_PROPERTIES

Column	Datatype	Description
TARGET_NAME	VARCHAR2(256)	Name of the target.
TARGET_TYPE	VARCHAR2(64)	Name of the target type.
PROPERTY_NAME	VARCHAR2(64)	Name of the property, Example:is_aggregate,is_service, IsBaselineable.
PROPERTY_VALUE	VARCHAR2(1024)	Value of the property

Usage Notes

List the properties applicable to the target and the default values.

9.3.8 MGMT$TARGET_PROPERTIES

MGMT$TARGET_PROPERTIES displays detailed target properties.

Table 9–25 MGMT$

Column	Datatype	Description
TARGET_NAME	VARCHAR2(256)	Name of the target where the metric was collected. The target name uniquely identifies a managed target within the Management Repository. The target name typically contains the name of the managed entity that was provided by the system or database administrator.

Table 9–25 (Cont.) MGMT$

Column	Datatype	Description
TARGET_TYPE	VARCHAR2(64)	The target type defines the set of metrics that are applicable for the target.
TARGET_GUID	RAW(16)	The unique global identifier for the target.
PROPERTY_NAME	VARCHAR2(64)	The name of the target property being defined.
PROPERTY_VALUE	VARCHAR2(1024)	The value of the target property being defined.
PROPERTY_TYPE	VARCHAR2(64)	The type of the target property being defined. Possible values are: INSTANCE, if the property is applicable to the target instance. DYNAMIC, if the property is calculated dynamically.

9.3.9 MGMT$METRIC_CATEGORIES

MGMT$METRIC_CATEGORIES displays the list of classes and categories to which the metric belongs. This view can be used to classify the metric based on the class (Example: Service, Functional) and category within the class (Example: security/configuration under functional class or usage/performance under service class).

Table 9–26 MGMT$METRIC_CATEGORIES

Column	Datatype	Description
TARGET_TYPE	VARCHAR2(64)	Name of the target type.
TYPE_VERSION	VARCHAR2(8)	Version of the target type.
METRIC_NAME	VARCHAR2(64)	Name of the metric.
METRIC_COLUMN	VARCHAR2(64)	For table metrics, the metric column contains the name of the column in the table that is being defined. If the metric that is being defined is not a table metric, the value in this column is a single space.
METRIC_CLASS_NAME	VARCHAR2(64)	Name of the metric class to which this metric belongs.
METRIC_CATEGORY_NAME	VARCHAR2(64)	Name of the category of the metric in the class.
METRIC_CATEGORY_NLSID	VARCHAR2(64)	The NLS ID of the category which is used by Enterprise Manager to translate the name to different languages.

Usage Notes

Classify the metrics into different buckets based on Class and category.

9.4 Policy Definition Views

9.4.1 MGMT$POLICIES

MGMT$POLICIES displays the list of policies defined in the Management Repository.

Table 9–27 MGMT$POLICIES

Column	Datatype	Description
TARGET_TYPE	VARCHAR2(64)	The target type defines the set of metrics and polciies that are applicable for the target.
METRIC_NAME	VARCHAR2(64)	The name of the metric based on which the policy is defined.
POLICY_NAME	VARCHAR2(128)	The name of the policy being defined.
POLICY_LABEL_NLSID	VARCHAR2(64)	The NSL ID for the policy name.
POLICY_GUID	RAW(16)	The unique global identifier for the policy.
AUTHOR	VARCHAR2(256)	The author of the policy.
DESCRIPTION	VARCHAR2(256)	The description of the policy.
DESCRIPTION_NLSID	VARCHAR2(64)	The NLS ID of the description.
IMPACT	VARCHAR2(500)	A description of th e impact of the policy being defined.
IMPACT_NLSID	VARCHAR2(64)	NLSid for the policy impact.
RECOMMENDATION	VARCHAR2(500)	A recommendation for the policy being defined.
RECOMMENDATION_NLSID	VARCHAR2(64)	NLS ID for the policy recommendataion.
VIOLATION_LEVEL	VARCHAR2(13)	The level at which the violations will be logged when the policy being defined is violated. Possible values are : Informational Warning Critical
CONDITION_TYPE	NUMBER(1)	The type condition used by the the policy being defined. Possible values are : 1. Simple condition 2. SQL expression condition 3. PL/SQL procedure for evaluating condition.
CONDITION	VARCHAR2(4000)	The conidtion details / expression used by the policy being defined. If the condition type is 1, this column contains the metric_column_ name 2, this column contains the conditional expression 3, this column contains the pl/sql procedure name that peforms the condition.
CONDITION_OPERATOR	NUMBER	Defines the policy condition operator to be applied, when the condition_type is 'Simple Condition'. Possible values are : 0 - GT 1 - EQ 2 - LT 3 - LE 4 - GE 5 - CONTAINS 6 - NE 7 - MATCH : regular expression
OWNER	VARCHAR2(256)	The Enterprise Manager administrator who created the policy in the repository.

Table 9–27 (Cont.) MGMT$POLICIES

Column	Datatype	Description
AUTO_ENABLED	NUMBER(1)	Specifies whether the policy should be enable automatically on the new targets that are being added.
CATEGORY	VARCHAR2(64)	The name of the category the policy. Refer to MGMT$METRIC_CATEGORIES for the list of all categories.
CATEGORY_NLSID	VARCHAR2(64)	The NLSid of the policy category.

9.4.2 MGMT$POLICY_PARAMETERS

MGMT$POLICY_PARAMETERS displays detailed information about a policy.

Table 9–28 MGMT$POLICY_PARAMETERS

Column	Datatype	Description
TARGET_TYPE	VARCHAR2(64)	The target type defines the set of metrics and polciies that are applicable for the target.
METRIC_NAME	VARCHAR2(64)	The name of the metric based on which the policy is defined.
POLICY_NAME	VARCHAR2(128)	The name of the policy being defined.
POLICY_GUID	RAW(16)	The unique global identifier for the policy.
PARAMETER_NAME	VARCHAR2(64)	The name of the policy parameter
PARAMETER_NAME_NLSID	VARCHAR2(64)	The NLSid of the policy parameter.
PARAMETER_TYPE	VARCHAR2(7)	The type of the policy parameter. Possible values are : 1 for NUMERIC parameter 2 for STRING parameter

9.4.3 MGMT$POLICY_VIOLATION_CTXT

MGMT$POLICY_VIOLATION_CTXT displays the policy violation context.

Table 9–29 MGMT$POLICY_VIOLATION_CTXT

Column	Datatype	Description
TARGET_TYPE	VARCHAR2(64)	The target type defines the set of metrics and polciies that are applicable for the target
METRIC_NAME	VARCHAR2(64)	The name of the metric based on which the policy is defined.
POLICY_NAME	VARCHAR2(128)	The name of the policy being defined.
POLICY_GUID	RAW(16)	The unique global identifier for the policy.
COLUMN_NAME	VARCHAR2(64)	The column name to be included in the policy violation context.
IS_HIDDEN	NUMBER(1)	Flag indicating whether or not the value for this metric (data source) column should be displayed when viewing a violation. Default 0.
URL_LINK_TYPE	VARCHAR2(7)	Flag indicating the type of the link. Possible values : JSP UIX

Table 9–29 (Cont.) MGMT$POLICY_VIOLATION_CTXT

Column	Datatype	Description
URL_LINK_TEMPLATE	VARCHAR2(4000)	URL template that contains references to any metric (data source) column name defined in the metric (data source). The actual values are substituted in at run-time to construct a URL that is associated with this metric columns value. Default NULL.

9.4.4 MGMT$TARGET_POLICY_EVAL_SUMM

MGMT$TARGET_POLICY_EVAL_SUMM displays the policy violation summary for a target.

Table 9–30 MGMT$TARGET_POLICY_EVAL_SUMM

Column	Datatype	Description
TARGET_NAME	VARCHAR2(256)	Name of the target for which the policies were being evaluated. The target name uniquely identifies a managed target within the Management Repository. The target name typically contains the name of the managed entity that was provided by the system or database administrator.
TARGET_TYPE	VARCHAR2(64)	The target type defines the set of metrics and policies that are applicable for the target
TARGET_GUID	RAW(16)	The unique global identifier for the target..
POLICY_NAME	VARCHAR2(128)	The name of the policy for which evaluation summary is provided.
POLICY_GUID	RAW(16)	The unique global identifier for the policy.
COLL_NAME	VARCHAR2(64)	The name of the collection on which the policy is evaluated.
LAST_EVALUATION_DATE	DATE	The last date on which the policy was evaluated for the target for the collection item.
TOTAL_VIOLATIONS	NUMBER	The number of violations logged for the given target policy association
NON_SUPPRESS_VIOLATIONS	NUMBER	The number of non-exempt violations logged for the given target policy association.
COMPLIANCE_SCORE	NUMBER	The compliance score calculated for the target policy association.

9.4.5 MGMT$POLICY_VIOL_ANNOTATIONS

MGMT$POLICY_VIOL_ANNOTATIONS displays a summary of policy violation annotations.

Table 9–31 MGMT$POLICY_VIOL_ANNOTATIONS

Column	Datatype	Description
TARGET_NAME	VARCHAR2(256)	Name of the target where the metric was collected. The target name uniquely identifies a managed target within the Management Repository. The target name typically contains the name of the managed entity that was provided by the system or database administrator.
TARGET_TYPE	VARCHAR2(64)	The target type defines the set of metrics that are applicable for the target.

Table 9–31 (Cont.) MGMT$POLICY_VIOL_ANNOTATIONS

Column	Datatype	Description
TARGET_GUID	RAW(16)	The unique global identifier for the target.
POLICY_NAME	VARCHAR2(128)	The name of the policy rule
DESCRIPTION	VARCHAR2(256)	A description of the policy rule
KEY_VALUE	VARCHAR2(256)	The key value for which the alert has been recorded. For composite keys, this is the first part of the key.
KEY_VALUE2	VARCHAR2(256)	For composite keys, this is the second part of the key for which the alert has been recorded
KEY_VALUE3	VARCHAR2(256)	For composite keys, this is the third part of the key for which the alert has been recorded
KEY_VALUE4	VARCHAR2(256)	For composite keys, this is the fourth part of the key for which the alert has been recorded
KEY_VALUE5	VARCHAR2(256)	For composite keys, this is the fifth part of the key for which the alert has been recorded
MESSAGE	VARCHAR2(4000)	The policy violation message.
VIOLATION_LEVEL	VARCHAR2(40)	A user readable description of the internal violation code that is sent from the Management Agent to identify the state of the policy violation. A policy violation record is transferred to the repository from the Management Agent each time the metric threshold is crossed in either direction, or if the Management Agent is restarted. The value of this column will contain one of the following strings: · Clear · Critical
COLLECTION_TIMESTAMP	DATE	The date-time when the policy violation was detected by the Management Agent.
ANNOTATION_MESSAGE	VARCHAR2(4000)	The annotation text.
ANNOTATION_TIMESTAMP	DATE	The time the annotation was created.
ANNOTATED_BY	VARCHAR2(256)	Enterprise Manager administrator who authored the the annotation.

9.4.6 MGMT$POLICY_VIOL_NOTIF_LOG

MGMT$POLICY_VIOL_NOTIF_LOG displays details of notification deliveries for policy violations.

Table 9–32 MGMT$POLICY_VIOL_NOTIF_LOG

Column	Datatype	Description
TARGET_NAME	VARCHAR2(256)	Name of the target where the metric was collected. The target name uniquely identifies a managed target within the Management Repository. The target name typically contains the name of the managed entity that was provided by the system or database administrator.
TARGET_TYPE	VARCHAR2(64)	The target type defines the set of metrics that are applicable for the target.
TARGET_GUID	RAW(16)	The unique global identifier for the target.
POLICY_NAME	VARCHAR2(128)	The name of the policy rule

Table 9–32 (Cont.) MGMT$POLICY_VIOL_NOTIF_LOG

Column	Datatype	Description
KEY_VALUE	VARCHAR2(256)	The key value for which the alert has been recorded. For composite keys, this is the first part of the key.
KEY_VALUE2	VARCHAR2(256)	For composite keys, this is the second part of the key for which the alert has been recorded
KEY_VALUE3	VARCHAR2(256)	For composite keys, this is the third part of the key for which the alert has been recorded
KEY_VALUE4	VARCHAR2(256)	For composite keys, this is the fourth part of the key for which the alert has been recorded
KEY_VALUE5	VARCHAR2(256)	For composite keys, this is the fifth part of the key for which the alert has been recorded
VIOLATION_MESSAGE	VARCHAR2(4000)	A user readable description of the internal violation code that is sent from the Management Agent to identify the state of the policy violation. A policy violation record is transferred to the repository from the Management Agent each time the metric threshold is crossed in either direction, or if the Management Agent is restarted. The value of this column will contain one of the following strings: · Clear · Critical
VIOLATION_LEVEL	VARCHAR2(40)	A user readable description of the internal violation code that is sent from the Management Agent to identify the state of the policy violation. A policy violation record is transferred to the repository from the Management Agent each time the metric threshold is crossed in either direction, or if the Management Agent is restarted. The value of this column will contain one of the following strings: · Clear · Critical
COLLECTION_TIMESTAMP	DATE	The date-time when the policy violation was detected by the Management Agent.
DELIVERY_MESSAGE	VARCHAR2(1024)	The message indicating the success or failure of the notification delivery.
DELIVERY_TIMESTAMP	DATE	The date-time when the notification was delivered.

9.5 Policy Association Views

9.5.1 MGMT$TARGET_POLICIES

MGMT$TARGET_POLICIES displays target policy information.

Table 9–33 MGMT$TARGET_POLICIES

Column	Datatype	Description
TARGET_NAME	VARCHAR2(256)	Name of the target for which the policies were being evaluated. The target name uniquely identifies a managed target within the Management Repository. The target name typically contains the name of the managed entity that was provided by the system or database administrator.
TARGET_TYPE	VARCHAR2(64)	The target type defines the set of metrics and policies that are applicable for the target
TARGET_GUID	RAW(16)	The unique global identifier for the target.
POLICY_NAME	VARCHAR2(128)	The name of the policy that is associated with the target.
POLICY_GUID	RAW(16)	The unique global identifier for the policy.
CATEGORY	VARCHAR2(64)	The name of the category the policy. Refer to MGMT$METRIC_CATEGORIES for the list of all categories.
IS_ENABLED	NUMBER(1)	Flag to indicate whether the policy is enabled or disabled on the target.

9.5.2 MGMT$TARGET_POLICY_SETTINGS

MGMT$TARGET_POLICY_SETTINGS displays policy settings for a target.

Table 9–34 MGMT$TARGET_POLICY_SETTINGS

Column	Datatype	Description
TARGET_NAME	VARCHAR2(256)	Name of the target for which the policies were being evaluated. The target name uniquely identifies a managed target within the Management Repository. The target name typically contains the name of the managed entity that was provided by the system or database administrator.
TARGET_TYPE	VARCHAR2(64)	The target type defines the set of metrics and policies that are applicable for the target
TARGET_GUID	RAW(16)	The unique global identifier for the target.
POLICY_NAME	VARCHAR2(128)	The name of the policy that is associated with the target.
POLICY_GUID	RAW(16)	The unique global identifier for the policy.
CATEGORY	VARCHAR2(64)	The name of the category the policy. Refer to MGMT$METRIC_CATEGORIES for the list of all categories.
KEY_VALUE	VARCHAR2(256)	The key value for which the alert has been recorded. For composite keys, this is the first part of the key.
KEY_VALUE2	VARCHAR2(256)	For composite keys, this is the second part of the key for which the alert has been recorded
KEY_VALUE3	VARCHAR2(256)	For composite keys, this is the third part of the key for which the alert has been recorded
KEY_VALUE4	VARCHAR2(256)	For composite keys, this is the fourth part of the key for which the alert has been recorded
KEY_VALUE5	VARCHAR2(256)	For composite keys, this is the fifth part of the key for which the alert has been recorded

Table 9–34 (Cont.) MGMT$TARGET_POLICY_SETTINGS

Column	Datatype	Description
KEY_OPERATOR	NUMBER	Specifies whether the key_value columns have any SQL wildcards. For single key column metrics, the value is 1 if the key_value has wildcard characters, 0 otherwise. For metrics with multiple keys, a list of operators for all key columns will be stored here. For example, a metric with 3 keys (k1, k2, k3) where K1 and K2 use wildcards and K3 uses exact match, then 011 is stored in this column
PARAMETER_NAME	VARCHAR2(64)	The name of the policy parameter.
PREVENT_OVERRIDE	NUMBER(1)	The is a flag that specifies that the metric rows with this key_value has a tempalte override flag. Once the tempalte override flag is ON, any template application will not update the threshold/parameter values.
POLICY_THRESHOLD	VARCHAR2(256)	The threshold value for the policy parameter.
ACTION_TYPE	VARCHAR2(17)	The corrective action type configured. Possible values are : No-Action : when no action is configured Corrective-Action : when a repository side corrective action is configured Agent-Fixit Job : when an Agent side fix-it job is configured.
ACTION_JOB_TYPE	VARCHAR2(32)	Specifies the job type of the corrective action when ACTION_TYPE is “Corrective-Action”.
ACTION_JOB_NAME	VARCHAR2(64)	Specifies the job name of the corrective action when ACTION_TYPE is “Corrective-Action”.
ACTION_JOB_OWNER	VARCHAR2(256)	Specifies the job owner of the corrective action when ACTION_TYPR is “Corrective-Action”.

9.6 Policy Violation Views

9.6.1 MGMT$POLICY_VIOLATION_CURRENT

MGMT$POLICY_VIOLATION_CURRENT displays current information for any policy violations that are logged in the Management Repository that are in a non-clear state. Only the most recent open violation in a non-clear status for a given policy will be provided through this view.

Table 9–35 MGMT$POLICY_VIOLATION_CURRENT

Column	Datatype	Description
TARGET_NAME	VARCHAR2(256)	Name of the target for which the policies were being evaluated. The target name uniquely identifies a managed target within the Management Repository. The target name typically contains the name of the managed entity that was provided by the system or database administrator.
TARGET_TYPE	VARCHAR2(64)	The target type defines the set of metrics and policies that are applicable for the target
TYPE_DISPLAY_NAME	VARCHAR2(128)	User-friendly name of the target type.

Table 9–35 (Cont.) MGMT$POLICY_VIOLATION_CURRENT

Column	Datatype	Description
TARGET_GUID	RAW(16)	The unique global identifier for the target.
POLICY_NAME	VARCHAR2(128)	The name of the policy that is associated with the target.
POLICY_GUID	RAW(16)	The unique global identifier for the policy.
CATEGORY	VARCHAR2(64)	The name of the category the policy. Refer to MGMT$METRIC_CATEGORIES for the list of all categories.
KEY_VALUE	VARCHAR2(256)	The key value for which the alert has been recorded. For composite keys, this is the first part of the key.
KEY_VALUE2	VARCHAR2(256)	For composite keys, this is the second part of the key for which the alert has been recorded
KEY_VALUE3	VARCHAR2(256)	For composite keys, this is the third part of the key for which the alert has been recorded
KEY_VALUE4	VARCHAR2(256)	For composite keys, this is the fourth part of the key for which the alert has been recorded
KEY_VALUE5	VARCHAR2(256)	For composite keys, this is the fifth part of the key for which the alert has been recorded
COLLECTION_TIMESTAMP	DATE	The date-time when the policy violation condition was detected.
VIOLATION_LEVEL	NUMBER	A user readable description of the internal violation level logged to identify the state of the policy condition. A violation record is logged each time the policy condition is violated in either direction. The value of this column will contain one of the following strings: . Informational · Warning · Critical If the policy condition goes into a clear state, it will no longer be visible from this view.
MESSAGE	VARCHAR2(4000)	An optional message that is generated when the violation is logged that provides additional information about the policy condition.
MESSAGE_NLSID	VARCHAR2(64)	The NLSid of the message.
MESSAGE_PARAMS	VARCHAR2(4000)	Contains the URL encoded parameters separated by "&" to be used to format the alert message.
SUPPRESS_CODE	NUMBER	Indication of exemption status. Legal values: 0 - not exempt 1 - exempt indefinitely 2 - exempt until date specified in SUPPRESS_UNTIL 3 - exempt until next evaluation Default 0'
SUPPRESS_UNTIL	DATE	Specifies the date until which exemption should be applied. Only application if SUPPRESS_ CODE is 2
SUPPRESS_BY	VARCHAR2(256)	Enterprise Manager adminstrator that suppressed the violation.

9.6.2 MGMT$POLICY_VIOLATION_HISTORY

MGMT$POLICY_VIOLATION_HISTORY displays historical information for any policies that are logged in the Management Repository.

Table 9–36 MGMT$POLICY_VIOLATION_HISTORY

Column	Datatype	Description
TARGET_NAME	VARCHAR2(256)	Name of the target for which the policies were being evaluated. The target name uniquely identifies a managed target within the Management Repository. The target name typically contains the name of the managed entity that was provided by the system or database administrator.
TARGET_TYPE	VARCHAR2(64)	The target type defines the set of metrics and policies that are applicable for the target
TYPE_DISPLAY_NAME	VARCHAR2(128)	User-friendly name of the target type.
TARGET_GUID	RAW(16)	The unique global identifier for the target.
POLICY_NAME	VARCHAR2(128)	The name of the policy that is associated with the target.
POLICY_GUID	RAW(16)	The unique global identifier for the policy.
CATEGORY	VARCHAR2(64)	The name of the category the policy. Refer to MGMT$METRIC_CATEGORIES for the list of all categories.
KEY_VALUE	VARCHAR2(256)	The key value for which the alert has been recorded. For composite keys, this is the first part of the key.
KEY_VALUE2	VARCHAR2(256)	For composite keys, this is the second part of the key for which the alert has been recorded
KEY_VALUE3	VARCHAR2(256)	For composite keys, this is the third part of the key for which the alert has been recorded
KEY_VALUE4	VARCHAR2(256)	For composite keys, this is the fourth part of the key for which the alert has been recorded
KEY_VALUE5	VARCHAR2(256)	For composite keys, this is the fifth part of the key for which the alert has been recorded
COLLECTION_TIMESTAMP	DATE	The date-time when the policy violation condition was detected.
VIOLATION_LEVEL	NUMBER	A user readable description of the internal violation level logged to identify the state of the policy condition. A violation record is logged each time the policy condition is violated in either direction. The value of this column will contain one of the following strings: . Informational · Warning · Critical If the policy condition goes into a clear state, it will no longer be visible from this view.
VIOLATION_DURATION	NUMBER	The time, in hours, from when the policy condition was first detected until it was cleared.
MESSAGE	VARCHAR2(4000)	An optional message that is generated when the violation is logged that provides additional information about the policy condition.
MESSAGE_NLSID	VARCHAR2(64)	NLSid of the policy message.

Table 9–36 (Cont.) MGMT$POLICY_VIOLATION_HISTORY

Column	Datatype	Description
MESSAGE_PARAMS	VARCHAR2(4000)	Contains the URL encoded parameters separated by "&" to be used to format the policy message.

9.6.3 MGMT$POLICY_VIOLATION_CONTEXT

MGMT$POLICY_VIOLATION_CONTEXT displays policy violation context.

Table 9–37 MGMT$POLICY_VIOLATION_CONTEXT

Column	Datatype	Description
TARGET_NAME	VARCHAR2(256)	Name of the target for which the policies were being evaluated. The target name uniquely identifies a managed target within the Management Repository. The target name typically contains the name of the managed entity that was provided by the system or database administrator.
TARGET_TYPE	VARCHAR2(64)	The target type defines the set of metrics and policies that are applicable for the target
TYPE_DISPLAY_NAME	VARCHAR2(128)	User-friendly name of the target type.
TARGET_GUID	RAW(16)	The unique global identifier for the target.
POLICY_NAME	VARCHAR2(128)	The name of the policy that is associated with the target.
POLICY_GUID	RAW(16)	The unique global identifier for the policy.
CATEGORY	VARCHAR2(64)	The name of the category the policy. Refer to MGMT$METRIC_CATEGORIES for the list of all categories.
KEY_VALUE	VARCHAR2(256)	The key value for which the alert has been recorded. For composite keys, this is the first part of the key.
KEY_VALUE2	VARCHAR2(256)	For composite keys, this is the second part of the key for which the alert has been recorded
KEY_VALUE3	VARCHAR2(256)	For composite keys, this is the third part of the key for which the alert has been recorded
KEY_VALUE4	VARCHAR2(256)	For composite keys, this is the fourth part of the key for which the alert has been recorded
KEY_VALUE5	VARCHAR2(256)	For composite keys, this is the fifth part of the key for which the alert has been recorded
COLLECTION_TIMESTAMP	DATE	The date-time when the policy violation condition was detected.
COLUMN_NAME	VARCHAR2(64)	The associated name stored along with the policy violation.
COLUMN_VALUE	VARCHAR2(4000)	The associated value stored along with the policy violation.

9.7 Management Template Views

9.7.1 MGMT$TEMPLATES

MGMT$TEMPLATES displays details of all the management templates stored in the Management Repository.

Table 9–38 MGMT$TEMPLATES

Column	Datatype	Description
TARGET_TYPE	VARCHAR2(64)	The target type defines the set of metrics and policies that are applicable for the target
TEMPLATE_NAME	VARCHAR2(64)	The name of the template.
TEMPLATE_GUID	RAW(16)	The unique global identifier for the template.
DESCRIPTION	VARCHAR2(1024)	The description of the template.
OWNER	VARCHAR2(256)	Enterprise Manager administrator who owns the template.
IS_PUBLIC	NUMBER(1)	The flag to specify whether the template is accessible to all EM administrators.
CREATED_DATE	DATE	The date/time when the template is created in the repository.
LAST_UPDATED_DATE	DATE	The date/time when the template was last modified in the repository.
LAST_UPDATED_BY	VARCHAR2(256)	The EM administrator who last updated the template.

9.7.2 MGMT$TEMPLATE_POLICY_SETTINGS

MGMT$TEMPLATE_POLICY_SETTINGS displays policy settings for management templates.

Table 9–39 MGMT$TEMPLATE_POLICY_SETTINGS

Column	Datatype	Description
TARGET_TYPE	VARCHAR2(64)	The target type defines the set of metrics and policies that are applicable for the target
TEMPLATE_NAME	VARCHAR2(64)	The name of the template.
TEMPLATE_GUID	RAW(16)	The unique global identifier for the template.
POLICY_NAME	VARCHAR2(128)	The name of the policy that is associated with the template.
POLICY_GUID	RAW(16)	The unique global identifier for the policy.
CATEGORY	VARCHAR2(64)	The name of the category the policy. Refer to MGMT$METRIC_CATEGORIES for the list of all categories.
KEY_VALUE	VARCHAR2(256)	The key value for which the alert has been recorded. For composite keys, this is the first part of the key.
KEY_VALUE2	VARCHAR2(256)	For composite keys, this is the second part of the key for which the alert has been recorded
KEY_VALUE3	VARCHAR2(256)	For composite keys, this is the third part of the key for which the alert has been recorded
KEY_VALUE4	VARCHAR2(256)	For composite keys, this is the fourth part of the key for which the alert has been recorded
KEY_VALUE5	VARCHAR2(256)	For composite keys, this is the fifth part of the key for which the alert has been recorded

Table 9–39 (Cont.) MGMT$TEMPLATE_POLICY_SETTINGS

Column	Datatype	Description
KEY_OPERATOR	NUMBER	Specifies whether the key_value columns have any SQL wildcards. For single key column metrics, the value is 1 if the key_value has wildcard characters, 0 otherwise. For metrics with multiple keys, a list of operators for all key columns will be stored here. For example, a metric with 3 keys (k1, k2, k3) where K1 and K2 use wildcards and K3 uses exact match, then 011 is stored in this column
PARAMETER_NAME	VARCHAR2(64)	The name of the parameter.
PREVENT_OVERRIDE	NUMBER(1)	The is a flag that specifies that the metric rows with this key_value has a tempalte override flag. Once the tempalte override flag is ON, any template application will not update the threshold/parameter values.
POLICY_THRESHOLD	VARCHAR2(256)	The threshold value confugured for the policy parameter.
ACTION_TYPE	VARCHAR2(17)	The corrective action type configured. Possible values are : No-Action : when no action is configured Corrective-Action : when a repository side corrective action is configured Agent-Fixit Job : when an Agent side fix-it job is configured.
ACTION_JOB_TYPE	VARCHAR2(32)	Specifies the job type of the corrective action when ACTION_TYPE is “Corrective-Action”.
ACTION_JOB_NAME	VARCHAR2(64)	Specifies the job name of the corrective action when ACTION_TYPE is “Corrective-Action”.
ACTION_JOB_OWNER	VARCHAR2(256)	Specifies the job owner of the corrective action when ACTION_TYPE is “Corrective-Action”.

9.7.3 MGMT$TEMPLATE_METRICCOLLECTION

MGMT$TEMPLATE_METRICCOLLECTIONS displays information on the metric collections defined for a template.

Table 9–40 MGMT$TEMPLATE_METRICCOLLECTION

Column	Datatype	Description
TEMPLATE_NAME	VARCHAR2(64)	The name of the template.
TARGET_TYPE	VARCHAR2(64)	The target type defines the set of metrics and policies that are applicable for the target
TEMPLATE_GUID	RAW(16)	The unique global identifier for the template.
METRIC_NAME	VARCHAR2(64)	The name of the metric for which the template collection is configured.
METRIC_COLUMN	VARCHAR2(64)	The name of the metric column for which the template collection is configured.
METRIC_GUID	RAW(16)	The unique global identifier for the metric column.
COLLECTION_NAME	VARCHAR2(64)	The name of the collection.

Table 9–40 (Cont.) MGMT$TEMPLATE_METRICCOLLECTION

Column	Datatype	Description
IS_REPOSITORY	NUMBER(1)	Indicates whether this is a repository-side collection. A repository-side collection has a PL/SQL evaluation procedure that is responsible for calculating the metric values.
FREQUENCY_CODE	VARCHAR2(9)	The metric collection frequency type. Possible values are: 1: One Time 2: Interval 3: Daily 4: Weekly 5: Monthly 6: Yearly 7: On Demand
COLLECTION_FREQUENCY	VARCHAR2(81)	The frequency of the metric collection. Value displayed is dependent on the frequency code: For One Time, the start date-time is stored in DD-MON-YY HH24:MI format. For Interval type, the frequency in minutes is stored. For Daily/Weekly/Monthly/Yearly types, the hour and minute of collection is stored in HH24:MI format. For On-Demand type, On-Demand is stored.
UPLOAD_POLICY	NUMBER	The frequency with which the metric data is uploaded/stored.

9.7.4 MGMT$TEMPLATE_METRIC_SETTINGS

MGMT$TEMPLATE_METRIC_SETTINGS displays management template settings.

Table 9–41 MGMT$TEMPLATE_METRIC_SETTINGS

Column	Datatype	Description
TEMPLATE_NAME	VARCHAR2(64)	The name of the template.
TARGET_TYPE	VARCHAR2(64)	The target type defines the set of metrics and policies that are applicable for the target
TEMPLATE_GUID	RAW(16)	The unique global identifier for the template.
METRIC_NAME	VARCHAR2(64)	The name of the metric for which the template collection is configured.
METRIC_COLUMN	VARCHAR2(64)	The name of the metric column for which the template collection is configured.
METRIC_GUID	RAW(16)	The unique global identifier for the metric column.
COLLECTION_NAME	VARCHAR2(64)	The name of the collection.
CATEGORY	VARCHAR2(64)	The name of the category the policy. Refer to MGMT$METRIC_CATEGORIES for the list of all categories.
KEY_VALUE	VARCHAR2(256)	The key value for which the alert has been recorded. For composite keys, this is the first part of the key.
KEY_VALUE2	VARCHAR2(256)	For composite keys, this is the second part of the key for which the alert has been recorded

Table 9–41 (Cont.) MGMT$TEMPLATE_METRIC_SETTINGS

Column	Datatype	Description
KEY_VALUE3	VARCHAR2(256)	For composite keys, this is the third part of the key for which the alert has been recorded
KEY_VALUE4	VARCHAR2(256)	For composite keys, this is the fourth part of the key for which the alert has been recorded
KEY_VALUE5	VARCHAR2(256)	For composite keys, this is the fifth part of the key for which the alert has been recorded
KEY_OPERATOR	NUMBER	Specifies whether the key_value columns have any SQL wildcards. For single key column metrics, the value is 1 if the key_value has wildcard characters, 0 otherwise. For metrics with multiple keys, a list of operators for all key columns will be stored here. For example, a metric with 3 keys (k1, k2, k3) where K1 and K2 use wildcards and K3 uses exact match, then 011 is stored in this column
PREVENT_OVERRIDE	NUMBER(1)	The is a flag that specifies that the metric rows with this key_value has a tempalte override flag. Once the tempalte override flag is ON, any template application will not update the threshold/parameter values.
WARNING_OPERATOR	NUMBER	Defines the warning threshold condition to be applied 0 - GT 1 - EQ 2 - LT 3 - LE 4 - GE 5 - CONTAINS 6 - NE 7 - MATCH : regular expression
WARNING_THRESHOLD	VARCHAR2(256)	The value of the warning threshold.
CRITICAL_OPERATOR	NUMBER	Defines the critical threshold condition to be applied 0 - GT 1 - EQ 2 - LT 3 - LE 4 - GE 5 - CONTAINS 6 - NE 7 - MATCH : regular expression
CRITICAL_THRESHOLD	VARCHAR2(256)	The value of the critical threshold.
OCCURRENCE_COUNT	NUMBER	The number of times the test has to trigger to raise a violation.
WARNING_ACTION_TYPE	VARCHAR2(17)	The warning corrective action type configured. Possible values are : No-Action : when no action is configured Corrective-Action : when a repository side corrective action is configured Agent-Fixit Job : when an Agent side fix-it job is configured.

Table 9–41 (Cont.) MGMT$TEMPLATE_METRIC_SETTINGS

Column	Datatype	Description
WARNING_ACTION_JOB_TYPE	VARCHAR2(32)	Specifies the job type of the warning corrective action when WARNING_ACTION_TYPE is "Corrective-Action".
WARNING_ACTION_JOB_OWNER	VARCHAR2(256)	Specifies the job owner of the warning corrective action when WARNING_ACTION_TYPE is "Corrective-Action".
WARNING_ACTION_JOB_NAME	VARCHAR2(64)	Specifies the job name of the warning corrective action when WARNING_ACTION_TYPE is "Corrective-Action".
CRITICAL_ACTION_TYPE	VARCHAR2(17)	The critical corrective action type configured. Possible values are : No-Action : when no action is configured Corrective-Action : when a repository side corrective action is configured Agent-Fixit Job : when an
CRITICAL_ACTION_JOB_TYPE	VARCHAR2(32)	Specifies the job type of the critical corrective action when CRITICAL_ACTION_TYPE is "Corrective-Action".
CRITICAL_ACTION_JOB_OWNER	VARCHAR2(256)	Specifies the job owner of the critical corrective action when CRITICAL_ACTION_TYPE is "Corrective-Action".
CRITICAL_ACTION_JOB_NAME	VARCHAR2(64)	Specifies the job name of the critical corrective action when CRITICAL_ACTION_TYPE is "Corrective-Action".

9.8 Job Views

9.8.1 MGMT$JOBS

MGMT$JOBS displays information about a job including the job's schedule.

Table 9–42 MGMT$JOBS

Column	Datatype	Description
JOB_NAME	VARCHAR2(64)	The unique name for the job.
JOB_ID	RAW(16)	The uniqe system identifier for the job.
JOB_OWNER	VARCHAR2(256)	The owner/creator of the job.
JOB_DESCRIPTION	VARCHAR2(4000)	Optional text describing the job function.
JOB_TYPE	VARCHAR2(32)	The job type. For example, multi-task, SQL script or OS Command.
TARGET_TYPE	VARCHAR2()	The type of target the job was submitted against. Applies to single-target jobs only.
IS_LIBRARY	NUMBER(1)	Indicates whether or not the job is part of the job library.
IS_RESTARTABLE	NUMBER(1)	Indicates whether the job can be restarted. "0" indicates the job is not restartable. "1" indicates the job is not restartable. By default, a job is not restartable if the original job owner is deleted and the job is transferred to another administrator.

Table 9–42 (Cont.) MGMT$JOBS

Column	Datatype	Description
START_TIME	DATE	The scheduled start time. For daily, days of week and days of month schedules, the start_ time denotes when the job should start.
END_TIME	DATE	For all periodic schedules, the last date (and time) to run the job. For daily, day of week and day of month schedules, only the date portion is used.
TIMEZONE_TYPE		Possible values are 1 - Repository timezone 2- Target timezone 4 - Specified timezone region
TIMEZONE_REGION	VARCHAR2(64)	The specified timezone region.
SCHEDULE_TYPE		Possible values are: 0 - Immediate schedule 1 - Run once at specified time 2 - Run on interval 3 - Run daily 4 - Run on specified days of the week 5 - Run on specified days of the month 6 - Run on specified days of the year
INTERVAL	NUMBER	If schedule_type is interval (2) , this is the interval at which the job repeats, in minutes
EXECUTION_HOURS	NUMBER(3)	Indicates the time of day at which the job will execute. Hours are specified using the 24-hour format (0 to 23).
EXECUTION_MINUTES	NUMBER(3)	Indicates the time of day at which the job will execute. Minues are specified as a number between 0 and 59.
MONTHS	Integer Array	For days-of-year job schedules, this indicates the "month" in the schedule.
DAYS	Integer Array	For day-of-week/month or day(s) of the week job schedules, this iindicates the "day" of the week/month. Days-of-week specified as numbers 1 (Sunday) to 7 (Saturday). Days-of-month specified as numbers 1 to 31.

9.8.2 MGMT$JOB_TARGETS

MGMT$JOB_TARGETS displays the target(s) the job was submitted against.

Table 9–43 MGMT$JOB_TARGETS

Column	Datatype	Description
JOB_NAME	VARCHAR2(64)	The unique name for the job.
JOB_OWNER	VARCHAR2(256)	The owner/creator of the job.
JOB_TYPE	VARCHAR2(32)	The job type. For example, multi-task, SQL script or OS Command.
TARGET_NAME	VARCHAR2(64)	Name of the target the job was submitted against.
TARGET_TYPE	VARCHAR2()	The type of target the job was submitted against. Applies to single-target jobs only.

Table 9–43 (Cont.) MGMT$JOB_TARGETS

Column	Datatype	Description
TARGET_GUID	RAW(16)	The unique global identifier for the target.

9.8.3 MGMT$JOB_EXECUTION_HISTORY

MGMT$JOB_EXECUTION_HISTORY displays a summary of job executions along with their status and targets for each execution.

Table 9–44 MGMT$JOB_EXECUTION_HISTORY

Column	Datatype	Description
JOB_NAME	VARCHAR2(64)	The unique name for the job.
JOB_OWNER	VARCHAR2(256)	The owner/creator of the job.
JOB_ID	RAW(16)	The uniqe system identifier for the job.
JOB_TYPE	VARCHAR2(32)	The job type. For example, multi-task, SQL script or OS Command.
EXECUTION_ID	RAW(16)	The unique execution identifier.
SCHEDULED_TIME	DATE	The scheduled time of job execution.
START_TIME	DATE	The actual time the job executed.
END_TIME	DATE	The actual time the job terminated execution.
STATUS	VARCHAR2(40)	The current status of the job execution.
TARGET_NAME	VARCHAR2(256)	Name of the target the job was submitted against.
TARGET_TYPE	VARCHAR2(64)	The type of target the job was submitted against. Applies to single-target jobs only.
TARGET_GUID	RAW(16)	The unique global identifier for the target.

9.8.4 MGMT$JOB_STEP_HISTORY

MGMT$JOB_STEP_HISTORY displays step-level details of job executions.

Table 9–45 MGMT$JOB_STEP_HISTORY

Column	Datatype	Description
JOB_NAME	VARCHAR2(64)	The unique name for the job.
JOB_OWNER	VARCHAR2(256)	The owner/creator of the job.
JOB_ID	RAW(16)	The uniqe system identifier for the job.
EXECUTION_ID	RAW(16)	The unique execution identifier.
STEP_NAME	VARCHAR2(64)	The name of the job step.
START_TIME	DATE	The start time of the job step.
END_TIME	DATE	The end time of the job step.
STATUS	VARCHAR2(40)	The current status of the job execution.
TARGET_NAME	VARCHAR2(256)	Name of the target the job was submitted against.
TARGET_TYPE	VARCHAR2(64)	The type of target the job was submitted against. Applies to single-target jobs only.
TARGET_GUID	RAW(16)	The unique global identifier for the target.
OUTPUT	CLOB	Generated job output.

9.8.5 MGMT$JOB_ANNOTATIONS

MGMT$JOB_ANNOTATIONS displays a summary of annotations for changes in job status.

Table 9–46 MGMT$JOB_ANNOTATIONS

Column	Datatype	Description
JOB_NAME	VARCHAR2(64)	The unique name for the job.
JOB_OWNER	VARCHAR2(256)	The owner/creator of the job.
JOB_STATUS	NUMBER(2)	The job status. Possible values are as follows: 1: Scheduled 2: Executing 3: Aborted 4: Failed 5: Completed 6: Suspended 7: Agent Down 8: Stopped 9: Suspended/Lock 11: Suspended/Blackout 13: Suspend Pending 15: Queued 16: Failed 17: Waiting 18: Skipped
OCCURRENCE_TIMESTAMP	DATE	The time at which the state change occurred.
ANNOTATION_MESSAGE	VARCHAR2(4000)	Annotation text.
ANNOTATION_TIMESTAMP	DATE	The time the annotation was created.
ANNOTATED_BY	VARCHAR2(256)	Enterprise Manager administrator who authored the the annotation.

9.8.6 MGMT$JOB_NOTIFICATION_LOG

MGMT$JOB_NOTIFICATION_LOG displays details of notification deliveries for changes in job status.

Table 9–47 MGMT$JOB_NOTIFICATION_LOG

Column	Datatype	Description
JOB_NAME	VARCHAR2(64)	The unique name for the job.
JOB_OWNER	VARCHAR2(256)	The owner/creator of the job.

Table 9–47 (Cont.) MGMT$JOB_NOTIFICATION_LOG

Column	Datatype	Description
JOB_STATUS	NUMBER(2)	The job status. Possible values are as follows: 1: Scheduled 2: Executing 3: Aborted 4: Failed 5: Completed 6: Suspended 7: Agent Down 8: Stopped 9: Suspended/Lock 11: Suspended/Blackout 13: Suspend Pending 15: Queued 16: Failed 17: Waiting 18: Skipped
OCCURRENCE_TIMESTAMP	DATE	The time at which the state change occurred.
DELIVERY_MESSAGE	VARCHAR2(1024)	The message indicating the success or failure of the notification delivery.
DELIVERY_TIMESTAMP	DATE	The time at which the log message was created.

9.9 Application Service Level Management Views

9.9.1 MGMT$CSM_REGION

MGMT$CSM_REGION displays a list of regions and their member targets.

Table 9–48 MGMT$CSM_REGION

Column	Datatype	Description
TARGET_NAME	VARCHAR2(64)	Name of the target from which the metrics will be collected (Webcache target).
TARGET_TYPE	VARCHAR2(64)	Type defining the set of metrics that are applicable for the target.
REGION_NAME	VARCHAR2(128)	A meaningful name for a logical group (region) containing one or more domains. For example, the REGION_NAME can be a country, company, or company division.
DESCRIPTION	VARCHAR2(256)	A useful description of region.
REGION_MEMBER	VARCHAR2(256)	A member of the region specified by REGION_NAME. REGION_MEMBER can be a domain name or an IP subnet address.

Table 9–48 (Cont.) MGMT$CSM_REGION

Column	Datatype	Description
REGION_MEMBER_TYPE	VARCHAR2(16)	Specifies the format type (domain name or IP subnet address) of REGION_ MEMBER. Member type can be specified using "D" for domain or "S" for subnet. For example: REGION_MEMBER=us.oracle MEMBER_ TYPE=D REGION_MEMBER=138.1 MEMBER_ TYPE=S

Usage Notes

List the regions with detailed information.

List the regions of which a target is a member.

List the targets that belong to a region

9.9.2 MGMT$CSM_WATCHLIST

MGMT$CSM_WATCHLIST displays a list of URLs defined as part of a website target's watchlist. These URLs are periodically visited to monitor the website.

Table 9–49 MGMT$CSM_WATCHLIST

Column	Datatype	Description
TARGET_NAME	VARCHAR2(64)	Name of the target from which the metrics will be collected (Webcache target).
TARGET_TYPE	VARCHAR2(64)	Type defining the set of metrics that are applicable for the target.
DISPLAY_NAME	VARCHAR2(128)	Display name for the URL.
URL	VARCHAR2(512)	Full URL filename containing the relative path from the web server. (For example, "/somepath/somefile.html").
DESCRIPTION	VARCHAR2(256)	Text description of the URL. (Optional)

Usage Notes

List the URL watchlist for a given target

9.9.3 MGMT$CSM_METRIC_DETAILS

MGMT$CSM_METRIC_DETAILS displays detailed information on each individual HTTP request for composite targets (for example, a website target) that has one or more Web Cache targets.

Table 9–50 MGMT$CSM_METRIC_DETAILS

Column	Datatype	Description
TARGET_NAME	VARCHAR2(64)	Name of the target from which the metrics will be collected (Webcache target).
TARGET_TYPE	VARCHAR2(64)	Type defining the set of metrics that are applicable for the target.
COLLECTION_ TIMESTAMP	DATE	The date/time when the metric was collected
METRIC_NAME	VARCHAR2(64)	The name of the metric that is being defined

Table 9–50 (Cont.) MGMT$CSM_METRIC_DETAILS

Column	Datatype	Description
METRIC_VALUE	NUMBER	Number of milliseconds required to access this URL. This value is the difference in milliseconds between SUBMIT_ACTION_TIMESTAMP and LOAD_ACTION_TIMESTAMP
URL	VARCHAR2(1024)	URL filename relative to the webserver host and port. The response of this url is measured in elapsed_time.
VISITOR_IP	VARCHAR2(32)	IP address of machine making requests
VISITOR_NODE	VARCHAR2(1024)	Resolved machine name from which a logged request is made.
VISITOR_DOMAIN	VARCHAR2(1024)	Domain of the machine making requests.
VISITOR_SUBNET	VARCHAR2(32)	Domain subnet of the machine making the requests.

Usage Notes

Analyze HTTP request patterns over time with regards to visitor information (IP, subnet, domain, region) or URLs (most requested, least requested) for composite targets.

Analyze HTTP request response times over time with regards to visitor information (IP, subnet, domain, region) and URLs for composite targets.

9.9.4 MGMT$CSM_MT_METRIC_DETAILS

MGMT$CSM_MT_METRIC_DETAILS displays detailed information on each individual HTTP request for Web Cache targets (i.e. member targets). If a Web Cache target is not a member of any composite targets, the composite target columns are presented as NULL. If a Web Cache target is a member of multiple composite targets, a row for each composite target is presented.

Table 9–51 MGMT$CSM_MT_METRIC_DETAILS

Column	Datatype	Description
MEMBER_TARGET_ NAME	VARCHAR2(64)	Name of the Web Cache target.
MEMBER_TARGET_TYPE	VARCHAR2(64)	Type of the Web Cache target.
COMPOSITE_TARGET_ NAME	VARCHAR2(64)	The name of the composite target. A composite target is a grouping of one or more target name and target types from the target table. This entry may be null if the Web Cache target is not a member of any composite targets.
COMPOSITE_TARGET_ TYPE	VARCHAR2(64)	The type of the composite target. The type defines the set of metrics that are applicable for the composite target. This entry may be null if the Web Cache target is not a member of any composite targets.
COLLECTION_ TIMESTAMP	DATE	The time when the target status change was last detected and logged in the Management Repository.
METRIC_NAME	VARCHAR2(64)	Name of the metric being defined.
METRIC_VALUE	NUMBER	Number of milliseconds required to access this url. This value is the difference in milliseconds between SUBMIT_ACTION_TIMESTAMP and LOAD_ACTION_TIMESTAMP

Table 9–51 (Cont.) MGMT$CSM_MT_METRIC_DETAILS

Column	Datatype	Description
URL	VARCHAR2(1024)	URL filename relative to the webserver host and port.
VISITOR_IP	VARCHAR2(32)	IP address of machine making requests
VISITOR_NODE	VARCHAR2(1024)	Resolved machine name from which a logged request is made.
VISITOR_DOMAIN	VARCHAR2(1024)	Domain of the machine making requests.
VISITOR_SUBNET	VARCHAR2(32)	Domain subnet of the machine making the requests.

Usage Notes

Analyze HTTP requests patterns over time with regards to visitor information (IP, subnet, domain, region) or URLs (most requested, least requested) for a particular Web Cache member target of a composite target, or for Web Cache targets that are not a member of any composite targets.

Analyze HTTP request response times over time with regards to visitor information (IP, subnet, domain, region) and URLs for a particular Web Cache member target of a composite target, or for Web Cache targets that are not a member of any composite targets.

9.9.5 MGMT$CSM_URL_HOURLY

MGMT$CSM_URL_HOURLY displays statistical information about the HTTP requests for a given composite target, aggregated by URL in hourly time periods.

Table 9–52 MGMT$CSM_URL_HOURLY

Column	Datatype	Description
TARGET_NAME	VARCHAR2(64)	The name of the composite target. A composite target is a grouping of one or more target name and target types from the target table. This entry may be null if the Web Cache target is not a member of any composite targets.
TARGET_TYPE	VARCHAR2(64)	The type of the composite target. The type defines the set of metrics that are applicable for the composite target. This entry may be null if the Web Cache target is not a member of any composite targets.
URL	VARCHAR2(1024)	URL filename relative to the webserver host and port.
ROLLUP_TIMESTAMP	DATE	Identifies the start of the rollup period. For the one-hour rollups, samples that fall within the hourly boundaries from minute 00 through minute 59 inclusive will be combined. For example, samples from 12:00 AM through 12:59 AM would be combined into a single aggregated record with a rollup timestamp of "date" 12:00 AM.
METRIC_NAME	VARCHAR2(64)	Name of the metric being defined.
SAMPLE_COUNT	NUMBER	The average of the metric values for the samples that have been included in the rollup period
AVERAGE	NUMBER	The average of the metric values for the samples that have been included in the rollup period

Table 9–52 (Cont.) MGMT$CSM_URL_HOURLY

Column	Datatype	Description
MINIMUM	NUMBER	The minimum value for the metric for the samples that have been included in the rollup period.
MAXIMUM	NUMBER	The maximum value for the metric for samples that have been included in the rollup period.
STANDARD_DEVIATION	NUMBER	The standard deviation for the metric values that have been included in the rollup period.
VARIANCE	NUMBER	The variance for the metric values that has been included in the rollup period

Usage Notes

Analyze hourly HTTP response time patterns and statistical (min, max, avg, stddev) information for a given URL of a composite target. (For example, what was the average response time experienced when accessing URL HTTP://my.oracle.com/home.html of the website my.oracle.com between 8 and 9 this morning).

9.9.6 MGMT$CSM_URL_DAILY

MGMT$CSM_URL_DAILY displays statistical information about the HTTP requests for a given composite target, aggregated by URL in daily time periods.

Table 9–53 MGMT$CSM_URL_DAILY

Column	Datatype	Description
TARGET_NAME	VARCHAR2(64)	The name of the composite target. A composite target is a grouping of one or more target name and target types from the target table.
TARGET_TYPE	VARCHAR2(64)	The type of the composite target. The type defines the set of metrics that are applicable for the target.
URL	VARCHAR2(1024)	Full URL filename containing the relative path from the web server. (For example, "/somepath/somefile.html").
ROLLUP_TIMESTAMP	DATE	Identifies the start of the rollup period. For the one-hour rollups, samples that fall within the hourly boundaries from minute 00 through minute 59 inclusive will be combined. For example, samples from 12:00 AM through 12:59 AM would be combined into a single aggregated record with a rollup timestamp of "date" 12:00 AM.
METRIC_NAME	VARCHAR2(64)	Name of the metric being defined.
SAMPLE_COUNT	NUMBER	The average of the metric values for the samples that have been included in the rollup period
AVERAGE	NUMBER	The average of the metric values for the samples that have been included in the rollup period
MINIMUM	NUMBER	The minimum value for the metric for the samples that have been included in the rollup period.
MAXIMUM	NUMBER	The maximum value for the metric for samples that have been included in the rollup period.
STANDARD_DEVIATION	NUMBER	The standard deviation for the metric values that have been included in the rollup period.

Table 9–53 (Cont.) MGMT$CSM_URL_DAILY

Column	Datatype	Description
VARIANCE	NUMBER	The variance for the metric values that has been included in the rollup period

Usage Notes

Analyze daily HTTP response time patterns and statistical (min, max, avg, stddev) information for a given URL of a composite target. (For example, what was the avg response time experienced when accessing URL HTTP://my.oracle.com/home.html of the website my.oracle.com last Monday).

9.9.7 MGMT$CSM_URL_DIST_HOURLY

MGMT$CSM_URL_DIST_HOURLY displays the distribution of the HTTP request response times, in seconds, for a URL of a composite target. The data is aggregated in hourly time periods.

Table 9–54 MGMT$CSM_URL_DIST_HOURLY

Column	Datatype	Description
TARGET_NAME	VARCHAR2(64)	The name of the composite target. A composite target is a grouping of one or more target name and target types from the target table.
TARGET_TYPE	VARCHAR2(64)	The type of the composite target. The type defines the set of metrics that are applicable for the target.
URL	VARCHAR2(1024)	Full URL filename containing the relative path from the web server. (For example, "/somepath/somefile.html").
ROLLUP_TIMESTAMP	DATE	Start of the rollup period (time zone of the target).
METRIC_NAME	VARCHAR2(64)	Name of the metric being defined.
METRIC_VALUE	NUMBER	Number of milliseconds required to access this url. This value is the difference in milliseconds between SUBMIT_ACTION_TIMESTAMP and LOAD_ACTION_TIMESTAMP
SAMPLE_COUNT	NUMBER	The number of non-NULL samples for the metric that were aggregated.

Usage Notes

Analyze the distribution of the HTTP response times for a URL of a composite target during a specific hour of the day (For example, compare the percentage of requests which response time was longer than 5 seconds at 8am and at 5pm).

9.9.8 MGMT$CSM_URL_DIST_DAILY

MGMT$CSM_URL_DIST_DAILY displays the distribution of the HTTP request response times, in seconds, for a URL of a composite target. The data is aggregated in daily time periods.

Table 9–55 MGMT$CSM_URL_DIST_DAILY

Column	Datatype	Description
TARGET_NAME	VARCHAR2(64)	The name of the composite target. A composite target is a grouping of one or more target name and target types from the target table.
TARGET_TYPE	VARCHAR2(64)	The type of the composite target. The type defines the set of metrics that are applicable for the target.
URL	VARCHAR2(1024)	Full URL filename containing the relative path from the web server. (For example, "/somepath/somefile.html").
ROLLUP_TIMESTAMP	DATE	Start of the rollup period (time zone of the target).
METRIC_NAME	VARCHAR2(64)	Name of the metric being defined.
METRIC_VALUE	NUMBER	Number of milliseconds required to access this url. This value is the difference in milliseconds between SUBMIT_ACTION_TIMESTAMP and LOAD_ACTION_TIMESTAMP
SAMPLE_COUNT	NUMBER	The number of non-NULL samples for the metric that were aggregated.

Usage Notes

Analyze the distribution of the HTTP response times for a given URL of a composite target during a specific day (For example, compare the percentage of requests which response time was longer than 5 seconds on Monday and on Saturday).

9.9.9 MGMT$CSM_MT_URL_HOURLY

MGMT$CSM_MT_URL_HOURLY displays statistical information about the HTTP requests for a given Web Cache target of a composite target, aggregated by URL in hourly time periods. If a Web Cache target is not a member of any composite targets, the composite target columns are presented as NULL. If a Web Cache target is a member of multiple composite targets, a row for each composite target is presented.

Table 9–56 MGMT$CSM_MT_URL_HOURLY

Column	Datatype	Description
MEMBER_TARGET_ NAME	VARCHAR2(64)	The Web Cache target.
MEMBER_TARGET_TYPE	VARCHAR2(64)	The target type defines the set of metrics that are applicable for the target.
COMPOSITE_TARGET_ NAME	VARCHAR2(64)	The name of the composite target. A composite target is a grouping of one or more target name and target types from the target table. May be null if the Web Cache target is not a member of any composite targets.
COMPOSITE_TARGET_ TYPE	VARCHAR2(64)	The type of the composite target. The type defines the set of metrics that are applicable for the target. May be null if the Web Cache target is not a member of any composite target.
URL	VARCHAR2(1024)	Full URL filename containing the relative path from the web server. (For example, "/somepath/somefile.html").
ROLLUP_TIMESTAMP	DATE	Start of the rollup period (time zone of the target).
METRIC_NAME	VARCHAR2(64)	Name of the metric being defined.

Table 9–56 (Cont.) MGMT$CSM_MT_URL_HOURLY

Column	Datatype	Description
SAMPLE_COUNT	NUMBER	The number of non-NULL samples for the metric that were aggregated.
AVERAGE	NUMBER	The average of the metric values for the samples that have been included in the rollup period
MINIMUM	NUMBER	The minimum value for the metric for the samples that have been included in the rollup period.
MAXIMUM	NUMBER	The maximum value for the metric for samples that have been included in the rollup period.
STANDARD_DEVIATION	NUMBER	The standard deviation for the metric values that have been included in the rollup period.
VARIANCE	NUMBER	The variance for the metric values that has been included in the rollup period

Usage Notes

Analyze hourly HTTP response time patterns and statistical (min, max, avg, stddev) information for a given URL of a Web Cache target and optionally of a composite target (For example, what was the avg response time experienced when accessing URL HTTP://my.oracle.com/home.html at Web Cache WEB5 of the website my.oracle.com between 8 and 9 this morning).

9.9.10 MGMT$CSM_MT_URL_DAILY

MGMT$CSM_MT_URL_DAILY displays statistical information about the HTTP requests for a given Web Cache target of a composite target, aggregated by URL in daily time periods. If a Web Cache target is not a member of any composite targets, the composite target columns are presented as NULL. If a Web Cache target is a member of multiple composite targets, a row for each composite target is presented.

Table 9–57 MGMT$CSM_MT_URL_DAILY

Column	Datatype	Description
MEMBER_TARGET_ NAME	VARCHAR2(64)	The Web Cache target.
MEMBER_TARGET_TYPE	VARCHAR2(64)	The target type defines the set of metrics that are applicable for the target.
COMPOSITE_TARGET_ NAME	VARCHAR2(64)	The name of the composite target. A composite target is a grouping of one or more target name and target types from the target table. May be null if the Web Cache target is not a member of any composite targets.
COMPOSITE_TARGET_ TYPE	VARCHAR2(64)	The type of the composite target. The type defines the set of metrics that are applicable for the target. May be null if the Web Cache target is not a member of any composite targets.
URL	VARCHAR2(1024)	Full URL filename containing the relative path from the web server. (For example, "/somepath/somefile.html").
ROLLUP_TIMESTAMP	DATE	Start of the rollup period (time zone of the target).
METRIC_NAME	VARCHAR2(64)	Name of the metric being defined.
SAMPLE_COUNT	NUMBER	The number of non-NULL samples for the metric that were aggregated.

Table 9–57 (Cont.) MGMT$CSM_MT_URL_DAILY

Column	Datatype	Description
AVERAGE	NUMBER	The average of the metric values for the samples that have been included in the rollup period
MINIMUM	NUMBER	The minimum value for the metric for the samples that have been included in the rollup period.
MAXIMUM	NUMBER	The maximum value for the metric for samples that have been included in the rollup period.
STANDARD_DEVIATION	NUMBER	The standard deviation for the metric values that have been included in the rollup period.
VARIANCE	NUMBER	The variance for the metric values that has been included in the rollup period

Usage Notes

Analyze daily HTTP response time patterns and statistical (min, max, avg, stddev) information for a given URL of a Web Cache target and optionally of a composite target. target (For example, what was the avg response time experienced when accessing URL HTTP://my.oracle.com/home.html at Web Cache WEB5 of the website my.oracle.com last Monday).

9.9.11 MGMT$CSM_MT_URL_DIST_HOURLY

MGMT$CSM_MT_URL_DIST_HOURLY displays the distribution of the HTTP request response times, in seconds, for a URL of a given Web Cache target of a composite target. The data is aggregated in hourly time periods. If a Web Cache target is not a member of any composite targets, the composite target columns are presented as NULL. If a Web Cache target is a member of multiple composite targets, a row for each composite target is presented.

Table 9–58 MGMT$CSM_MT_URL_DIST_HOURLY

Column	Datatype	Description
MEMBER_TARGET_ NAME	VARCHAR2(64)	The Web Cache target.
MEMBER_TARGET_TYPE	VARCHAR2(64)	The target type defines the set of metrics that are applicable for the target.
COMPOSITE_TARGET_ NAME	VARCHAR2(64)	The name of the composite target. A composite target is a grouping of one or more target name and target types from the target table. May be null if the Web Cache target is not a member of any composite targets.
COMPOSITE_TARGET_ TYPE	VARCHAR2(64)	The type of the composite target. The type defines the set of metrics that are applicable for the target. May be null if the Web Cache target is not a member of any composite targets.
URL	VARCHAR2(1024)	Full URL filename containing the relative path from the web server. (For example, "/somepath/somefile.html").
ROLLUP_TIMESTAMP	DATE	Start of the rollup period (time zone of the target).
METRIC_NAME	VARCHAR2(64)	Name of the metric being defined.
METRIC_VALUE	NUMBER	Number of milliseconds required to access this url. This value is the difference in milliseconds between SUBMIT_ACTION_TIMESTAMP and LOAD_ACTION_TIMESTAMP

Table 9–58 (Cont.) MGMT$CSM_MT_URL_DIST_HOURLY

Column	Datatype	Description
SAMPLE_COUNT	NUMBER	The number of non-NULL samples for the metric that were aggregated.

Usage Notes

Analyze the distribution of the HTTP response times for a URL of a Web Cache target of a composite target during a specific hour of the day (For example, how many requests for URL HTTP://my.oracle.com/home.html at Web Cache WEB5 of the website my.oracle.com took longer than 5 seconds between 8 and 9 this morning).

9.9.12 MGMT$CSM_MT_URL_DIST_DAILY

MGMT$CSM_MT_URL_DIST_DAILY displays the distribution of the HTTP request response times, in seconds, for a URL of a given Web Cache target of a composite target. The data is aggregated in daily time periods. If a Web Cache target is not a member of any composite targets, the composite target columns are presented as NULL. If a Web Cache target is a member of multiple composite targets, a row for each composite target is presented.

Table 9–59 MGMT$CSM_MT_URL_DIST_DAILY

Column	Datatype	Description
MEMBER_TARGET_ NAME	VARCHAR2(64)	The Web Cache target.
MEMBER_TARGET_TYPE	VARCHAR2(64)	The target type defines the set of metrics that are applicable for the target.
COMPOSITE_TARGET_ NAME	VARCHAR2(64)	The name of the composite target. A composite target is a grouping of one or more target name and target types from the target table. May be null if the Web Cache target is not a member of any composite targets.
COMPOSITE_TARGET_ TYPE	VARCHAR2(64)	The type of the composite target. The type defines the set of metrics that are applicable for the target. May be null if the Web Cache target is not a member of any composite targets.
URL	VARCHAR2(1024)	Full URL filename containing the relative path from the web server. (For example, "/somepath/somefile.html").
ROLLUP_TIMESTAMP	DATE	Start of the rollup period (time zone of the target).
METRIC_NAME	VARCHAR2(64)	Name of the metric being defined.
METRIC_VALUE	NUMBER	Number of milliseconds required to access this url. This value is the difference in milliseconds between SUBMIT_ACTION_TIMESTAMP and LOAD_ACTION_TIMESTAMP
SAMPLE_COUNT	NUMBER	The number of non-NULL samples for the metric that were aggregated.

Usage Notes

Analyze the distribution of the HTTP response times for a URL of a Web Cache target of a composite target during a specific day (For example, how many requests for URL HTTP://my.oracle.com/home.html at Web Cache WEB5 of the website my.oracle.com took longer than 5 seconds last Monday).

9.9.13 MGMT$CSM_IP_HOURLY

MGMT$CSM_IP_HOURLY displays statistical information about the HTTP requests for a given composite target, aggregated by the visitor IP address in hourly time periods.

Table 9–60 MGMT$CSM_IP_HOURLY

Column	Datatype	Description
TARGET_NAME	VARCHAR2(64)	The name of the composite target. A composite target is a grouping of one or more target name and target types from the target table.
TARGET_TYPE	VARCHAR2(64)	The type of the composite target. The type defines the set of metrics that are applicable for the target.
VISITOR	VARCHAR2(1024)	Node name (if available) or IP address of the requesting target. This may be a gateway or router if one exists between the user machine and the webserver target.
ROLLUP_TIMESTAMP	DATE	Start of the rollup period (time zone of the target).
METRIC_NAME	VARCHAR2(64)	Name of the metric being defined.
SAMPLE_COUNT	NUMBER	The number of non-NULL samples for the metric that were aggregated.
AVERAGE	NUMBER	The average of the metric values for the samples that have been included in the rollup period
MINIMUM	NUMBER	The minimum value for the metric for the samples that have been included in the rollup period.
MAXIMUM	NUMBER	The maximum value for the metric for samples that have been included in the rollup period.
STANDARD_DEVIATION	NUMBER	The standard deviation for the metric values that have been included in the rollup period.
VARIANCE	NUMBER	The variance for the metric values that has been included in the rollup period

Usage Notes

Analyze hourly HTTP response time patterns and statistical (min, max, avg, stddev) information for a given visitor IP address of a composite target (For example, what was the avg response time experienced from IP address 192.168.1.1 when accessing the website my.oracle.com between 8 and 9 this morning).

9.9.14 MGMT$CSM_IP_DAILY

MGMT$CSM_IP_DAILY displays statistical information about the HTTP requests for a given composite target, aggregated by visitor IP address in daily time periods.

Table 9–61 MGMT$CSM_IP_DAILY

Column	Datatype	Description
TARGET_NAME	VARCHAR2(64)	The name of the composite target. A composite target is a grouping of one or more target name and target types from the target table.
TARGET_TYPE	VARCHAR2(64)	The type of the composite target. The type defines the set of metrics that are applicable for the target.

Table 9–61 (Cont.) MGMT$CSM_IP_DAILY

Column	Datatype	Description
VISITOR	VARCHAR2(1024)	Node name (if available) or IP address of the requesting target. This may be a gateway or router if one exists between the user machine and the webserver target.
ROLLUP_TIMESTAMP	DATE	Start of the rollup period (time zone of the target).
METRIC_NAME	VARCHAR2(64)	Name of the metric being defined.
SAMPLE_COUNT	NUMBER	The number of non-NULL samples for the metric that were aggregated.
AVERAGE	NUMBER	The average of the metric values for the samples that have been included in the rollup period
MINIMUM	NUMBER	The minimum value for the metric for the samples that have been included in the rollup period.
MAXIMUM	NUMBER	The maximum value for the metric for samples that have been included in the rollup period.
STANDARD_DEVIATION	NUMBER	The standard deviation for the metric values that have been included in the rollup period.
VARIANCE	NUMBER	The variance for the metric values that has been included in the rollup period

Usage Notes

Analyze daily HTTP response time patterns and statistical (min, max, avg, stddev) information for a given visitor IP address of a composite target (For example, what was the avg response time experienced from IP address 192.168.1.1 when accessing the website my.oracle.com last Monday).

9.9.15 MGMT$CSM_IP_DIST_HOURLY

MGMT$CSM_IP_DIST_HOURLY displays the distribution of the HTTP request response times, in seconds, for a visitor IP address of a composite target. The data is aggregated in hourly time periods.

Table 9–62 MGMT$CSM_IP_DIST_HOURLY

Column	Datatype	Description
TARGET_NAME	VARCHAR2(64)	The name of the composite target. A composite target is a grouping of one or more target name and target types from the target table.
TARGET_TYPE	VARCHAR2(64)	The type of the composite target. The type defines the set of metrics that are applicable for the target.
VISITOR	VARCHAR2(1024)	Node name (if available) or IP address of the requesting target. This may be a gateway or router if one exists between the user machine and the webserver target.
ROLLUP_TIMESTAMP	DATE	Start of the rollup period (time zone of the target).
METRIC_NAME	NUMBER	Name of the metric being defined.
METRIC_VALUE	NUMBER	Number of milliseconds required to access this url. This value is the difference in milliseconds between SUBMIT_ACTION_TIMESTAMP and LOAD_ACTION_TIMESTAMP

Table 9–62 (Cont.) MGMT$CSM_IP_DIST_HOURLY

Column	Datatype	Description
SAMPLE_COUNT	NUMBER	The number of non-NULL samples for the metric that were aggregated.

Usage Notes

Analyze the distribution of the HTTP response times for a visitor IP address of a composite target during a specific hour of the day (For example, how many requests from IP address 192.168.1.1 to the website my.oracle.com took longer than 5 seconds between 8 and 9 this morning).

9.9.16 MGMT$CSM_IP_DIST_DAILY

MGMT$CSM_IP_DIST_DAILY displays the distribution of the HTTP request response times, in seconds, for a visitor IP address of a composite target. The data is aggregated in daily time periods.

Table 9–63 MGMT$CSM_IP_DIST_DAILY

Column	Datatype	Description
TARGET_NAME	VARCHAR2(64)	The name of the composite target. A composite target is a grouping of one or more target name and target types from the target table.
TARGET_TYPE	VARCHAR2(64)	The type of the composite target. The type defines the set of metrics that are applicable for the target.
VISITOR	VARCHAR2(1024)	Node name (if available) or IP address of the requesting target. This may be a gateway or router if one exists between the user machine and the webserver target.
ROLLUP_TIMESTAMP	DATE	Start of the rollup period (time zone of the target).
METRIC_NAME	NUMBER	Name of the metric being defined.
METRIC_VALUE	NUMBER	Number of milliseconds required to access this url. This value is the difference in milliseconds between SUBMIT_ACTION_TIMESTAMP and LOAD_ACTION_TIMESTAMP
SAMPLE_COUNT	NUMBER	The number of non-NULL samples for the metric that were aggregated.

Usage Notes

Analyze the distribution of the HTTP response times for a given visitor IP address of a composite target during a specific day (For example, how many requests from IP address 192.168.1.1 to the website my.oracle.com took longer than 5 seconds last Monday).

9.9.17 MGMT$CSM_MT_IP_HOURLY

MGMT$CSM_MT_IP_HOURLY displays statistical information about the HTTP requests for a given Web Cache target of a composite target, aggregated by visitor IP address in hourly time periods. If a Web Cache target is not a member of any composite targets, the composite target columns are presented as NULL. If a Web Cache target is a member of multiple composite targets, a row for each composite target is presented.

Table 9–64 MGMT$CSM_MT_IP_HOURLY

Column	Datatype	Description
MEMBER_TARGET_ NAME	VARCHAR2(64)	The Web Cache target.
MEMBER_TARGET_TYPE	VARCHAR2(64)	The target type defines the set of metrics that are applicable for the target.
COMPOSITE_TARGET_ NAME	VARCHAR2(64)	The name of the composite target. A composite target is a grouping of one or more target name and target types from the target table. May be null if the Web Cache target is not a member of any composite targets.
COMPOSITE_TARGET_ TYPE	VARCHAR2(64)	The type of the composite target. The type defines the set of metrics that are applicable for the target. May be null if the Web Cache target is not a member of any composite targets.
VISITOR	VARCHAR2(1024)	Node name (if available) or IP address of the requesting target. This may be a gateway or router if one exists between the user machine and the webserver target.
ROLLUP_TIMESTAMP	DATE	Start of the rollup period (time zone of the target).
METRIC_NAME	VARCHAR2(64)	Name of the metric being defined.
SAMPLE_COUNT	NUMBER	The number of non-NULL samples for the metric that were aggregated.
AVERAGE	NUMBER	The average of the metric values for the samples that have been included in the rollup period
MINIMUM	NUMBER	The minimum value for the metric for the samples that have been included in the rollup period.
MAXIMUM	NUMBER	The maximum value for the metric for samples that have been included in the rollup period.
STANDARD_DEVIATION	NUMBER	The standard deviation for the metric values that have been included in the rollup period.
VARIANCE	NUMBER	The variance for the metric values that has been included in the rollup period

Usage Notes

Analyze hourly HTTP response time patterns and statistical (min, max, avg, stddev) information for a given visitor IP address of a Web Cache target and optionally of a composite target (For example, what was the avg response time experienced from IP address 192.168.1.1 when accessing Web Cache WEB5 of the website my.oracle.com between 8 and 9 this morning).

9.9.18 MGMT$CSM_MT_IP_DAILY

MGMT$CSM_MT_IP_DAILY displays statistical information about the HTTP requests for a given Web Cache target of a composite target, aggregated by visitor IP address in daily time periods. If a Web Cache target is not a member of any composite targets, the composite target columns are presented as NULL. If a Web Cache target is a member of multiple composite targets, a row for each composite target is presented.

Table 9–65 MGMT$CSM_MT_IP_DAILY

Column	Datatype	Description
MEMBER_TARGET_ NAME	VARCHAR2(64)	The Web Cache target.

Table 9–65 (Cont.) MGMT$CSM_MT_IP_DAILY

Column	Datatype	Description
MEMBER_TARGET_TYPE	VARCHAR2(64)	The target type defines the set of metrics that are applicable for the target.
COMPOSITE_TARGET_ NAME	VARCHAR2(64)	The name of the composite target. A composite target is a grouping of one or more target name and target types from the target table. May be null if the Web Cache target is not a member of any composite targets.
COMPOSITE_TARGET_ TYPE	VARCHAR2(64)	The type of the composite target. The type defines the set of metrics that are applicable for the target. May be null if the Web Cache target is not a member of any composite targets.
VISITOR	VARCHAR2(1024)	Node name (if available) or IP address of the requesting target. This may be a gateway or router if one exists between the user machine and the webserver target.
ROLLUP_TIMESTAMP	DATE	Start of the rollup period (time zone of the target).
METRIC_NAME	VARCHAR2(64)	Name of the metric being defined.
SAMPLE_COUNT	NUMBER	The number of non-NULL samples for the metric that were aggregated.
AVERAGE	NUMBER	The average of the metric values for the samples that have been included in the rollup period
MINIMUM	NUMBER	The minimum value for the metric for the samples that have been included in the rollup period.
MAXIMUM	NUMBER	The maximum value for the metric for samples that have been included in the rollup period.
STANDARD_DEVIATION	NUMBER	The standard deviation for the metric values that have been included in the rollup period.
VARIANCE	NUMBER	The variance for the metric values that has been included in the rollup period

Usage Notes

Analyze daily HTTP response time patterns and statistical (min, max, avg, stddev) information for a given visitor IP address of a Web Cache target and optionally of a composite target (For example, what was the avg response time experienced from IP address 192.168.1.1 when accessing Web Cache WEB5 of the website my.oracle.com last Monday).

9.9.19 MGMT$CSM_MT_IP_DIST_HOURLY

MGMT$CSM_MT_IP_DIST_HOURLY displays the distribution of the HTTP request response times, in seconds, for a visitor IP address of a given Web Cache target of a composite target. The data is aggregated in hourly time periods. If a Web Cache target is not a member of any composite targets, the composite target columns are presented as NULL. If a Web Cache target is a member of multiple composite targets, a row for each composite target is presented.

Table 9–66 MGMT$CSM_MT_IP_DIST_HOURLY

Column	Datatype	Description
MEMBER_TARGET_ NAME	VARCHAR2(64)	The Web Cache target.

Table 9–66 (Cont.) MGMT$CSM_MT_IP_DIST_HOURLY

Column	Datatype	Description
MEMBER_TARGET_TYPE	VARCHAR2(64)	The target type defines the set of metrics that are applicable for the target.
COMPOSITE_TARGET_ NAME	VARCHAR2(64)	The name of the composite target. A composite target is a grouping of one or more target name and target types from the target table. May be null if the Web Cache target is not a member of any composite targets.
COMPOSITE_TARGET_ TYPE	VARCHAR2(64)	The type of the composite target. The type defines the set of metrics that are applicable for the target. May be null if the Web Cache target is not a member of any composite targets.
VISITOR	VARCHAR2(1024)	Node name (if available) or IP address of the requesting target. This may be a gateway or router if one exists between the user machine and the webserver target.
ROLLUP_TIMESTAMP	DATE	Start of the rollup period (time zone of the target).
METRIC_NAME	VARCHAR2(64)	Name of the metric being defined.
METRIC_VALUE	NUMBER	Number of milliseconds required to access this url. This value is the difference in milliseconds between SUBMIT_ACTION_TIMESTAMP and LOAD_ACTION_TIMESTAMP
SAMPLE_COUNT	NUMBER	The number of non-NULL samples for the metric that were aggregated.

Usage Notes

Analyze the distribution of the HTTP response times for a visitor IP address of a Web Cache target of a composite target during a specific hour of the day (For example, how many requests from IP address 192.168.1.1 to Web Cache WEB5 of the website my.oracle.com took longer than 5 seconds between 8 and 9 this morning).

9.9.20 MGMT$CSM_MT_IP_DIST_DAILY

MGMT$CSM_MT_IP_DIST_DAILY displays the distribution of the HTTP request response times, in seconds, for a visitor IP address of a given Web Cache target of a composite target. The data is aggregated in daily time periods. If a Web Cache target is not a member of any composite targets, the composite target columns are presented as NULL. If a Web Cache target is a member of multiple composite targets, a row for each composite target is presented.

Table 9–67 MGMT$CSM_MT_IP_DIST_DAILY

Column	Datatype	Description
MEMBER_TARGET_ NAME	VARCHAR2(64)	The Web Cache target.
MEMBER_TARGET_TYPE	VARCHAR2(64)	The target type defines the set of metrics that are applicable for the target.
COMPOSITE_TARGET_ NAME	VARCHAR2(64)	The name of the composite target. A composite target is a grouping of one or more target name and target types from the target table. May be null if the Web Cache target is not a member of any composite targets.
COMPOSITE_TARGET_ TYPE	VARCHAR2(64)	The type of the composite target. The type defines the set of metrics that are applicable for the target. May be null if the Web Cache target is not a member of any composite targets.

Table 9–67 (Cont.) MGMT$CSM_MT_IP_DIST_DAILY

Column	Datatype	Description
VISITOR	VARCHAR2(1024)	Node name (if available) or IP address of the requesting target. This may be a gateway or router if one exists between the user machine and the webserver target.
ROLLUP_TIMESTAMP	DATE	Start of the rollup period (time zone of the target).
METRIC_NAME	VARCHAR2(64)	Name of the metric being defined.
METRIC_VALUE	NUMBER	Number of milliseconds required to access this url. This value is the difference in milliseconds between SUBMIT_ACTION_TIMESTAMP and LOAD_ACTION_TIMESTAMP
SAMPLE_COUNT	NUMBER	The number of non-NULL samples for the metric that were aggregated.

Usage Notes

Analyze the distribution of the HTTP response times for a visitor IP address of a Web Cache target of a composite target during a specific day (For example, how many requests from IP address 192.168.1.1 to Web Cache WEB5 of the website my.oracle.com took longer than 5 seconds last Monday).

9.9.21 MGMT$CSM_DOMAIN_HOURLY

MGMT$CSM_DOMAIN_HOURLY displays statistical information about the HTTP requests for a given composite target, aggregated by the visitor IP domain in hourly time periods.

Table 9–68 MGMT$CSM_DOMAIN_HOURLY

Column	Datatype	Description
TARGET_NAME	VARCHAR2(64)	The name of the composite target. A composite target is a grouping of one or more target name and target types from the target table.
TARGET_TYPE	VARCHAR2(64)	The type of the composite target. The type defines the set of metrics that are applicable for the target.
VISITOR_DOMAIN	VARCHAR2(1024)	Domain of the machine making requests.
ROLLUP_TIMESTAMP	DATE	Start of the rollup period (time zone of the target).
METRIC_NAME	VARCHAR2(64)	Name of the metric being defined.
SAMPLE_COUNT	NUMBER	The number of non-NULL samples for the metric that were aggregated.
AVERAGE	NUMBER	The average of the metric values for the samples that have been included in the rollup period
MINIMUM	NUMBER	The minimum value for the metric for the samples that have been included in the rollup period.
MAXIMUM	NUMBER	The maximum value for the metric for samples that have been included in the rollup period.
STANDARD_DEVIATION	NUMBER	The standard deviation for the metric values that have been included in the rollup period.
VARIANCE	NUMBER	The variance for the metric values that has been included in the rollup period

Usage Notes

Analyze hourly HTTP response time patterns and statistical (min, max, avg, stddev) information for a given visitor IP domain of a composite target (For example, what was the avg response time experienced from users of the domain oracle.co.uk when accessing the website my.oracle.com between 8 and 9 this morning).

9.9.22 MGMT$CSM_DOMAIN_DAILY

MGMT$CSM_DOMAIN_DAILY displays statistical information about the HTTP requests for a given composite target, aggregated by visitor IP domain in daily time periods.

Table 9–69 MGMT$CSM_DOMAIN_DAILY

Column	Datatype	Description
TARGET_NAME	VARCHAR2(64)	The name of the composite target. A composite target is a grouping of one or more target name and target types from the target table.
TARGET_TYPE	VARCHAR2(64)	The type of the composite target. The type defines the set of metrics that are applicable for the target.
VISITOR_DOMAIN	VARCHAR2(1024)	Domain of the machine making requests.
ROLLUP_TIMESTAMP	DATE	Start of the rollup period (time zone of the target).
METRIC_NAME	VARCHAR2(64)	Name of the metric being defined.
SAMPLE_COUNT	NUMBER	The number of non-NULL samples for the metric that were aggregated.
AVERAGE	NUMBER	The average of the metric values for the samples that have been included in the rollup period
MINIMUM	NUMBER	The minimum value for the metric for the samples that have been included in the rollup period.
MAXIMUM	NUMBER	The maximum value for the metric for samples that have been included in the rollup period.
STANDARD_DEVIATION	NUMBER	The standard deviation for the metric values that have been included in the rollup period.
VARIANCE	NUMBER	The variance for the metric values that has been included in the rollup period

Usage Notes

Analyze daily HTTP response time patterns and statistical (min, max, avg, stddev) information for a given visitor IP domain of a composite target (For example, what was the avg response time experienced from users of the domain oracle.co.uk when accessing the website my.oracle.com last Monday).

9.9.23 MGMT$CSM_DOMAIN_DIST_HOURLY

MGMT$CSM_DOMAIN_DIST_HOURLY displays the distribution of the HTTP request response times, in seconds, for a visitor IP domain of a composite target. The data is aggregated in hourly time periods.

Table 9–70 MGMT$CSM_DOMAIN_DIST_HOURLY

Column	Datatype	Description
TARGET_NAME	VARCHAR2(64)	The name of the composite target. A composite target is a grouping of one or more target name and target types from the target table.
TARGET_TYPE	VARCHAR2(64)	The type of the composite target. The type defines the set of metrics that are applicable for the target.
VISITOR_DOMAIN	VARCHAR2(1024)	Domain of the machine making requests.
ROLLUP_TIMESTAMP	DATE	Start of the rollup period (time zone of the target).
METRIC_NAME	VARCHAR2(64)	Name of the metric being defined.
METRIC_VALUE	NUMBER	Number of milliseconds required to access this url. This value is the difference in milliseconds between SUBMIT_ACTION_TIMESTAMP and LOAD_ACTION_TIMESTAMP
SAMPLE_COUNT	NUMBER	The number of non-NULL samples for the metric that were aggregated.

Usage Notes

Analyze the distribution of the HTTP response times for a visitor IP domain of a composite target during a specific hour of the day (For example, how many requests from users of the domain oracle.co.uk to the website my.oracle.com took longer than 5 seconds between 8 and 9 this morning).

9.9.24 MGMT$CSM_DOMAIN_DIST_DAILY

MGMT$CSM_DOMAIN_DIST_DAILY displays the distribution of the HTTP request response times, in seconds, for a visitor IP domain of a composite target. The data is aggregated in daily time periods.

Table 9–71 MGMT$CSM_DOMAIN_DIST_DAILY

Column	Datatype	Description
TARGET_NAME	VARCHAR2(64)	The name of the composite target. A composite target is a grouping of one or more target name and target types from the target table.
TARGET_TYPE	VARCHAR2(64)	The type of the composite target. The type defines the set of metrics that are applicable for the target.
VISITOR_DOMAIN	VARCHAR2(1024)	Domain of the machine making requests.
ROLLUP_TIMESTAMP	DATE	Start of the rollup period (time zone of the target).
METRIC_NAME	VARCHAR2(64)	Name of the metric being defined.
METRIC_VALUE	NUMBER	Number of milliseconds required to access this url. This value is the difference in milliseconds between SUBMIT_ACTION_TIMESTAMP and LOAD_ACTION_TIMESTAMP
SAMPLE_COUNT	NUMBER	The number of non-NULL samples for the metric that were aggregated.

Usage Notes

Analyze the distribution of the HTTP response times for a given visitor IP domain of a composite target during a specific day (For example, how many requests from users of

the domain oracle.co.uk to the website my.oracle.com took longer than 5 seconds last Monday).

9.9.25 MGMT$CSM_SUBNET_HOURLY

MGMT$CSM_SUBNET_HOURLY displays statistical information about the HTTP requests for a given composite target, aggregated by the visitor IP subnet in hourly time periods.

Table 9–72 MGMT$CSM_SUBNET_HOURLY

Column	Datatype	Description
TARGET_NAME	VARCHAR2(64)	The name of the composite target. A composite target is a grouping of one or more target name and target types from the target table.
TARGET_TYPE	VARCHAR2(64)	The type of the composite target. The type defines the set of metrics that are applicable for the target.
VISITOR_SUBNET	VARCHAR2(32)	Domain subnet of the machine making the requests.
ROLLUP_TIMESTAMP	DATE	Start of the rollup period (time zone of the target).
METRIC_NAME	VARCHAR2(64)	Name of the metric being defined.
SAMPLE_COUNT	NUMBER	The number of non-NULL samples for the metric that were aggregated.
AVERAGE	NUMBER	The average of the metric values for the samples that have been included in the rollup period
MINIMUM	NUMBER	The minimum value for the metric for the samples that have been included in the rollup period.
MAXIMUM	NUMBER	The maximum value for the metric for samples that have been included in the rollup period.
STANDARD_DEVIATION	NUMBER	The standard deviation for the metric values that have been included in the rollup period.
VARIANCE	NUMBER	The variance for the metric values that has been included in the rollup period

Usage Notes

Analyze hourly HTTP response time patterns and statistical (min, max, avg, stddev) information for a given visitor IP subnet of a composite target (For example, what was the avg response time experienced from users of the subnet 192.168.1 when accessing the website my.oracle.com between 8 and 9 this morning).

9.9.26 MGMT$CSM_SUBNET_DAILY

MGMT$CSM_SUBNET_DAILYdisplays statistical information about the HTTP requests for a given composite target, aggregated by visitor IP subnet in daily time periods.

Table 9–73 MGMT$CSM_SUBNET_DAILY

Column	Datatype	Description
TARGET_NAME	VARCHAR2(64)	The name of the composite target. A composite target is a grouping of one or more target name and target types from the target table.

Table 9–73 (Cont.) MGMT$CSM_SUBNET_DAILY

Column	Datatype	Description
TARGET_TYPE	VARCHAR2(64)	The type of the composite target. The type defines the set of metrics that are applicable for the target.
VISITOR_SUBNET	VARCHAR2(32)	Domain subnet of the machine making the requests.
ROLLUP_TIMESTAMP	DATE	Start of the rollup period (time zone of the target).
METRIC_NAME	VARCHAR2(64)	Name of the metric being defined.
SAMPLE_COUNT	NUMBER	The number of non-NULL samples for the metric that were aggregated.
AVERAGE	NUMBER	The average of the metric values for the samples that have been included in the rollup period
MINIMUM	NUMBER	The minimum value for the metric for the samples that have been included in the rollup period.
MAXIMUM	NUMBER	The maximum value for the metric for samples that have been included in the rollup period.
STANDARD_DEVIATION	NUMBER	The standard deviation for the metric values that have been included in the rollup period.
VARIANCE	NUMBER	The variance for the metric values that has been included in the rollup period

Usage Notes

Analyze daily HTTP response time patterns and statistical (min, max, avg, stddev) information for a given visitor IP subnet of a composite target (For example, what was the avg response time experienced from users of the subnet 192.168.1 when accessing the website my.oracle.com last Monday).

9.9.27 MGMT$CSM_SUBNET_DIST_HOURLY

MGMT$CSM_SUBNET_DIST_HOURLY displays the distribution of the HTTP request response times, in seconds, for a visitor IP subnet of a composite target. The data is aggregated in hourly time periods.

Table 9–74 MGMT$CSM_SUBNET_DIST_HOURLY

Column	Datatype	Description
TARGET_NAME	VARCHAR2(64)	The name of the composite target. A composite target is a grouping of one or more target name and target types from the target table.
TARGET_TYPE	VARCHAR2(64)	The type of the composite target. The type defines the set of metrics that are applicable for the target.
VISITOR_SUBNET	VARCHAR2(32)	Domain subnet of the machine making the requests.
ROLLUP_TIMESTAMP	DATE	Start of the rollup period (time zone of the target).
METRIC_NAME	VARCHAR2(64)	Name of the metric being defined.
METRIC_VALUE	NUMBER	Number of milliseconds required to access this url. This value is the difference in milliseconds between SUBMIT_ACTION_TIMESTAMP and LOAD_ACTION_TIMESTAMP

Table 9–74 (Cont.) MGMT$CSM_SUBNET_DIST_HOURLY

Column	Datatype	Description
SAMPLE_COUNT	NUMBER	The number of non-NULL samples for the metric that were aggregated.

Usage Notes

Analyze the distribution of the HTTP response times for a visitor IP subnet of a composite target during a specific hour of the day (For example, how many requests from users of the subnet 192.168.1 to the website my.oracle.com took longer than 5 seconds between 8 and 9 this morning).

9.9.28 MGMT$CSM_SUBNET_DIST_DAILY

MGMT$CSM_SUBNET_DIST_DAILY displays the distribution of the HTTP request response times, in seconds, for a visitor IP subnet of a composite target. The data is aggregated in daily time periods.

Table 9–75 MGMT$CSM_SUBNET_DIST_DAILY

Column	Datatype	Description
TARGET_NAME	VARCHAR2(64)	The name of the composite target. A composite target is a grouping of one or more target name and target types from the target table.
TARGET_TYPE	VARCHAR2(64)	The type of the composite target. The type defines the set of metrics that are applicable for the target.
VISITOR_SUBNET	VARCHAR2(32)	Domain subnet of the machine making the requests.
ROLLUP_TIMESTAMP	DATE	Start of the rollup period (time zone of the target).
METRIC_NAME	VARCHAR2(64)	Name of the metric being defined.
METRIC_VALUE	NUMBER	Number of milliseconds required to access this url. This value is the difference in milliseconds between SUBMIT_ACTION_TIMESTAMP and LOAD_ACTION_TIMESTAMP
SAMPLE_COUNT	NUMBER	The number of non-NULL samples for the metric that were aggregated.

Usage Notes

Analyze the distribution of the HTTP response times for a given visitor IP subnet of a composite target during a specific day (For example, how many requests from users of the subnet 192.168.1 to the website my.oracle.com took longer than 5 seconds last Monday).

9.9.29 MGMT$E2E_1DAY

MGMT$E2E_1DAY displays the E2E data where rollup time spans from 7 days to 31 days.

Table 9–76 MGMT$E2E_1DAY

Column	Datatype	Description
URI	VARCHAR2(2000)	URL for which the end to end trace is sampled
HITS	NUMBER(8)	No of times the uri is hit for before aggregating the sampled data.

Table 9–76 (Cont.) MGMT$E2E_1DAY

Column	Datatype	Description
TOTAL_HIT_TIME	NUMBER(10)	Total time taken for the uri to respond.
SERVLET_COUNT	NUMBER(8)	No of servlets used for serving the uri
SERVLET_TIME	NUMBER(10)	Total time taken by all servlets for serving the uri.
JSP_COUNT	NUMBER(8)	No of jsps used for sering the uri.
JSP_TIME	NUMBER(10)	Total time taken by all jsps involved in serving the uri.
EJB_COUNT	NUMBER(8)	No of EJB objects involved in serving the uri.
EJB_TIME	NUMBER(10)	Total time taken by the all the EJB objects involved in the serving the uri.
JDBC_TIME	NUMBER(10)	Total JDBC including SQL, PL/SQL times for serving the uri.
TARGET_GUID	RAW(16)	Target GUID of the web application
ROLLUP_TIMESTAMP	DATE	Since each record in the base table, mgmt_e2e_summary_1day, of this view is a rollup of daily data from mgmt_e2e_summary_1hour, this date corresponds to the time when that rollup has happened.

9.9.30 MGMT$E2E_HOURLY

MGMT$E2E_HOURLY displays E2E data where rollup time spans from 24 hours to last 7 days.

Table 9–77 MGMT$E2E_HOURLY

Column	Datatype	Description
URI	VARCHAR2(2000)	URL for which the end to end trace is sampled
HITS	NUMBER(8)	No of times the uri is hit for before aggregating the sampled data.
TOTAL_HIT_TIME	NUMBER(10)	Total time taken for the uri to respond.
SERVLET_COUNT	NUMBER(8)	No of servlets used for serving the uri
SERVLET_TIME	NUMBER(10)	Total time taken by all servlets for serving the uri.
JSP_COUNT	NUMBER(8)	No of jsps used for sering the uri.
JSP_TIME	NUMBER(10)	Total time taken by all jsps involved in serving the uri.
EJB_COUNT	NUMBER(8)	No of EJB objects involved in serving the uri.
EJB_TIME	NUMBER(10)	Total time taken by the all the EJB objects involved in the serving the uri.
JDBC_TIME	NUMBER(10)	Total JDBC including SQL, PL/SQL times for serving the uri.
TARGET_GUID	RAW(16)	Target GUID of the web application
ROLLUP_TIMESTAMP	DATE	Since each record in the base table, mgmt_e2e_summary_1day, of this view is a rollup of daily data from mgmt_e2e_summary_1hour, this date corresponds to the time when that rollup has happened.

9.9.31 MGMT$E2E_RAW

MGMT$E2E_RAW displays E2E data where rollup time is less than 24 hours.

Table 9–78 MGMT$E2E_RAW

Column	Datatype	Description
URI	VARCHAR2(2000)	URL for which the end to end trace is sampled
HITS	NUMBER(8)	No of times the uri is hit for before aggregating the sampled data.
TOTAL_HIT_TIME	NUMBER(10)	Total time taken for the uri to respond.
SERVLET_COUNT	NUMBER(8)	No of servlets used for serving the uri
SERVLET_TIME	NUMBER(10)	Total time taken by all servlets for serving the uri.
JSP_COUNT	NUMBER(8)	No of jsps used for sering the uri.
JSP_TIME	NUMBER(10)	Total time taken by all jsps involved in serving the uri.
EJB_COUNT	NUMBER(8)	No of EJB objects involved in serving the uri.
EJB_TIME	NUMBER(10)	Total time taken by the all the EJB objects involved in the serving the uri.
JDBC_TIME	NUMBER(10)	Total JDBC including SQL, PL/SQL times for serving the uri.
TARGET_GUID	RAW(16)	Target GUID of the web application
ROLLUP_TIMESTAMP	DATE	Since each record in the base table, mgmt_e2e_ summary_1day, of this view is a rollup of daily data from mgmt_e2e_summary_1hour, this date corresponds to the time when that rollup has happened.

9.10 Configuration Views

9.10.1 MGMT$DB_TABLESPACES

MGMT$DB_TABLESPACES displays configuration settings for tablespaces. Tablespace settings are collected from the sys.dba_tablespaces, dba_free_space, dba_data_files, dba_temp_files, and v$temp_extent_pool tables..

Table 9–79 MGMT$DB_TABLESPACES

Column	Datatype	Description
HOST	VARCHAR2(256)	Name of the target where the metrics will be collected.
TARGET_NAME	VARCHAR2(256)	Name of the database containing the datafiles.
TARGET_TYPE	VARCHAR2(64)	The type of target, for example, Oracle_ database.
TARGET_GUID	RAW(16)	The unique ID for the database target.
COLLECTION_ TIMESTAMP	DATE	The date and time when the metrics were collected.
TABLESPACE_NAME	VARCHAR2(30)	Name of the tablespace.
CONTENTS	VARCHAR2(9)	Tablespace contents: PERMANENT or TEMPORARY.

Table 9–79 (Cont.) MGMT$DB_TABLESPACES

Column	Datatype	Description
STATUS	VARCHAR2(10),	Tablespace status: ONLINE, OFFLINE, or READ ONLY.
EXTENT_MANAGEMENT	VARCHAR2(10),	Extent management tracking: DICTIONARY or LOCAL.
ALLOCATION_TYPE	VARCHAR2(10),	Type of extent allocation in effect for this tablespace.
LOGGING	VARCHAR2(10),	Default logging attribute.
TABLESPACE_SIZE	NUMBER	Current size of the tablespace in bytes.
INITIAL_EXT_SIZE	NUMBER	Default initial extent size
NEXT_EXTENT	NUMBER	Next extent in the sequence.
INCREMENT_BY	NUMBER	Default percent increase for extent size.
MAX_EXTENTS	NUMBER	Default maximum number of extents
TABLESPACE_USED_SIZE	NUMBER	Amount of data (in bytes) contained in the tablespace.
SEGMENT_SPACE_ MANAGEMENT	VARCHAR2(6)	Indicates whether the free and used segment space in the tablespace is managed using free lists (MANUAL) or bitmaps (AUTO)
BLOCK_SIZE	NUMBER	Tablespace block size
MIN_EXTENTS	NUMBER	Default minimum number of extents
MIN_EXTLEN	NUMBER	Minimum extent size for this tablespace
BIGFILE	VARCHAR2(3)	Indicates whether the tablespace is a bigfile tablespace (YES) or a smallfile tablespace (NO)

Usage Notes

Obtain control file configuration settings across all managed database targets.

9.10.2 MGMT$DB_DATAFILES

MGMT$DB_DATAFILES displays the configuration settings for datafiles. The datafile settings are collected from sources such as sys.dba_data_files, v$datafile, sys.dba_free_ space, sys.dba_tablespaces, sys.dba_ temp_files, v$tempfile.

Table 9–80 MGMT$DB_DATAFILES

Column	Datatype	Description
HOST	VARCHAR2(256)	Name of the target where the metrics will be collected.
TARGET_NAME	VARCHAR2(256)	Name of the database containing the datafiles.
TARGET_TYPE	VARCHAR2(64)	The type of target, for example, Oracle_ database.
TARGET_GUID	RAW(16)	The unique ID for the database target.
COLLECTION_ TIMESTAMP	DATE	The date and time when the metrics were collected.
FILE_NAME	VARCHAR2(512)	Name of the datafile.
TABLESPACE_NAME	VARCHAR2(30)	Name of the tablespace containing
STATUS	VARCHAR2(10)	Datafile status: ACTIVE or NOT ACTIVE
FILE_SIZE	NUMBER	Size of the datafile

Table 9–80 (Cont.) MGMT$DB_DATAFILES

Column	Datatype	Description
AUTOEXTENSIBLE	VARCHAR2(3)	Autoextensible indicator
INCREMENT_BY	NUMBER	Autoextension increment.
MAX_FILE_SIZE	NUMBER	Maximum file size in bytes
OS_STORAGE_ENTITY	VARCHAR2(512)	OS level storage entity on which the file resides. For regular files it is the name of the filesystem on which the file resides. For character or raw files it is the name of the raw device

9.10.3 MGMT$DB_CONTROLFILES

MGMT$DB_CONTROLFILES displays the configuration settings for database control files.

Table 9–81 MGMT$DB_CONTROLFILES

Column	Datatype	Description
HOST	VARCHAR2(256)	Name of the target where the metrics will be collected.
TARGET_NAME	VARCHAR2(256)	Name of the database containing the datafiles.
TARGET_TYPE	VARCHAR2(64)	The type of target, for example, Oracle_ database.
TARGET_GUID	RAW(16)	The unique ID for the database target.
COLLECTION_ TIMESTAMP	DATE	The date and time when the metrics were collected.
FILE_NAME	VARCHAR2(512)	Name of the database control file.
STATUS	VARCHAR2(10),	The type of control file: STANDBY - indicates database is in standby mode LOGICAL - indicates the database is a logical standby database (not a physical standby) CLONE - indicates a clone database BACKUP | CREATED - indicates database is being recovered using a backup or created control file. CURRENT - the control file changes to this type following a standby database activate or database open after recovery.
CREATION_DATE	DATE	Control file creation date.
SEQUENCE_NUM	NUMBER	Control file sequence number incremented by control file transactions.
CHANGE_NUM	NUMBER	Last change number in the backup control file. Value is NULL if the control file is not a backup.
MOD_DATE	DATE	Last timestamp in the backup control file. NULL if the control file is not a backup.
OS_STORAGE_ENTITY	VARCHAR2(512)	OS level storage entity on which the file resides. For regular files it is the name of the filesystem on which the file resides. For character or raw files it is the name of the raw device

9.10.4 MGMT$DB_DBNINSTANCEINFO

MGMT$DB_DBNINSTANCEINFO displays general information about database instance. The instance information is collected from v$database, v$version, v$instance, global_name, database_properties and v$nls_parameters.

Table 9–82 MGMT$DB_DBNINSTANCEINFO

Column	Datatype	Description
HOST_NAME	VARCHAR2(256)	Name of the target host where the metrics will be collected.
TARGET_NAME	VARCHAR2(256)	Name of the database target from which the metrics are collected.
TARGET_TYPE	VARCHAR2(64)	The type of target, for example, Oracle_ database.
TARGET_GUID	RAW(16)	The unique ID for the database target.
COLLECTION_ TIMESTAMP	DATE	The date and time when the metrics were collected.
DATABASE_NAME	VARCHAR2(9)	Name of the database.
GLOBAL_NAME	VARCHAR2(4000)	Global name of the database
BANNER	VARCHAR2(64)	Component name and version number
HOST	VARCHAR2(64)	Name of the host machine.
INSTANCE_NAME	VARCHAR2(16)	Name of the instance.
STARTUP_TIME	DATE	Time when instance was started up.
LOGINS	VARCHAR2(10)	ALLOWED or RESTRICTED
LOG_MODE	VARCHAR2(12)	The archive log mode, either ARCHIVELOG or NOARCHIVELOG.
OPEN_MODE	VARCHAR2(10)	Open mode information.
DEFAULT_TEMP_TABLESPACE	VARCHAR2(30)	Default temporary tablespace name.
CHARACTERSET	VARCHAR2(64)	NLS parameter value for NLS_ CHARACTERSET
NATIONAL_CHARACTERSET	VARCHAR2(64)	NLS parameter value for NLS_NCHAR_ CHARACTERSET

Usage Notes

Obtain general instance information across all database targets.

9.10.5 MGMT$DB_FEATUREUSAGE

MGMT$DB_FEATUREUSAGE displays information about database feature usage.

Table 9–83 MGMT$DB_FEATUREUSAGE

Column	Datatype	Description
HOST	VARCHAR2(256)	Name of the host target where the database feature usage information is collected.
DATABASE_NAME	VARCHAR2(256)	Name of the database where the database feature usage information is collected.
INSTANCE_NAME	VARCHAR2(16)	Name of the instance where the database feature usage information is collected.
TARGET_TYPE	VARCHAR2(64)	Either Oracle_database or rac_database.

Table 9–83 (Cont.) MGMT$DB_FEATUREUSAGE

Column	Datatype	Description
DBID	NUMBER	A unique number that identifies a database instance.
NAME	VARCHAR2(64)	The feature name.
CURRENTLY_USED	VARCHAR2(5)	TRUE if the feature is currently in use, FALSE if the feature is not in use.
DETECTED_USAGES	NUMBER	The number of times the feature has been used by the database.
FIRST_USAGE_DATE	DATE	The date that the first usage of the feature occurred.
LAST_USAGE_DATE	DATE	The date of the most recent usage of the feature.
VERSION	VARCHAR2(17)	The version number of the database.
LAST_SAMPLE_DATE	DATE	The date that the database was last evaluated for feature usage.
LAST_SAMPLE_PERIOD	NUMBER	The interval between the LAST_ SAMPLE_ DATE date and the database feature usage evaluation before that (by default, seven days).
SAMPLE_INTERVAL	NUMBER	The number of seconds between the LAST_ SAMPLE_DATE date and the next database feature usage evaluation.
TOTAL_SAMPLES	NUMBER	The total number of database feature usage evaluation samples that have been collected.
AUX_COUNT	NUMBER	For Oracle internal use only.
DESCRIPTION	VARCHAR2(128)	The description of the feature.

Usage Notes

This view can be used to gain an enterprise-wide view of database feature usage across all Oracle databases.

9.10.6 MGMT$DB_INIT_PARAMS

MGMT$DB_INIT_PARAMS displays initialization parameter settings for the database. Initialization parameter settings are collected from v$parameter.

Table 9–84 MGMT$DB_INIT_PARAMS

Column	Datatype	Description
HOST_NAME	VARCHAR2(256)	Name of the target where the metrics will be collected.
TARGET_NAME	VARCHAR2(256)	Name of the database target from which the metrics are collected.
TARGET_TYPE	VARCHAR2(64)	The type of target, for example, Oracle_ database.
TARGET_GUID	RAW(16)	The unique ID for the database target.
COLLECTION_ TIMESTAMP	DATE	The date and time when the metrics were collected.
NAME	VARCHAR2(64)	Name of the initialization parameter.
ISDEFAULT	VARCHAR2(6)	Indicates whether the parameter value is the default.
VALUE	VARCHAR2(512)	The parameter value.

Table 9–84 (Cont.) MGMT$DB_INIT_PARAMS

Column	Datatype	Description
DATATYPE	VARCHAR2	The data type that the value string can be mapped to, for example, NUMBER, DATE, or TEXT.'

Usage Notes

Obtain initialization parameter settings across all database targets.

9.10.7 MGMT$DB_LICENSE

MGMT$DB_LICENSE displays database license configuration settings. Database license configuration settings are collected from v$license.

Table 9–85 MGMT$DB_LICENSE

Column	Datatype	Description
HOST_NAME	VARCHAR2(256)	The name of the host on which the database is running.
TARGET_NAME	VARCHAR2(256)	Name of the database containing the tablespace.
TARGET_TYPE	VARCHAR2(64)	The type of target, for example, Oracle_ database.
TARGET_GUID	RAW(16)	The unique ID for the database target.
COLLECTION_ TIMESTAMP	DATE	The date and time when the metrics were collected.
SESSIONS_MAX	NUMBER	The maximum number of sessions allowed for the database.
SESSIONS_WARNING	NUMBER	The number of sessions which will generate a warning for the database.
SESSIONS_CURRENT	NUMBER	The current number of sessions for the database.
SESSIONS_HIGHWATER	NUMBER	The highest water mark of sessions for the database.
USERS_MAX	NUMBER	The maximum number of users for the database.

Usage Notes

This view can be used to obtain database license configuration settings across all database targets.

9.10.8 MGMT$DB_REDOLOGS

MGMT$DB_REDOLOGS displays redo log configuration settings for the database. Redo log configuration settings are collected from the v$log and v$logfile tables.

Table 9–86 MGMT$DB_REDOLOGS

Column	Datatype	Description
HOST_NAME	VARCHAR2(256)	Name of the target where the metrics will be collected.
TARGET_NAME	VARCHAR2(256)	Name of the database target from which the metrics are collected.
TARGET_TYPE	VARCHAR2(64)	The type of target, for example, Oracle_ database.

Table 9–86 (Cont.) MGMT$DB_REDOLOGS

Column	Datatype	Description
TARGET_GUID	RAW(16)	The unique ID for the database target.
COLLECTION_ TIMESTAMP	DATE	The date and time when the metrics were collected.
GROUP_NUM	NUMBER	Redo log group identifier number.
STATUS	VARCHAR2(16),	og status: UNUSED - The online redo log has never been written to. This is the state of a redo log that was just added, or just after a RESETLOGS, when it is not the current redo log. CURRENT - This is the current redo log. This implies that the redo log is active. The redo log could be open or closed. ACTIVE - The log is active but is not the current log. It is needed for crash recovery. It may be in use for block recovery. It might or might not be archived. CLEARING - The log is being re-created as an empty log after an ALTER DATABASE CLEAR LOGFILE statement. After the log is cleared, the status changes to UNUSED. CLEARING_CURRENT - The current log is being cleared of a closed thread. The log can stay in this status if there is some failure in the switch such as an I/O error writing the new log header. INACTIVE - The log is no longer needed for instance recovery. It may be in use for media recovery. It might or might not be archived.
MEMBERS	NUMBER	Number of members in the log group
FILE_NAME	VARCHAR2(513),	Redo log file (member) name.
ARCHIVED	VARCHAR2(3),	Archive status either YES or NO.
LOGSIZE	NUMBER	Size of the log file in bytes
SEQUENCE_NUM	NUMBER	Log sequence number
FIRST_CHANGE_SCN	NUMBER	Lowest SCN in the log.
OS_STORAGE_ENTITY	VARCHAR2(512)	OS level storage entity on which the file resides. For regular files it is the name of the filesystem on which the file resides. For character or raw files it is the name of the raw device.
THREAD_NUM	NUMBER	Log thread number.

Usage Notes

Obtain redo log group / file configuration settings across all database targets.

9.10.9 MGMT$DB_ROLLBACK_SEGS

MGMT$DB_ROLLBACK_SEGS displays rollback segments configuration settings for the database. Rollback segments configuration settings are collected from the sys.dba_rollback_segs and v$rollstat tables.

Table 9–87 MGMT$DB_ROLLBACK_SEGS

Column	Datatype	Description
HOST_NAME	VARCHAR2	Name of the target where the metrics will be collected.
TARGET_NAME	VARCHAR2(256)	Name of the database containing the datafiles.
TARGET_TYPE	VARCHAR2(64)	The type of target, for example, Oracle_ database.
TARGET_GUID	RAW(16)	The unique ID for the database target.
COLLECTION_ TIMESTAMP	DATE	The date and time when the metrics were collected.
ROLLNAME	VARCHAR2(64))	Name of the rollback segment
STATUS	VARCHAR2(10)	Rollback segment status
TABLESPACE_NAME	VARCHAR2(30)	Name of the tablespace containing the rollback segment.
EXTENTS	NUMBER	Number of extents in rollback segment.
ROLLSIZE	NUMBER	Size in bytes of rollback segment. This values differs by the number of bytes in one database block from the value of the BYTES column of the ALL/DBA/USER_ SEGMENTS views.
INITIAL_SIZE	NUMBER	Initial extent size in bytes.
NEXT_SIZE	NUMBER	Secondary extent size in bytes.
MAXIMUM_EXTENTS	NUMBER	Maximum number of extents.
MINIMUM_EXTENTS	NUMBER	Minimum number of extents.
PCT_INCREASE	NUMBER	Percent increase for extent size.
OPTSIZE	NUMBER	Optimal size for rollback segments.
AVEACTIVE	NUMBER	Current size of active extents averaged over time.
WRAPS	NUMBER	Number of times rollback segment is wrapped.
SHRINKS	NUMBER	Number of times the size of a rollback segment decreases.
AVESHRINK	NUMBER	Average shrink size.
HWMSIZE	NUMBER	High water mark of rollback segment size.

Usage Notes

Obtain rollback segments configuration settings across all database targets.

9.10.10 MGMT$DB_SGA

MGMT$DB_SGA displays System Global Area (SGA) configuration settings. SGA settings are collected from the v$sga and v$sgastat tables.

Table 9–88 MGMT$DB_SGA

Column	Datatype	Description
HOST_NAME	VARCHAR2(256)	Name of the target where the metrics will be collected.
TARGET_NAME	VARCHAR2(256)	Name of the database containing the datafiles.
TARGET_TYPE	VARCHAR2(64)	The type of target, for example, Oracle_ database.

Table 9–88 (Cont.) MGMT$DB_SGA

Column	Datatype	Description
TARGET_GUID	RAW(16)	The unique ID for the database target
COLLECTION_ TIMESTAMP	DATE	The date and time when the metrics were collected.
SGANAME	VARCHAR2(64)	SGA component name.
SGASIZE	NUMBER	SGA component size in kilobytes or megabytes

Usage Notes

Obtain System Global Area configuration settings across all database targets.

9.10.11 MGMT$DB_TABLESPACES

MGMT$DB_TABLESPACES displays configuration settings for tablespaces. Tablespace settings are collected from the sys.dba_tablespaces, dba_free_space, dba_data_files, dba_temp_files, and v$temp_extent_pool tables.

Table 9–89 MGMT$DB_TABLESPACES

Column	Datatype	Description
HOST	VARCHAR2(256)	Name of the target where the metrics will be collected.
TARGET_NAME	VARCHAR2(64)	The type of target, for example, Oracle_ database.
TARGET_GUID	RAW(16)	The unique ID for the database target.
COLLECTION_ TIMESTAMP	DATE	The date and time when the metrics were collected.
TABLESPACE_NAME	VARCHAR2(30)	Name of the tablespace.
CONTENTS	VARCHAR2(9)	Tablespace contents: PERMANENT or TEMPORARY.
STATUS	VARCHAR2(10),	Tablespace status: ONLINE, OFFLINE, or READ ONLY.
EXTENT_MANAGEMENT	VARCHAR2(10),	Extent management tracking: DICTIONARY or LOCAL.
ALLOCATION_TYPE	VARCHAR2(10),	Type of extent allocation in effect for this tablespace.
LOGGING	VARCHAR2(10),	Default logging attribute.
TABLESPACE_SIZE	NUMBER	Current size of the tablespace in bytes.
INITIAL_EXT_SIZE	NUMBER	Default initial extent size
INCREMENT_BY	NUMBER	Default percent increase for extent size.
MAX_EXTENTS	NUMBER	Default maximum number of extents

Usage Notes

Obtain tablespace configuration settings across all database targets.

9.10.12 MGMT$DB_OPTIONS

MGMT$DB_OPTIONS displays whether or not the option is currently LOADED and ACTIVE, or either the option does not exist or is NOT LOADED or INACTIVE.

Options settings are collected by checking user name and status in the sys.dba_users and dba_registry tables.

Table 9–90 MGMT$DB_OPTIONS

Column	Datatype	Description
HOST	VARCHAR2(256)	Name of the target where the metrics will be collected.
TARGET_NAME	VARCHAR2(256)	Name of the database containing the datafiles.
TARGET_TYPE	VARCHAR2(64)	The type of target, for example, Oracle_ database.
TARGET_GUID	RAW(16)	The unique ID for the database target.
COLLECTION_ TIMESTAMP	DATE	The date and time when the metrics were collected.
NAME	VARCHAR2(30)	Name of the database option.
SELECTED	VARCHAR2(5)	If the option is currently LOADED and ACTIVE (TRUE), or either the option does not exist or is NOT LOADED or INACTIVE (FALSE)

Usage Notes

Obtain tablespace configuration settings across all database targets.

9.11 Oracle Home Patching Views

9.11.1 MGMT$EM_HOMES_PLATFORM

MGMT$EM_HOMES_PLATFORM displays the platform information about the Homes. If the home does not have an ARU platform id, then the platform of the Operating System is considered as the Platform of the home.

Table 9–91 MGMT$EM_HOMES_PLATFORM

Column	Datatype	Description
HOME_ID	RAW(16)	Unique id for home
PLATFORM_ID	NUMBER(10)	If the home has an ARU platform it is used, else the platform id of the host is picked
PLATFORM	VARCHAR2(1024)	The platform corresponding to the platform_id

9.11.2 MGMT$HOMES_AFFECTED

MGMT$HOMES_AFFECTED displays the list of homes, vulnerable to bugs, which are fixed by the critical patches released. The number of alerts which are applicable to the home are calculated.

Table 9–92 MGMT$HOMES_AFFECTED

Column	Datatype	Description
HOST	VARCHAR2(1024))	Host name
HOME_DIRECTORY	VARCHAR2(1024)	Home directory location
TARGET_GUID	RAW(16)	Unique id for target
ALERTS	NUMBER	Number of alerts for this home

9.11.3 MGMT$APPL_PATCH_AND_PATCHSET

MGMT$APPL_PATCH_AND_PATCHSET displays the list of interim patches and patchsets that are applicable to the Homes.

Table 9–93 MGMT$APPL_PATCH_AND_PATCHSET

Column	Datatype	Description
PATCH_ID	NUMBER	The patch Id
TYPE	VARCHAR2(32)	Patch/Patchset
PRODUCT	VARCHAR2(50)	The product on pertaining to the patch.
PATCH_RELEASE	VARCHAR2(30)	Release version.
PLATFORM	VARCHAR2(40)	Platform on which patch is applicable
ADVISORY	VARCHAR2(256)	The alert name.
HOST_NAME	VARCHAR2(256)	Host name
HOME_LOCATION	VARCHAR2(128)	Home directory location.
PATCH_GUID	RAW(16)	Unique id for the patch/patchset.
TARGET_GUID	RAW(16)	Unique id for target.

9.11.4 MGMT$APPLIED_PATCHES

MGMT$APPLIED_PATCHES displays the list of patches that have been applied on the homes along with the installation time. Each patch can fix more than one bug. The bugs are listed in a comma-separated string.

Table 9–94 MGMT$APPLIED_PATCHES

Column	Datatype	Description
PATCH	VARCHAR2(128)	Patch name.
BUGS	VARCHAR2(256)	The bugs fixed by this patch.
INSTALLATION_TIME	DATE	Time of Installation. Timezone of the target.
HOST	VARCHAR2(256)	Host name.
HOME_LOCATION	VARCHAR2(128)	Home location.
HOME_NAME	VARCHAR2(64)	Name of the home.
CONTAINER_GUID	RAW(16)	Name of the home.
TARGET_GUID	RAW(16)	Unique id for target.

9.11.5 MGMT$APPLIED_PATCHSETS

MGMT$APPLIED_PATCHSETS displays the list of patchsets that have been applied on the Homes along with the installation time.

Table 9–95 MGMT$APPLIED_PATCHSETS

Column	Datatype	Description
VERSION	VARCHAR2(64)	The version to which the home will get upgraded to when this patchset is applied
NAME	VARCHAR2(128)	Patchset external name
TIMESTAMP	DATE	Time of Installation.Timezone of the target
HOST	VARCHAR2(256)	Host name.

Table 9–95 (Cont.) MGMT$APPLIED_PATCHSETS

Column	Datatype	Description
HOME_LOCATION	VARCHAR2(128)	Home location.
HOME_NAME	VARCHAR2(64)	Name of the home.
CONTAINER_GUID	RAW(16)	Name of the home.
TARGET_GUID	RAW(16)	Unique id for target.

9.12 Linux Patching Views

9.12.1 MGMT$HOSTPATCH_HOSTS

MGMT$HOSTPATCH_HOSTS displays information required to generate compliance reports.

Table 9–96 MGMT$HOSTPATCH_HOSTS

Column	Datatype	Description
HOST_NAME	VARCHAR2(256)	Host name
GROUP_NAME	VARCHAR2(256)	The group the host belongs to.
OUT_OF_DATE_PACKAGES	NUMBER(6)	Number of Packages which have a newer version available
ROGUE_PACKAGES	NUMBER(6)	The packages that are not supposed to be installed on the host

9.12.2 MGMT$HOSTPATCH_GROUPS

MGMT$HOSTPATCH_GROUPS displays additional information about a group, the maturity level which is set by the administrator and the packages which need the host to be rebooted on application.

Table 9–97 MGMT$HOSTPATCH_GROUPS

Column	Datatype	Description
GROUP_NAME	VARCHAR2(256)	The (unique) name of the Group
MATURITY_LEVEL	VARCHAR2(32)	The maturity level of the group. This is set by the administrator.
NEED_REBOOT_PKGS	VARCHAR2(256)	Comma seperated list of packages which need the machine to be rebooted on application

9.12.3 MGMT$HOSTPATCH_GRP_COMPL_HIST

MGMT$HOSTPATCH_GRP_COMPL_HIST displays information required to generate compliance history reports.

Table 9–98 MGMT$HOSTPATCH_GRP_COMPL_HIST

Column	Datatype	Description
GROUP_NAME	VARCHAR2(256)	Name of the Group
TOTAL_HOSTS	NUMBER(6)	Number of hosts in the Group
COMPLIANT_HOSTS	NUMBER(6)	Number of compliant Hosts in the Group.
LAST_CHECKED_ON	DATE	Date on which this record was collected.

9.12.4 MGMT$HOSTPATCH_HOST_COMPL

MGMT$HOSTPATCH_HOST_COMPL displays information required to generate advisory reports.

Table 9–99 MGMT$HOSTPATCH_HOST_COMPL

Column	Datatype	Description
HOST_NAME	VARCHAR2(256)	Host name
PKG_NAME	VARCHAR2(256)	Package name
VERSION	VARCHAR2(64)	Version of the package
IS_OUT_OF_DATE	NUMBER(1)	If out of date
IS_ROGUE	NUMBER(1)	If it is rogue

9.13 Security Views

9.13.1 MGMT$ESA_ALL_PRIVS_REPORT

MGMT$ESA_ALL_PRIVS_REPORT displays a table containing users and roles that have the 'GRANT ANY' privilege in database security reports.

Table 9–100 MGMT$ESA_ALL_PRIVS_REPORT

Column	Datatype	Description
TARGET_GUID	RAW(16)	The GUID of the target for which the report has the data
TARGET_NAME	VARCHAR2(256)	The name of the target for which the report has the data
PRINCIPAL	VARCHAR2(512)	The user or role which has been granted a privilege which amounts to all privileges on the database.
OBJECT_NAME	VARCHAR2(512)	The privilege (GRANT ANY PRIVILEGE) if granted directly, or the role through it has been granted.

9.13.2 MGMT$ESA_ANY_DICT_REPORT

MGMT$ESA_ANY_DICT_REPORT displays a table and a chart containing users and roles with access to any dictionary in database security reports.

Table 9–101 MGMT$ESA_ANY_DICT_REPORT

Column	Datatype	Description
TARGET_GUID	RAW(16)	The GUID of the target for which the report has the data.
TARGET_NAME	VARCHAR2(256)	The name of the target for which the report has the data.
PRINCIPAL	VARCHAR2(512)	The user or role which has been granted a privilege which provides it access to any dictionary in the database.
OBJECT_NAME	VARCHAR2(512)	The privilege if granted directly, or the role through it has been granted.

9.13.3 MGMT$ESA_ANY_PRIV_REPORT

MGMT$ESA_ANY_PRIV_REPORT displays a table and a chart containing users with 'ANY' in some privilege granted to them in database security reports.

Table 9–102 MGMT$ESA_ANY_PRIV_REPORT

Column	Datatype	Description
TARGET_GUID	RAW(16)	The GUID of the target for which the report has the data.
TARGET_NAME	VARCHAR2(256)	The name of the target for which the report has the data.
PRINCIPAL	VARCHAR2(512)	The user or role which has been granted a privilege having 'ANY'
OBJECT_NAME	VARCHAR2(512)	The privilege if granted directly, or the role through it has been granted.

9.13.4 MGMT$ESA_AUDIT_SYSTEM_REPORT

MGMT$ESA_AUDIT_SYSTEM_REPORT displays a table containing users and roles with the 'AUDIT SYSTEM' privilege in database security reports.

Table 9–103 MGMT$ESA_AUDIT_SYSTEM_REPORT

Column	Datatype	Description
TARGET_GUID	RAW(16)	The GUID of the target for whichthe report has the data
TARGET_NAME	VARCHAR2(256)	The name of the target for which the report has the data.
PRINCIPAL	VARCHAR2(512)	The user or role which has been granted 'AUDIT SYSTEM' privilege.
OBJECT_NAME	VARCHAR2(512)	The privilege if granted directly, or the role through it has been granted.

9.13.5 MGMT$ESA_BECOME_USER_REPORT

MGMT$ESA_BECOME_USER_REPORT displays a table containing users and roles with the 'BECOME USER' privilege in database security reports.

Table 9–104 MGMT$ESA_BECOME_USER_REPORT

Column	Datatype	Description
TARGET_GUID	RAW(16)	The GUID of the target for which the report has the data
TARGET_NAME	VARCHAR2(256)	The name of the target for which the report has the data
PRINCIPAL	VARCHAR2(512)	The user or role which has been granted 'BECOME USER' privilege.
OBJECT_NAME	VARCHAR2(512)	The privilege if granted directly, or the role through it has been granted.

9.13.6 MGMT$ESA_CATALOG_REPORT

MGMT$ESA_CATALOG_REPORT displays a table and a chart containing all the users that have a role such as '%CATALOG%' in database security reports.

Table 9–105 MGMT$ESA_CATALOG_REPORT

Column	Datatype	Description
TARGET_GUID	RAW(16)	The GUID of the target for which the report has the data.
TARGET_NAME	VARCHAR2(256)	The name of the target for which the report has the data.
PRINCIPAL	VARCHAR2(512)	The user or role which has been granted a role like '%CATALOG%'
OBJECT_NAME	VARCHAR2(512)	The role if granted directly, or the role through it has been granted.

9.13.7 MGMT$ESA_CONN_PRIV_REPORT

MGMT$ESA_CONN_PRIV_REPORT displays a table and a chart containing users and roles with the CONNECT or RESOURCE role in database security reports.

Table 9–106 MGMT$ESA_CONN_PRIV_REPORT

Column	Datatype	Description
TARGET_GUID	RAW(16)	The GUID of the target for which the report has the data.
TARGET_NAME	VARCHAR2(256)	The name of the target for which the report has the data.
PRINCIPAL	VARCHAR2(512)	The user or role which has been granted the CONNECT or RESOURCE role.
OBJECT_NAME	VARCHAR2(512)	The role if granted directly, or the role through it has been granted.

9.13.8 MGMT$ESA_CREATE_PRIV_REPORT

MGMT$ESA_CREATE_PRIV_REPORT displays a table and a chart containing users and roles with the CREATE privilege in database security reports.

Table 9–107 MGMT$ESA_CREATE_PRIV_REPORT

Column	Datatype	Description
TARGET_GUID	RAW(16)	The GUID of the target for which the report has the data.
TARGET_NAME	VARCHAR2(256)	The name of the target for which the report has the data.
PRINCIPAL	VARCHAR2(512)	The user or role which has been granted the privilege to create an object in the database.
OBJECT_NAME	VARCHAR2(512)	The role if granted directly, or the role through it has been granted.

9.13.9 MGMT$ESA_DBA_GROUP_REPORT

MGMT$ESA_DBA_GROUP_REPORT displays a table containing members of the operating system user group DBA in database security reports.

Table 9–108 MGMT$ESA_DBA_GROUP_REPORT

Column	Datatype	Description
TARGET_GUID	RAW(16)	The GUID of the target for which the report has the data.

Table 9–108 (Cont.) MGMT$ESA_DBA_GROUP_REPORT

Column	Datatype	Description
TARGET_NAME	VARCHAR2(256)	The name of the target for which the report has the data.
PRINCIPAL	VARCHAR2(512)	The operating system user which is in the user group DBA.
OBJECT_NAME	VARCHAR2(512)	DBA Group

9.13.10 MGMT$ESA_DBA_ROLE_REPORT

MGMT$ESA_DBA_ROLE_REPORT displays a table containing users and roles with the DBA role granted to them in database security reports.

Table 9–109 MGMT$ESA_DBA_ROLE_REPORT

Column	Datatype	Description
TARGET_GUID	RAW(16)	The GUID of the target for which the report has the data.
TARGET_NAME	VARCHAR2(256)	The name of the target for which the report has the data.
PRINCIPAL	VARCHAR2(512)	The user or role which has been granted the DBA Role.
OBJECT_NAME	VARCHAR2(512)	The role if granted directly, or the role through it has been granted.

9.13.11 MGMT$ESA_DIRECT_PRIV_REPORT

MGMT$ESA_DIRECT_PRIV_REPORT displays a table and a chart containing privileges granted directly in database security reports.

Table 9–110 MGMT$ESA_DIRECT_PRIV_REPORT

Column	Datatype	Description
TARGET_GUID	RAW(16)	The GUID of the target for which the report has the data.
TARGET_NAME	VARCHAR2(256)	The name of the target for which the report has the data.
PRINCIPAL	VARCHAR2(512)	User which has been granted a privilege directly i.e. not via a role.
OBJECT_NAME	VARCHAR2(512)	The privilege that has been granted directly..

9.13.12 MGMT$ESA_EXMPT_ACCESS_REPORT

MGMT$ESA_EXMPT_ACCESS_REPORT displays a table containing users and roles with the EXEMPT ACCESS POLICY privilege in database security reports.

Table 9–111 MGMT$ESA_EXMPT_ACCESS_REPORT

Column	Datatype	Description
TARGET_GUID	RAW(16)	The GUID of the target for which the report has the data.
TARGET_NAME	VARCHAR2(256)	The name of the target for which the report has the data.
PRINCIPAL	VARCHAR2(512)	The user or role which has been granted the 'EXEMPT ACCESS POLICY' privilege.

Table 9–111 (Cont.) MGMT$ESA_EXMPT_ACCESS_REPORT

Column	Datatype	Description
OBJECT_NAME	VARCHAR2(512)	The privilege if granted directly, or the role through it has been granted.

9.13.13 MGMT$ESA_KEY_OBJECTS_REPORT

MGMT$ESA_KEY_OBJECTS_REPORT displays a table and a chart containing users and roles with access to key objects in database security reports.

Table 9–112 MGMT$ESA_KEY_OBJECTS_REPORT

Column	Datatype	Description
TARGET_GUID	RAW(16)	The GUID of the target for which the report has the data.
TARGET_NAME	VARCHAR2(256)	The name of the target for which the report has the data.
USER	VARCHAR2(512)	The user which has access to key objects.
OBJECT_NAME	VARCHAR2(23)	The key object to which that use has access.
PRIVILEGE	VARCHAR2(512)	The privilege on the key object that has been granted to the user.

9.13.14 MGMT$ESA_OH_OWNERSHIP_REPORT

MGMT$ESA_OH_OWNERSHIP_REPORT displays a table containing file ownership by Oracle Home in database security reports.

Table 9–113 MGMT$ESA_OH_OWNERSHIP_REPORT

Column	Datatype	Description
TARGET_GUID	RAW(16)	The GUID of the target for which the report has the data.
TARGET_NAME	VARCHAR2(256)	The name of the target for which the report has the data.
PRINCIPAL	VARCHAR2(512)	The file whose owner is not the ORACLE HOME owner.
OBJECT_NAME	VARCHAR2(512)	The owner of the file.

9.13.15 MGMT$ESA_OH_PERMISSION_REPORT

MGMT$ESA_OH_PERMISSION_REPORT displays a table containing file permissions by Oracle Home in database security reports.

Table 9–114 MGMT$ESA_OH_PERMISSION_REPORT

Column	Datatype	Description
TARGET_GUID	RAW(16)	The GUID of the target for which the report has the data.
TARGET_NAME	VARCHAR2(256)	The name of the target for which the report has the data.
PRINCIPAL	VARCHAR2(512)	The file that has an insecure permission.
OBJECT_NAME	VARCHAR2(512)	The permission of the file.

9.13.16 MGMT$ESA_POWER_PRIV_REPORT

MGMT$ESA_POWER_PRIV_REPORT displays a table and a chart containing all the users and roles with ALTER SESSION, ALTER SYSTEM, CREATE PROCEDURE or CREATE LIBRARY privileges in database security reports.

Table 9–115 MGMT$ESA_POWER_PRIV_REPORT

Column	Datatype	Description
TARGET_GUID	RAW(16)	The GUID of the target for which the report has the data.
TARGET_NAME	VARCHAR2(256)	The name of the target for which the report has the data.
PRINCIPAL	VARCHAR2(512)	The user that has powerful privileges.
OBJECT_NAME	VARCHAR2(512)	The powerful privilege held by the user.

9.13.17 MGMT$ESA_PUB_PRIV_REPORT

MGMT$ESA_PUB_PRIV_REPORT displays a table and a chart containing privileges granted to PUBLIC in database security reports.

Table 9–116 MGMT$ESA_PUB_PRIV_REPORT

Column	Datatype	Description
TARGET_GUID	RAW(16)	The GUID of the target for which the report has the data.
TARGET_NAME	VARCHAR2(256)	The name of the target for which the report has the data.
PRINCIPAL	VARCHAR2(512)	The object on which some privilege has been granted to PUBLIC.
OBJECT_NAME	VARCHAR2(512)	The privilege on the object which has been granted to PUBLIC.

9.13.18 MGMT$ESA_SYS_PUB_PKG_REPORT

MGMT$ESA_SYS_PUB_PKG_REPORT displays a table containing system packages with public execute privileges in database security reports.

Table 9–117 MGMT$ESA_SYS_PUB_PKG_REPORT

Column	Datatype	Description
TARGET_GUID	RAW(16)	The GUID of the target for which the report has the data.
TARGET_NAME	VARCHAR2(256)	The name of the target for which the report has the data.
PRINCIPAL	VARCHAR2(512)	PUBLIC
OBJECT_NAME	VARCHAR2(512)	The package owned by SYS on which PUBLIC has execute privileges.

9.13.19 MGMT$ESA_TABSP_OWNERS_REPORT

MGMT$ESA_TABSP_OWNERS_REPORT displays a table containing tablespaces and their owners in database security reports.

Table 9–118 MGMT$ESA_TABSP_OWNERS_REPORT

Column	Datatype	Description
TARGET_GUID	RAW(16)	The GUID of the target for which the report has the data.
TARGET_NAME	VARCHAR2(256)	The name of the target for which the report has the data.
PRINCIPAL	VARCHAR2(512)	The tablespace.
OBJECT_NAME	VARCHAR2(512)	The owner of the tablespace.

9.13.20 MGMT$ESA_TRC_AUD_PERM_REPORT

MGMT$ESA_TRC_AUD_PERM_REPORT displays a table containing trace and audit files permissions in database security reports.

Table 9–119 MGMT$ESA_TRC_AUD_PERM_REPORT

Column	Datatype	Description
TARGET_GUID	RAW(16)	The GUID of the target for which the report has the data.
TARGET_NAME	VARCHAR2(256)	The name of the target for which the report has the data.
PRINCIPAL	VARCHAR2(512)	The file path.
OBJECT_NAME	VARCHAR2(512)	The purpose of the file.
PERMISSION	VARCHAR2(512)	Permission of the file.

9.13.21 MGMT$ESA_WITH_ADMIN_REPORT

MGMT$ESA_WITH_ADMIN_REPORT displays a table and a chart containing users and roles having some privileges granted to them with the WITH ADMIN option in database security reports.

Table 9–120 MGMT$ESA_WITH_ADMIN_REPORT

Column	Datatype	Description
TARGET_GUID	RAW(16)	The GUID of the target for which the report has the data.
TARGET_NAME	VARCHAR2(256)	The name of the target for which the report has the data.
PRINCIPAL	VARCHAR2(512)	The user or role which has been granted a privilege 'WITH ADMIN' option.
OBJECT_NAME	VARCHAR2(512)	The privilege which has been granted 'WITH ADMIN' option.

9.13.22 MGMT$ESA_WITH_GRANT_REPORT

MGMT$ESA_WITH_GRANT_REPORT displays a table and a chart containing users and roles having some privileges granted to them with 'WITH GRANT' option in database security reports.

Table 9–121 MGMT$ESA_WITH_GRANT_REPORT

Column	Datatype	Description
TARGET_GUID	RAW(16)	The GUID of the target for which the report has the data.

Table 9–121 (Cont.) MGMT$ESA_WITH_GRANT_REPORT

Column	Datatype	Description
TARGET_NAME	VARCHAR2(256)	The name of the target for which the report has the data.
PRINCIPAL	VARCHAR2(512)	The user or role which has been granted a privilege 'WITH GRANT' option.
OBJECT_NAME	VARCHAR2(512)	The privilege which has been granted 'WITH GRANT' option.

9.14 Configuration Management Views

9.14.1 MGMT$CSA_COLLECTIONS

MGMT$CSA_COLLECTIONS displays top-level information about all client configurations.

Table 9–122 MGMT$CSA_COLLECTIONS

Column	Datatype	Description
DISPLAY_TARGET_NAME	VARCHAR2(256)	The display name of the client
CSACLIENT	VARCHAR2(4000)	The display name plus the custom keys, if they exist
COLLECTION_TIMESTAMP	DATE	The time at which the data was collected from the client
NET_IP	VARCHAR2(20)	The actual IP address of the client.
NET_EFFECTIVE_IP	VARCHAR2(20)	The client IP address seen by the server
COLLECTION_MESSAGE	VARCHAR2(4000)	Error message generated while applet was running
OS_USER_NAME	VARCHAR2(500)	The client's OS username.
HOSTNAME	VARCHAR2(128)	The client's hostname.
DOMAIN	VARCHAR2(500)	The client's domain.
BOOT_DISK_VOLUME_ SERIAL_NUM	VARCHAR2(629)	The client's boot disk volume serial number
COMPLIANCE	NUMBER	The overall compliance score for the client (15=passed, 18=info, 20=warning, 25=critical)
APPID	VARCHAR2(128)	The collection tag for this client configuration
NET_SUBNET	VARCHAR2(20)	The client's subnet mask
NET_LATENCY_IN_MS	NUMBER	The client's HTTP response time with the server
NET_BANDWIDTH_IN_KBITPS	NUMBER	The client's download bandwidth from the server
BROWSER_TYPE	VARCHAR2(100)	The name of the browser used to run CSA
BROWSER_VERSION	VARCHAR2(20)	The version of the browser used to run CSA
BROWSER	VARCHAR2(121)	A summary column that combines the browser name and version
BROWSER_JVM_VENDOR	VARCHAR2(20)	The version of the JVM used to run the applet
BROWSER_JVM_VERSION	VARCHAR2(20)	The version of the JVM used to run the applet
BROWSER_PROXY_SERVER	VARCHAR2(4000)	The proxy server used by the browser

Table 9–122 (Cont.) MGMT$CSA_COLLECTIONS

Column	Datatype	Description
BROWSER_PROXY_ EXCEPTIONS	VARCHAR2(4000)	The client's browser proxy exceptions
BROWSER_CACHE_SIZE_IN_ MB	NUMBER	The client browser's disk cache size
BROWSER_CACHE_UPATE_ FRQ	VARCAHR2(200)	The browser's cache update policy
BROWSER_HTTP1_1_SUPPORT	VARCHAR2(1)	Whether or not the browser supports HTTP 1.1
REFERRING_URL_HEADER	VARCHAR2(4000)	The URL from which the user came to CSA, minus the query string
REFERRING_URL_PARAMS	VARCHAR2(4000)	The query string of the URL from which the user came to CSA
REFURL	VARCHAR2(4000)	The complete URL from which the user came to CSA
CSA_URL_HEADER	VARCHAR2(4000)	The URL from which the user ran CSA, minus the query string
CSA_URL_PARAMS	VARCHAR2(4000)	The query string of the URL from which the user ran CSA
CSAURL	VARCHAR2(4000)	The complete URL from which the user ran CSA
DESTINAION_URL_HEADER	VARCHAR2(4000)	The destination URL, minus the query string
DESTINATION_URL_PARAMS	VARCHAR2(4000)	The query string of the destination URL
DESTURL	VARCHAR2(4000)	The complete destination URL
CONNECTION_TYPE	NUMBER	The estimated connection type, based on the download bandwidth (1=LAN, 2=cable, 3=dialup)
IS_WINDOWS_ADMIN	VARCHAR2(1)	Whether or not the os user is a windows admin
WINDOWS_DOMAIN	VARCHAR2(100)	The windows domain of the host
BROWSER_PROXY_ENABLED	VARCHAR2(1)	Whether or not the proxy server is enabled in the browser
AUTO_CONFIG_URL	VARCHAR2(4000)	The URL of the proxy auto-configuration script used by the browser
NUMBER_OF_COOKIES	NUMBER	The number of cookies collected by CSA
NUMBER_OF_CUSTOM_ VALUES	NUMBER	The number of custom properties collected by CSA
HARDWARE	VARCHAR2(4000)	A summary of the system configuration, machine architecture, memory, disk space, and CPU
HARDWARE_VENDOR_NAME	VARCHAR2(128)	The name of the hardware vendor, e.g. ìDellî
SYSTEM_CONFIG	VARCHAR2(4000)	The client's system configuration
MACHINE_ARCHITECTURE	VARCHAR2(500)	The client's machine architecture
BUS_FREQ_IN_MHZ	NUMBER	The frequency of the motherboard's FSB
MEMORY_SIZE_IN_MB	NUMBER	The total amount of physical memory
AVAIL_MEMORY_SIZE_IN_MB	NUMBER	The amount of available physical memory when CSA was run
LOCAL_DISK_SPACE_IN_GB	NUMBER	The total amount of disk space
AVAIL_LOCAL_DISK_SPACE_ IN_GB	NUMBER	The available disk space
CPU_COUNT	NUMBER(8)	The number of CPUs

Table 9–122 (Cont.) MGMT$CSA_COLLECTIONS

Column	Datatype	Description
SYSTEM_SERIAL_NUMBER	VARCAHR2(100)	The host's serial number
MIN_CPU_SPEED_IN_MHZ	NUMBER	The minimum possible CPU speed
MAX_CPU_SPEED_IN_MHZ	NUMBER	The maximum possible CPU speed
CPU	VARCHAR2(673)	The CPU vendor, implementation, and frequency
CPU_BOARD_COUNT	NUMBER(8)	The number of CPU boards
IOCARD_COUNT	NUMBER(8)	The number of IO cards
NIC_COUNT	NUMBER	The number of NICs
FAN_COUNT	NUMBER(8)	The number of fans
POWER_SUPPLY_COUNT	NUMBER(8)	The number of power supplies
SYSTEM_BIOS	VARCHAR2(100)	The system BIOS
OPERATINGSYSTEM	VARCHAR2(453)	A summary of the OS name, version, update level, address length, and distributor version
OS_NAME	VARCHAR2(128)	The OS name
OS_VENDOR_NAME	VARCHAR2(128)	The OS vendor name
OS_BASE_VERSION	VARCAR2(100)	The OS base version
OS_UPDATE_LEVEL	VARCHAR2(100)	The OS update level
OS_DISTRIBUTOR_VERSION	VARCHAR2(100)	The OS distributor version
MAX_SWAP_SPACE_IN_MB	NUMBER	The maximum amount of swap space
OS_ADDRESS_LENGTH_IN_BITS	VARCHAR2(20)	The OS address length in bits
MAX_PROCESS_VIRTUAL_MEMORY	NUMBER	The maximum amount of virtual memory that can be allocated to a process
TIMEZONE	VARCHAR2(64)	The time zone as reported in the registry
TIMEZONE_REGION	VARCHAR2(64)	The time zone region as reported by the JVM
TIMEZONE_DELTA	NUMBER	The offset in minutes from GMT
NUMBER_OF_OS_PROPERTIES	NUMBER	The number of OS properties found
NUMBER_OF_OS_PATCHES	NUMBER	The number of OS patches found
NUMBER_OF_OS_FILESYSTEMS	NUMBER	The number of filesystems found
NUMBER_OF_OS_REGISTERED_SW	NUMBER	The number of OS-registered software products found
SNAPSHOT_ID	RAW(16)	The GUID of this configuration
TARGET_ID	RAW(16)	The GUID of the collector target
INTERNAL_TARGET_NAME	VARCHAR2(264)	The internal name of the client configuration
INTERNAL_TARGET_TYPE	VARCHAR2(64)	oracle_csa_client
COLLECTION_DURATION	NUMBER(16)	The amount of time it took to run CSA
LOADED_TIMESTAMP	DATE	The time at which the data was loaded into the repository
APPLET_VERSION	VARCHAR2(20)	The version of the applet
TARGET_ID_METHOD	VARCHAR2(100)	not used
CUSTOM_CLASS	VARCHAR2(1000)	The name of the custom class (if any)

Table 9–122 (Cont.) MGMT$CSA_COLLECTIONS

Column	Datatype	Description
CUSTOM_CLASS_VERSION	VARCHAR2(1000)	not used
KEY1	VARCHAR2(4000)	Custom key 1(optional)
KEY2	VARCHAR2(4000)	Custom key 2 (optional)
KEY3	VARCHAR2(4000)	Custom key 3 (optional)
PROXY_TARGET_NAME	VARCHAR2(256)	The name of the collector target
PROXY_TARGET_DISPLAY_NAME	VARCHAR2(256)	The display name of the collector target
PROXY_TARGET_ID	RAW(16)	The GUID of the collector target
RULES_COUNT	NUMBER	The total number of rules evaluated (including rules with status of NA)
RULES_NA_COUNT	NUMBER	The number of rules that were not applicable
RULES_PASSED_COUNT	NUMBER	The number of rules that passed
RULES_INFO_COUNT	NUMBER	The number of rules that failed with status info
RULES_WARNING_COUNT	NUMBER	The number of rules that failed with status warning
RULES_CRITICAL_COUNT	NUMBER	The number of rules that failed with status critical

9.14.2 MGMT$CSA_FAILED

MGMT$CSA_FAILED displays all failed collections.

Table 9–123 MGMT$CSA_FAILED

Column	Datatype	Description
ID	RAW(16)	The GUID of this failed collection
TIMESTAMP	DATE	The time at which this failed collection occurred
TIMEZONE_DELTA	NUMBER	The offset in minutes from GMT
SAVED_TIMESTAMP	DATE	The time at which the data was loaded in the repository
EFFECTIVE_IP	VARCHAR2(20)	The effective IP address of the client
APPID	VARCHAR2(128)	The collection tag
REFERRING_URL_HEADER	VARCHAR2(4000)	The URL from which the user was referred to CSA, minus the query string
REFERRING_URL_PARAMS	VARCHAR2(4000)	The query string of the URL from which the user was referred to CSA
CSA_URL_HEADER	VARCHAR2(4000)	The URL from which the user ran CSA, minus the query string
CSA_URL_PARAMS	VARCHAR2(4000)	The query string of the URL from which the user tried to run CSA
DESTINATION_URL_HEADER	VARCHAR2(4000)	The destination URL minus the query string
DESTINATION_URL_PARAMS	VARCHAR2(4000)	The query string of the destination URL
BROWSER_TYPE	VARCHAR2(100)	The type of browser used to run CSA
BROWSER_VERSION	VARCHAR2(20)	The version of the browser used to run CSA
BROWSER_JVM_VENDOR	VARCHAR(100)	The vendor of the JVM used to run CSA
BROWSER_JVM_VERSION	VARCHAR2(20)	The version of the JVM used to run CSA

Table 9–123 (Cont.) MGMT$CSA_FAILED

Column	Datatype	Description
OS_ARCH	VARCHAR2(100)	The OS architecture of the client as reported in the ìos.archî Java system property
OS_NAME	VARCHAR2(100)	The OS name of the client as reported in the ìos.nameî Java system property
HTTP_REQUEST_USER_AGENT	VARCHAR2(100)	The HTTP user-Agent header sent by the client
ERROR_CODE	VARCHAR2(1)	The error condition that caused the failed collection (0=OS not supported, 1=browser not supported, 2=applet certificate refused by user, 3=other error)
ERROR_TEXT	VARCHAR2(1024)	Text that is collected along with the error code, such as a stack trace

9.14.3 MGMT$CSA_HOST_OS_COMPONENTS

MGMT$CSA_HOST_OS_COMPONENTS displays all OS components find on CSA client machines.

Table 9–124 MGMT$CSA_HOST_OS_COMPONENTS

Column	Datatype	Description
DISPLAY_TARGET_NAME	VARCHAR2(256)	The display name of the client
TYPE	VARCHAR2(100)	The type of the component
NAME	VARCHAR2(128)	The name of the component
VERSION	VARCHAR2(100)	The version of the component
DESCRIPTION	VARCHAR2(2000)	The description of the component
ISNTALLATION_DATE	DATE	The date the component was installed
SNAPSHOT_ID	RAW(16)	The snapshot ID of the client configuration
TARGET_ID	RAW(16)	The target GUID of the collector target
COLLECTION_TIMESTAMP	DATE	The time at which the client configuration was collected

9.14.4 MGMT$CSA_HOST_SW

MGMT$CSA_HOST_SW displays all OS-registered software found on CSA hosts.

Table 9–125 MGMT$CSA_HOST_SW

Column	Datatype	Description
DISPLAY_TARGET_NAME	VARCHAR2(256)	The display name of the client
NAME	VARCHAR2(128)	The name of the software
VENDOR_NAME	VARCHAR2(128)	The name of the software vendor
VERSION	VARCHAR2(100)	The version of the software
INSTALLATION_DATE	DATE	The date on which the software as installed
INSTALLED_LOCATION	VARCHAR2(1024)	The location in which the software is installed
DESCRIPTION	VARCHAR2(2000)	The description of the software
VENDOR_SW_SPECIFIC_INFO	VARCHAR2(4000)	Any additional information provided by the vendor
SNAPSHOT_ID	RAW(16)	The snapshot ID of the client configuration

Table 9–125 (Cont.) MGMT$CSA_HOST_SW

Column	Datatype	Description
TARGET_ID	RAW(16)	The ID of the collector target
COLLECTION_TIMESTAMP	DATE	The time at which the data was collected

9.14.5 MGMT$CSA_HOST_COOKIES

MGMT$CSA_HOST_COOKIES displays the cookies collected with client configurations.

Table 9–126 MGMT$CSA_HOST_COOKIES

Column	Datatype	Description
DISPLAY_TARGET_NAME	VARCHAR2(256)	The display name of the client
NAME	VARCHAR2(4000)	The name of the cookie
VALUE	VARCHAR2(4000)	The payload of the cookie
SNAPSHOT_ID	RAW(16)	The snapshot ID of the client configuration
TARGET_ID	RAW(16)	The ID of the collector target
COLLECTION_TIMESTAMP	DATE	The time at which the data was collected

9.14.6 MGMT$CSA_HOST_CUSTOM

MGMT$CSA_HOST_CUSTOM displays the custom properties collected with client configurations.

Table 9–127 MGMT$CSA_HOST_CUSTOM

Column	Datatype	Description
DISPLAY_TARGET_NAME	VARCHAR2(256)	The display name of the client
TYPE	VARCHAR2(512)	The category of the custom property
NAME	VARCHAR2(512)	The name of the custom property
TYPE_UI	VARCHAR2(4000)	The display category of the custom property
NAME_UI	VARCHAR2(4000)	The display name of the custom property
VALUE	VARCHAR2(4000)	The value of the custom property
DISPLAY_UI	VARCHAR2(1)	Should this property be displayed in the UI? Y or N
HISTORY_TRACKING	VARCHAR2(1)	Not used
SNAPSHOT_ID	RAW(16)	The snapshot ID of the client configuration
TARGET_ID	RAW(16)	The ID of the collector target
COLLECTION_TIMESTAMP	DATE	The time at which the data was collected

9.14.7 MGMT$CSA_HOST_RULES

MGMT$CSA_HOST_RULES displays the rules that were evaluated with each client configuration.

Table 9–128 MGMT$CSA_HOST_RULES

Column	Datatype	Description
SNAPSHOT_ID	RAW(16)	The snapshot ID of the client configuration

Table 9–128 (Cont.) MGMT$CSA_HOST_RULES

Column	Datatype	Description
NAME	VARCHAR2(128)	The name of the rule
DESCRIPTION	VARCHAR2(256)	The description of the rule
STATUS	NUMBER	The status of the rule (-2=NA, 15=passed, 18=info, 20=warning, 25=critical)
MOREINFO	VARCHAR2(1024)	Any additional information for the rule

9.14.8 MGMT$CSA_HOST_CPUS

MGMT$CSA_HOST_CPUS displays information about the CPUs of CSA hosts. CSA assumes that in a multi-CPU host, all CPUs are identical.

Table 9–129 MGMT$CSA_HOST_CPUS

Column	Datatype	Description
DISPLAY_TARGET_NAME	VARCHAR2(256)	The display name of the client
VENDOR_NAME	VARCHAR2(128)	The name of the CPU vendor
FREQ_IN_MHZ	NUMBER	The clock frequency of the CPU
ECACHE_IN_MB	NUMBER	The size of the extended cache
IMPL	VARCAHR2(500)	The CPU implementation
REVISION	VARCHAR2(2000)	The CPU revision
MASK	VARCHAR2(500)	The CPU mask
NUMBER_OF_CPUS	NUMBER	The number of CPUs
SNAPSHOT_ID	RAW(16)	The snapshot ID of the client configuration
TARGET_ID	RAW(16)	The ID of the collector target
COLLECTION_TIMESTAMP	DATE	The time at which the data was collected

9.14.9 MGMT$CSA_HOST_IOCARDS

MGMT$CSA_HOST_IOCARDS displays all IO cards collected from client configurations.

Table 9–130 MGMT$CSA_HOST_IOCARDS

Column	Datatype	Description
DISPLAY_TARGET_NAME	VARCHAR2(256)	The display name of the client
VENDOR_NAME	VARCHAR2(128)	The name of the IO card vendor
NAME	VARCHAR2(128)	The name of the IO card
FREQ_IN_MHZ	NUMBER	The frequency of the IO card bus
BUS	VARCHAR2(500)	The bus type (PCI or AGP)
REVISION	VARCHAR2(2000)	The IO card revision
NUMBER_OF_IOCARDS	NUMBER	The number of cards
SNAPSHOT_ID	RAW(16)	The snapshot ID of the client configuration
TARGET_ID	RAW(16)	The ID of the collector target
COLLECTION_TIMESTAMP	DATE	The time at which the data was collected

9.14.10 MGMT$CSA_HOST_NICS

MGMT$CSA_HOST_NICS displays all network interface cards collected from client configurations.

Table 9–131 MGMT$CSA_HOST_NICS

Column	Datatype	Description
DISPLAY_TARGET_NAME	VARCHAR2(256)	The display name of the client
NAME	VARCHAR2(128)	The name of the NIC
DESCRIPTION	VARCHAR2(500)	The description of the NIC
FLAGS	VARCHAR2(1024)	Flags set on the NIC – not applicable for Windows
MAX_TRANSFER_UNIT	NUMBER	The maximum transfer unit of the NIC
INET_ADDRESS	VARCHAR2(20)	The IP address of the NIC
MASK	VARCHAR2(20)	The subnet mask of the NIC
BROADCAST_ADDRESS	VARCHAR2(20)	The broadcast address of the NIC
MAC_ADDRESS	VARCHAR2(20)	The MAC address of the NIC
HOSTNAME_ALIASES	VARCHAR2(4000)	Any aliases for the hostname that are stored in the NIC
DEFAULT_GATEWAY	VARCHAR2(20)	The default gateway for the NIC
DHCP_ENABLED	VARCHAR2(1)	Whether or not DHCP is enabled
SNAPSHOT_ID	RAW(16)	The snapshot ID of the client configuration
TARGET_ID	RAW(16)	The ID of the collector target
COLLECTION_TIMESTAMP	DATE	The time at which the data was collected

9.14.11 MGMT$CSA_HOST_OS_PROPERTIES

MGMT$CSA_HOST_OS_PROPERTIES displays all OS properties, such as environment variables, found on CSA hosts.

Table 9–132 MGMT$CSA_HOST_OS_PROPERTIES

Column	Datatype	Description
DISPLAY_TARGET_NAME	VARCHAR2(256)	The display name of the client
SOURCE	VARHCAR2(128)	The source (e.g. the system environment) of the property
NAME	VARCHAR2(128)	The name of the property
VALUE	VARCHAR2(2000)	The value of the property
SNAPSHOT_ID	RAW(16)	The snapshot ID of the client configuration
TARGET_ID	RAW(16)	The ID of the collector target
COLLECTION_TIMESTAMP	DATE	The time at which the data was collected

9.14.12 MGMT$CSA_HOST_OS_FILESYSEMS

MGMT$CSA_HOST_OS_FILESYSEMS displays all filesystems found on CSA hosts.

Table 9–133 MGMT$CSA_HOST_OS_FILESYSEMS

Column	Datatype	Description
DISPLAY_TARGET_NAME	VARCHAR2(256)	The display name of the client

Table 9–133 (Cont.) MGMT$CSA_HOST_OS_FILESYSEMS

Column	Datatype	Description
RESOURCE_NAME	VARCHAR2(128)	The name of the filesystem
MOUNT_LOCATION	VARCHAR2(1024)	The location from which it is mounted
TYPE	VARCHAR2(100)	The filesystem type
DISK_SPACE_IN_GB	NUMBER	The total disk space
AVAIL_DISK_SPACE_IN_GB	NUMBER	The available disk space
LOCAL_DRIVE	VARCHAR2(1)	The Windows drive letter on which it is mounted
MOUNT_OPTIONS	VARCHAR2(1024)	The mount options
SNAPSHOT_ID	RAW(16)	The snapshot ID of the client configuration
TARGET_ID	RAW(16)	The ID of the collector target
COLLECTION_TIMESTAMP	DATE	The time at which the data was collected

9.14.13 MGMT$ECM_CONFIG_HISTORY

MGMT$ECM_CONFIG_HISTORY displays the data needed for generic categories.

Table 9–134 MGMT$ECM_CONFIG_HISTORY

Column	Datatype	Description
DELTATIME	DATE	The time at which the diff was recorded
DELTAGUID	RAW(16)	The unique ID of the diff
TIMEZONE	VARCHAR2(64)	The time zone of the target
TARGET_NAME	VARCHAR2(256)	The name of the target
HOSTNAME	VARCHAR2(256)	The name of the target's host
TARGET_TYPE	VARCHAR2(64)	The type of the target
SNAPSHOTTYPE	VARCHAR2(64)	The type of the snapshot in which the diff was recorded
COLLECTIONTYPE	VARCHAR2(64)	The name of the table in which the diff occurred
TABLE_PATH	VARCHAR2(1000)	The full path of the table
CATEGORY	VARCHAR2(1132)	The fully qualified leaf category
OPERATION	VARCHAR2(10)	The type of change (update, delete, insert, same)
ROWGUID	RAW(16)	The entry ID GUID
ATTRIBUTE	VARCHAR2(64)	Attribute column name
NEWVALUE	VARCHAR2(4000)	New value of attribute
OLDVALUE	VARCHAR2(4000)	Old value of attribute

9.14.14 MGMT$ECM_CONFIG_HISTORY_KEY1

MGMT$ECM_CONFIG_HISTORY_KEY1 displays the data needed for specific categories.

Table 9–135 MGMT$ECM_CONFIG_HISTORY_KEY1

Column	Datatype	Description
DELTATIME	DATE	The time at which the diff was recorded
DELTAGUID	RAW(16)	The unique ID of the diff

Table 9–135 (Cont.) MGMT$ECM_CONFIG_HISTORY_KEY1

Column	Datatype	Description
TIMEZONE	VARCHAR2(64)	The time zone of the target
TARGET_NAME	VARCHAR2(256)	The name of the target
HOSTNAME	VARCHAR2(256)	The name of the target's host
TARGET_TYPE	VARCHAR2(64)	The type of the target
SNAPSHOTTYPE	VARCHAR2(64)	The type of the snapshot in which the diff was recorded
COLLECTIONTYPE	VARCHAR2(64)	The name of the table in which the diff occurred
TABLE_PATH	VARCHAR2(1000)	The full path of the table
CATEGORY	VARCHAR2(1132)	The fully qualified leaf category
OPERATION	VARCHAR2(10)	The type of change (update, delete, insert, same)
KEY1	VARCHAR2(4000)	Value of Key 1
ATTRIBUTE	VARCHAR2(64)	Attribute column name
NEWVALUE	VARCHAR2(4000)	New value of attribute
OLDVALUE	VARCHAR2(4000)	Old value of attribute

9.14.15 MGMT$ECM_CONFIG_HISTORY_KEY2

MGMT$ECM_CONFIG_HISTORY_KEY2 displays the data needed for specific categories.

Table 9–136 MGMT$ECM_CONFIG_HISTORY_KEY2

Column	Datatype	Description
DELTATIME	DATE	The time at which the diff was recorded
DELTAGUID	RAW(16)	The unique ID of the diff
TIMEZONE	VARCHAR2(64)	The time zone of the target
TARGET_NAME	VARCHAR2(256)	The name of the target
HOSTNAME	VARCHAR2(256)	The name of the target's host
TARGET_TYPE	VARCHAR2(64)	The type of the target
SNAPSHOTTYPE	VARCHAR2(64)	The type of the snapshot in which the diff was recorded
COLLECTIONTYPE	VARCHAR2(64)	The name of the table in which the diff occurred
TABLE_PATH	VARCHAR2(1000)	The full path of the table
CATEGORY	VARCHAR2(1132)	The fully qualified leaf category
OPERATION	VARCHAR2(10)	The type of change (update, delete, insert, same)
KEY1	VARCHAR2(4000)	Value of Key 1
KEY2	VARCHAR2(4000)	Value of Key 2
ATTRIBUTE	VARCHAR2(64)	Attribute column name
NEWVALUE	VARCHAR2(4000)	New value of attribute
OLDVALUE	VARCHAR2(4000)	Old value of attribute

9.14.16 MGMT$ECM_CONFIG_HISTORY_KEY3

MGMT$ECM_CONFIG_HISTORY_KEY3 displays the data needed for specific categories.

Table 9–137 MGMT$ECM_CONFIG_HISTORY_KEY3

Column	Datatype	Description
DELTATIME	DATE	The time at which the diff was recorded
DELTAGUID	RAW(16)	The unique ID of the diff
TIMEZONE	VARCHAR2(64)	The time zone of the target
TARGET_NAME	VARCHAR2(256)	The name of the target
HOSTNAME	VARCHAR2(256)	The name of the target's host
TARGET_TYPE	VARCHAR2(64)	The type of the target
SNAPSHOTTYPE	VARCHAR2(64)	The type of the snapshot in which the diff was recorded
COLLECTIONTYPE	VARCHAR2(64)	The name of the table in which the diff occurred
TABLE_PATH	VARCHAR2(1000)	The full path of the table
CATEGORY	VARCHAR2(1132)	The fully qualified leaf category
OPERATION	VARCHAR2(10)	The type of change (update, delete, insert, same)
KEY1	VARCHAR2(4000)	Value of Key 1
KEY2	VARCHAR2(4000)	Value of Key 2
KEY3	VARCHAR2(4000)	Value of Key 3
ATTRIBUTE	VARCHAR2(64)	Attribute column name
NEWVALUE	VARCHAR2(4000)	New value of attribute
OLDVALUE	VARCHAR2(4000)	Old value of attribute

9.14.17 MGMT$ECM_CONFIG_HISTORY_KEY4

MGMT$ECM_CONFIG_HISTORY_KEY4 displays the data needed for specific categories.

Table 9–138 MGMT$ECM_CONFIG_HISTORY_KEY4

Column	Datatype	Description
DELTATIME	DATE	The time at which the diff was recorded
DELTAGUID	RAW(16)	The unique ID of the diff
TIMEZONE	VARCHAR2(64)	The time zone of the target
TARGET_NAME	VARCHAR2(256)	The name of the target
HOSTNAME	VARCHAR2(256)	The name of the target's host
TARGET_TYPE	VARCHAR2(64)	The type of the target
SNAPSHOTTYPE	VARCHAR2(64)	The type of the snapshot in which the diff was recorded
COLLECTIONTYPE	VARCHAR2(64)	The name of the table in which the diff occurred
TABLE_PATH	VARCHAR2(1000)	The full path of the table
CATEGORY	VARCHAR2(1132)	The fully qualified leaf category
OPERATION	VARCHAR2(10)	The type of change (update, delete, insert, same)
KEY1	VARCHAR2(4000)	Value of Key 1

Table 9–138 (Cont.) MGMT$ECM_CONFIG_HISTORY_KEY4

Column	Datatype	Description
KEY2	VARCHAR2(4000)	Value of Key 2
KEY3	VARCHAR2(4000)	Value of Key 3
KEY4	VARCHAR2(4000)	Value of Key 4
ATTRIBUTE	VARCHAR2(64)	Attribute column name
NEWVALUE	VARCHAR2(4000)	New value of attribute
OLDVALUE	VARCHAR2(4000)	Old value of attribute

9.14.18 MGMT$ECM_CONFIG_HISTORY_KEY5

MGMT$ECM_CONFIG_HISTORY_KEY5 displays the data needed for specific categories.

Table 9–139 MGMT$ECM_CONFIG_HISTORY_KEY5

Column	Datatype	Description
DELTATIME	DATE	The time at which the diff was recorded
DELTAGUID	RAW(16)	The unique ID of the diff
TIMEZONE	VARCHAR2(64)	The time zone of the target
TARGET_NAME	VARCHAR2(256)	The name of the target
HOSTNAME	VARCHAR2(256)	The name of the target's host
TARGET_TYPE	VARCHAR2(64)	The type of the target
SNAPSHOTTYPE	VARCHAR2(64)	The type of the snapshot in which the diff was recorded
COLLECTIONTYPE	VARCHAR2(64)	The name of the table in which the diff occurred
TABLE_PATH	VARCHAR2(1000)	The full path of the table
CATEGORY	VARCHAR2(1132)	The fully qualified leaf category
OPERATION	VARCHAR2(10)	The type of change (update, delete, insert, same)
KEY1	VARCHAR2(4000)	Value of Key 1
KEY2	VARCHAR2(4000)	Value of Key 2
KEY3	VARCHAR2(4000)	Value of Key 3
KEY4	VARCHAR2(4000)	Value of Key 4
KEY5	VARCHAR2(4000)	Value of Key 5
ATTRIBUTE	VARCHAR2(64)	Attribute column name
NEWVALUE	VARCHAR2(4000)	New value of attribute
OLDVALUE	VARCHAR2(4000)	Old value of attribute

9.14.19 MGMT$ECM_CONFIG_HISTORY_KEY6

MGMT$ECM_CONFIG_HISTORY_KEY6 displays the data needed for specific categories.

Table 9–140 MGMT$ECM_CONFIG_HISTORY_KEY6

Column	Datatype	Description
DELTATIME	DATE	The time at which the diff was recorded

Table 9–140 (Cont.) MGMT$ECM_CONFIG_HISTORY_KEY6

Column	Datatype	Description
DELTAGUID	RAW(16)	The unique ID of the diff
TIMEZONE	VARCHAR2(64)	The time zone of the target
TARGET_NAME	VARCHAR2(256)	The name of the target
HOSTNAME	VARCHAR2(256)	The name of the target's host
TARGET_TYPE	VARCHAR2(64)	The type of the target
SNAPSHOTTYPE	VARCHAR2(64)	The type of the snapshot in which the diff was recorded
COLLECTIONTYPE	VARCHAR2(64)	The name of the table in which the diff occurred
TABLE_PATH	VARCHAR2(1000)	The full path of the table
CATEGORY	VARCHAR2(1132)	The fully qualified leaf category
OPERATION	VARCHAR2(10)	The type of change (update, delete, insert, same)
KEY1	VARCHAR2(4000)	Value of Key 1
KEY2	VARCHAR2(4000)	Value of Key 2
KEY3	VARCHAR2(4000)	Value of Key 3
KEY4	VARCHAR2(4000)	Value of Key 4
KEY5	VARCHAR2(4000)	Value of Key 5
KEY6	VARCHAR2(4000)	Value of Key 6
ATTRIBUTE	VARCHAR2(64)	Attribute column name
NEWVALUE	VARCHAR2(4000)	New value of attribute
OLDVALUE	VARCHAR2(4000)	Old value of attribute

9.14.20 MGMT$HW_NIC

MGMT$HW_NIC displays performance information for host hardware network cards.

Table 9–141 MGMT$HW_NIC

Column	Datatype	Description
HOST_NAME	VARCHAR2(256)	The host name
NAME	VARCHAR2(128)	The name of the card
INET_ADDRESS	VARCHAR2(20)	The IP address of the card
MAX_TRANSFER_UNIT	NUMBER	The maximum transfer unit
BROADCAST_ADDRESS	VARCHAR2(20	The broadcast address of the card
MASK	VARCHAR2(20)	The card's subnet mask
FLAGS	VARCHAR2(1024)	The flags on the card
MAC_ADDRESS	VARCHAR2(20)	The MAC address of the card
HOST_ALIASES	VARCHAR2(4000)	The aliases for the hostname
SNAPSHOT_GUID	RAW(16)	The GUID of the config. snapshot

9.14.21 MGMT$OS_COMPONENTS

MGMT$OS_COMPONENTS displays performance information for host OS components.

Table 9–142 MGMT$OS_COMPONENTS

Column	Datatype	Description
HOST	VARCHAR2(256)	The name of the host
NAME	VARCHAR2(128)	The name of the component
TYPE	VARCHAR2(100)	The type of the component
VERSION	VARCHAR2(100)	The version of the component
DESCRIPTION	VARCHAR2(2000)	The description of the component
INSTALLATION_DATE	DATE	The installation date of the component
SNAPSHOT_GUID	RAW(16)	The GUID of the config. snapshot

9.14.22 MGMT$OS_FS_MOUNT

MGMT$OS_FS_MOUNT displays performance information for mounted filesystems.

Table 9–143 MGMT$OS_FS_MOUNT

Column	Datatype	Description
HOST_NAME	VARCHAR2(256)	The host name
RESOURCE_NAME	VARCHAR2(128)	The name of the mounted resource
TYPE	VARCHAR2(100)	The filesystem type
MOUNT_LOCATION	VARCHAR2(1024)	The mount location
MOUNT_OPTIONS	VARCHAR2(1024)	The mount options
SNAPSHOT_GUID	RAW(16)	The GUID of the config. snapshot

9.14.23 MGMT$OS_HW_SUMMARY

MGMT$OS_HW_SUMMARY displays summary information for both operating systems and hardware.

Table 9–144 MGMT$OS_HW_SUMMARY

Column	Datatype	Description
HOST_NAME	VARCHAR2(256)	The host name
DOMAIN	VARCHAR2(500)	The host's domain
OS_SUMMARY	VARCHAR2(352)	A summary of OS information
SYSTEM_CONFIG	VARCHAR2(4000)	The system configuration
MA	VARCHAR2(500)	The machine architecture
FREQ	NUMBER	The CPU frequency
MEM	NUMBER	The total amount of memory
DISK	NUMBER	The total amount of disk space
CPU_COUNT	NUMBER	The number of CPUs
VENDOR_NAME	VARCHAR2(128)	The name of the system vendor
OS_VENDOR	VARCHAR2(128)	The name of the OS vendor
DISTRIBUTOR_VERSION	VARCHAR2(100)	The OS distributor version
SNAPSHOT_GUID	RAW(16)	The GUID of the config. snapshot

9.14.24 MGMT$OS_KERNEL_PARAMS

MGMT$OS_KERNEL_PARAMS displays performance information for OS kernel parameters.

Table 9–145 MGMT$OS_KERNEL_PARAMS

Column	Datatype	Description
HOST	VARCHAR2(256)	The name of the host
SOURCE	VARCHAR2(128)	The source of the parameter
NAME	VARCHAR2(128)	The name of the parameter
VALUE	VARCHAR2(2000)	The value of the parameter
DATATYPE	VARCHAR2(4000)	The data type of the parameter
SNAPSHOT_GUID	RAW(16)	The GUID of the config. snapshot

9.14.25 MGMT$OS_PATCHES

MGMT$OS_PATCHES displays performance information for OS patches.

Table 9–146 MGMT$OS_PATCHES

Column	Datatype	Description
HOST	VARCHAR2(256)	The name of the host
OS_EXTENDED	VARCHAR2(352)	A summary column that includes update level and address length
OS	VARCHAR2(229)	A summary column that includes name and version
PATCH	VARCHAR2(128)	The name of the patch

9.14.26 MGMT$OS_SUMMARY

MGMT$OS_SUMMARY displays operating system summary information.

Table 9–147 MGMT$OS_SUMMARY

Column	Datatype	Description
HOST	VARCHAR2(256)	The name of the host
NAME	VARCHAR2(128)	The name of the OS
VENDOR_NAME	VARCHAR2(128)	The name of the OS vendor
BASE_VERSION	VARCHAR2(100)	The base version of the OS
UPDATE_LEVEL	VARCHAR2(100)	The update level of the OS
DISTRIBUTOR_VERSION	VARCHAR2(100)	The distributor version of the OS
MAX_SWAP_SPACE_IN_MB	NUMBER	The maximum amount of swap space
SNAPSHOT_GUID	RAW(16)	The GUID of the config. snapshot

9.14.27 MGMT$SOFTWARE_COMP_PATCHSET

MGMT$SOFTWARE_COMP_PATCHSET displays information on components and patchsets.

Table 9–148 MGMT$SOFTWARE_COMP_PATCHSET

Column	Datatype	Description
HOST_NAME	VARCHAR2(256)	The name of the host
HOME_NAME	VARCHAR2(64)	The name of the Oracle Home
HOME_LOCATION	VARCHAR2(128)	The path of the Oracle Home
COMPONENT_NAME	VARCHAR2(128)	The name of the component
COMPONENT_BASE_VERSION	VARCHAR2(64)	The base version of the component
COMPONENT_VERSION	VARCHAR2(64)	The current version of the component
PATCHSET_NAME	VARCHAR2(128)	The name of the patchset
PATCHSET_VERSION	VARCHAR2(64)	The version of the patchset
SNAPSHOT_GUID	RAW(16)	The GUID of the config. snapshot

9.14.28 MGMT$SOFTWARE_COMPONENT_ONEOFF

MGMT$SOFTWARE_COMPONENT_ONEOFF displays information on oneoff patches.

Table 9–149 MGMT$SOFTWARE_COMPONENT_ONEOFF

Column	Datatype	Description
HOST_NAME	VARCHAR2(256)	The name of the host
HOME_NAME	VARCHAR2(64)	The name of the Oracle Home
HOME_LOCATION	VARCHAR2(128)	The path of the Oracle Home
COMPONENT_NAME	VARCHAR2(128)	The name of the component
COMPONENT_EXTERNAL_ NAME	VARCHAR2(128)	The external name of the component
COMPONENT_BASE_VERSION	VARCHAR2(64)	The base version of the component
COMPONENT_VERSION	VARCHAR2(64)	The current version of the component
PATCH_ID	VARCHAR2(128)	The ID of the patch
SNAPSHOT_GUID	RAW(16)	The GUID of the config. snapshot

9.14.29 MGMT$SOFTWARE_COMPONENTS

MGMT$SOFTWARE_COMPONENTS displays information on components.

Table 9–150 MGMT$SOFTWARE_COMPONENTS

Column	Datatype	Description
NAME	VARCHAR2(128)	The name of the component
EXTERNAL_NAME	VARCHAR2(128)	The external name of the component
BASE_VERSION	VARCHAR2(64)	The base version of the component
PATCHSETS_IN_HOME	VARCHAR2(4000)	List of patchsets in home
VERSION	VARCHAR2(64)	The version of the component
HOST_NAME	VARCHAR2(256)	The name of the host
HOME_LOCATION	VARCHAR2(128)	The path of the Oracle Home
HOME_NAME	VARCHAR2(64)	The name of the Oracle Home
DESCRIPTION	VARCHAR2(1024)	A description of the component

Table 9–150 (Cont.) MGMT$SOFTWARE_COMPONENTS

Column	Datatype	Description
INSTALLER_VERSION	VARCHAR2(64)	The version of the installer
MIN_DEINSTALLER_VERSION	VARCHAR2(64)	The minimum deinstaller version
INSTALL_TIMESTAMP	DATE	The installation date
IS_TOP_LEVEL	VARCHAR2(1)	Whether or not the component is a top-level component
INTERIM_PATCHES_IN_HOME	VARCHAR2(4000)	The interim patches in the home
BUGS_FIXED_BY_INTERM_ INPATCEHS	VARCHAR2(4000)	A list of bugs fixed by the interim patches in the home
SNAPSHOT_GUID	RAW(16)	The GUID of the config. snapshot

9.14.30 MGMT$SOFTWARE_DEPENDENCIES

MGMT$SOFTWARE_COMPONENTS displays information on components.

Table 9–151 MGMT$SOFTWARE_DEPENDENCIES

Column	Datatype	Description
HOST_NAME	VARHCAR2(256)	The name of the host
REFERENCER_HOME_NAME	VARCHAR2(64)	The Oracle Home of the referencing component
REFERENCER_HOME_ LOCATION	VARCHAR2(128)	The Oracle Home path of the referencing component
REFERENCER_NAME	VARCHAR2(128)	The name of the referencing component
REFERNCER_BASE_VERSION	VARCHAR2(64)	The base version of the referencing component
REFERENCED_HOME_NAME	VARCHAR2(64)	The Oracle Home of the referenced component
REFERENCED_HOME_ LOCATION	VARCHAR2(128)	The Oracle Home path of the referenced component
REFERENCED_NAME	VARCHAR2(128)	The name of the referenced component
REFERENCED_BASE_VERSION	VARCHAR2(64)	The base version of the referenced component
SNAPSHOT_GUID	RAW(16)	The GUID of the configuration snapshot

9.14.31 MGMT$SOFTWARE_HOMES

MGMT$SOFTWARE_HOMES displays information about Oracle Homes.

Table 9–152 MGMT$SOFTWARE_HOMES

Column	Datatype	Description
HOST_NAME	VARHCAR2(256)	The name of the host
HOME_NAME	VARCHAR2(64)	The name of the Oracle Home
HOME_TYPE	VARCHAR2(11)	The type of the Oracle Home
HOME_LOCATION	VARCHAR2(128)	The path of the Oracle Home
SNAPSHOT_GUID	RAW(16)	The GUID of the config. snapshot

9.14.32 MGMT$SOFTWARE_ONEOFF_PATCHES

MGMT$SOFTWARE_ONEOFF_PATCHES displays information on oneoff patches applied in Oracle Homes.

Table 9–153 MGMT$SOFTWARE_ONEOFF_PATCHES

Column	Datatype	Description
HOST_NAME	VARHCAR2(256)	The name of the host
HOME_NAME	VARCHAR2(64)	The name of the Oracle Home
HOME_LOCATION	VARCHAR2(128)	The path of the Oracle Home
PATCH_ID	VARCHAR2(128)	The ID of the patch
INSTALL_TIMESTAMP	DATE	The installation date
DESCRIPTION	VARCHAR2(1024)	A description of the patch
IS_ROLLBACKABLE	VARCHAR2(1)	Whether or not the patch can be rolled back
SNAPSHOT_GUID	RAW(16)	The GUID of the configuration snapshot

9.14.33 MGMT$SOFTWARE_OTHERS

MGMT$SOFTWARE_OTHERS displays information on other software installed on hosts.

Table 9–154 MGMT$SOFTWARE_OTHERS

Column	Datatype	Description
HOST_NAME	VARHCAR2(256)	The name of the host
SOFTWARE_NAME	VARCHAR2(128)	The name of the software
SOFTWARE_VENDOR	VARCHAR2(128)	The software vendor
SOFTWARE_VERSION	VARCHAR2(128)	The software version
INSTALLATION_DATE	DATE	The software installation date
INSTALLATION_LOCATION	VARCHAR2(1024)	The software installation location
SNAPSHOT_GUID	RAW(16)	The GUID of the configurtion snapshot

9.14.34 MGMT$SOFTWARE_PATCHES_IN_HOMES

MGMT$SOFTWARE_PATCHES_IN_HOMES displays information on software patches in Oracle Homes.

Table 9–155 MGMT$SOFTWARE_PATCHES_IN_HOMES

Column	Datatype	Description
HOST_NAME	VARHCAR2(256)	The name of the host
HOME_NAME	VARCHAR2(64)	The name of the Oracle Home
HOME_LOCATION	VARCHAR2(128)	The path of the Oracle Home
PATCH_ID	VARCHAR2(128)	The ID of the patch
BUGS_FIXED	VARCHAR2(4000)	List of bugs fixed by the patch
SNAPSHOT_GUID	RAW(16)	The GUID of the config. snapshot

9.14.35 MGMT$SOFTWARE_PATCHSETS

MGMT$SOFTWARE_PATCHSETS displays information on patchsets installed in Oracle Homes.

Table 9–156 MGMT$SOFTWARE_PATCHSETS

Column	Datatype	Description
HOST_NAME	VARHCAR2(256)	The name of the host
HOME_NAME	VARCHAR2(64)	The name of the Oracle Home
HOME_LOCATION	VARCHAR2(128)	The path of the Oracle Home
NAME	VARCHAR2(128)	The name of the patchset
VERSION	VARCHAR2(64)	The version of the patchset
DESCRIPTION	VARCHAR2(1024)	A description of the patchset
INSTALLER_VERSION	VARCHAR2(64)	The version of the installer
MIN_DEINSTALLER_VERSION	VARCHAR2(64)	The minimum deinstaller version
INSTALL_TIMESTAMP	DATE	The time at which the patchset was installed
SNAPSHOT_GUID	RAW(16)	The GUID of the config. snapshot

9.15 Database Cluster Views

9.15.1 MGMT$CLUSTER_INTERCONNECTS

MGMT$CLUSTER_INTERCONNECTS displays statistics of network interfaces on the hosts in clusters.

Table 9–157 MGMT$CLUSTER_INTERCONNECTS

Column	Datatype	Description
CLUSTER_NAME	VARCHAR2(256)	Cluster target name
HOST_NAME	VARCHAR2(256)	Host target status
IF_NAME	VARCHAR2(256)	Interface name
IF_SUBNET	VARCHAR2(16)	Subnet of the interface
IF_PUBLIC	VARCHAR2(10)	Whether the interface is public (YES) or private(NO)
TOTRATE_5MIN	NUMBER	The average total transfer rate (MB/sec) on the interface in the past 5 minutes
TOTERR_5MIN	NUMBER	The percentage of error packets on the interface in the past 5 minutes
INRATE_5MIN	NUMBER	The average input rate (MB/sec) on the interface in the past 5 minutes
CURR_WARNING	NUMBER	The number of warning alerts on the interface
CURR_CRITICAL	NUMBER	The number of critial alerts on the interface.
LATEST_COLLECTION_ TIMESTAMP	DATE	The time when the data is collected.

9.15.2 MGMT$RACDB_INTERCONNECTS

MGMT$RACDB_INTERCONNECTS displays statistics of the inter-instance traffic of cluster databases.

Table 9–158 MGMT$RACDB_INTERCONNECTS

Column	Datatype	Description
CLUSTER_NAME	VARCHAR2(1024)	Cluster target name
DB_TARGET	VARCHAR2(256)	Cluster database target name
INSTANCE_TARGET	NUMBER	Cluster database instance target name.
INSTANCE_STATUS	NUMBER	Cluster database instance status
DB_NAME	VARCHAR2(1024)	Database name
SID	VARCHAR2(1024)	Instance name
IF_NAME	VARCHAR2(50)	The network interface the instance is using for inter-instance communication
HOST_NAME	VARCHAR2(256)	Host target name on which the instance is running
IF_IP	VARCHAR2(16)	The ip address of the network interface
IF_PUBLIC	VARCHAR2(10)	Whether the interface is public (YES) or private (NO)
IF_SOURCE	VARCHAR2(100)	Where does the instance pick up the interface
XFERRATE_5MIN	NUMBER	The trasfer rate of the instance to other instances
CURR_WARNING	NUMBER	The number of warning alerts of the instance regarding inter_instance communication
CURR_CRITICAL	NUMBER	The number of critical alerts of the instance regarding inter_instance communication
LATEST_COLLECTION_ TIMESTAMP	DATE	The time the data is collected

9.15.3 MGMT$HA_BACKUP

MGMT$HA_BACKUP displays details of the latest backup of the databases.

Table 9–159 MGMT$HA_BACKUP

Column	Datatype	Description
HOST	VARCHAR2(256)	Host the database is on
DATABASE_NAME	VARCHAR2(256)	Database target name
TARGET_TYPE	VARCHAR2(64)	Target type of the database. (cluster_database or oracle_database)
DISPLAY_NAME	VARCHAR2(256)	Display name of the target
TARGET_GUID	RAW(16)	The target GUID
SESSION_KEY	NUMBER	Session identifier
SESSION_RECID	NUMBER	Together, with SESSION_STAMP, used to uniquely identify job output from V$RMAN_ OUTPUT
SESSION_STAMP	NUMBER	Together, with SESSION_RECID, used to uniquely identify job output from V$RMAN_ OUTPUT
COMMAND_ID	VARCHAR2(33)	Either a user-specified SET COMMAND ID or a unique command ID generated by RMAN

Table 9–159 (Cont.) MGMT$HA_BACKUP

Column	Datatype	Description
STATUS	VARCHAR2(23)	One of the following values: ² RUNNING WITH WARNINGS ² RUNNING WITH ERRORS ² COMPLETED ² COMPLETED WITH WARNINGS ² COMPLETED WITH ERRORS ² FAILED
START_TIME	DATE	Start time of the first BACKUP command in the job
END_TIME	DATE	End time of the last BACKUP command in the job
TIME_TAKEN_DISPLAY	VARCHAR2(4000)	Time taken, shown in user-displayable format <nn>h:<nn>m:<nn>s
INPUT_TYPE	VARCHAR2(13)	Contains one of the following values. If the user command does not satisfy one of them, then preference is given in order, from top to bottom of the list. ² DB FULL ² RECVR AREA ² DB INCR ² DATAFILE FULL ² DATAFILE INCR ² ARCHIVELOG ² CONTROLFILE ² SPFILE
OUTPUT_DEVICE_TYPE	VARCHAR2(17)	Can be DISK, SBT, or *. An * indicates more than one device (in most cases, it will be DISK or SBT).
INPUT_BYTES_DISPLAY	VARCHAR2(4000)	Values in user-displayable form. They will be converted to a format of nM, nG, nT, nP, and so on.
OUTPUT_BYTES_DISPLAY	VARCHAR2(4000)	Values in user-displayable form. They will be converted to a format of nM, nG, nT, nP, and so on
OUTPUT_BYTES_PER_SEC_DISPLAY	VARCHAR2(4000)	Output write-rate-per-second. These values are in user-displayable form. They will be converted to a format of nM, nG, nT, nP, and so on.

9.16 Storage Reporting Views

9.16.1 MGMT$STORAGE_REPORT_DATA

MGMT$STORAGE_REPORT_DATA displays the Storage Data metric attributes which are common across all instrumented Storage Entities.

Table 9–160 MGMT$STORAGE_REPORT_DATA

Column	Datatype	Description
TARGET_NAME	VARCHAR2(256)	Target Name in Enterprise Manager

Table 9–160 (Cont.) MGMT$STORAGE_REPORT_DATA

Column	Datatype	Description
TARGET_TYPE	VARCHAR2(64)	Target Type in Enterprise Manager
KEY_VALUE	RAW(20)	Unique Key Value for the Storage Entity
GLOBAL_UNIQUE_ID	RAW(20)	A globally unique persistent identifier for a storage entity. All instances of a shared Storage Entity will have the same global_unique_ identifier
NAME	VARCHAR2(256)	Name of the Storage Entity
STORAGE_LAYER	VARCHAR2(32)	Storage layer of the Storage Entity. Example: - OS_DISK - VOLUME_MANAGER - LOCAL_FILESYSTEM - NFS
ENTITY_TYPE	VARCHAR2(64)	Indicates the type of Entity. Value is vendor specific. Example: Plex, Sub Disk, Diskgroup, Volume group, Metadevice etc.
RAWSIZEB	NUMBER	Total space in bytes
SIZEB	NUMBER	Size in bytes
USEDB	NUMBER	Used size in bytes
FREEB	NUMBER	Free size in bytes

9.16.2 MGMT$STORAGE_REPORT_KEYS

MGMT$STORAGE_REPORT_KEYS displays the relationship between instrumented Storage Entities.

Table 9–161 MGMT$STORAGE_REPORT_KEYS

Column	Datatype	Description
TARGET_NAME	VARCHAR2(256)	Target Name in Enterprise Manager
TARGET_TYPE	VARCHAR2(64)	Target Type in Enterprise Manager
KEY_VALUE	RAW(20)	Unique KEY_VALUE for the Storage Entity
PARENT_KEY_VALUE	RAW(20)	Key value for the parent Storage Entity.

9.16.3 MGMT$STORAGE_REPORT_PATHS

MGMT$STORAGE_REPORT_PATHS displays the OS paths for all instrumented storage Entities.

Table 9–162 MGMT$STORAGE_REPORT_PATHS

Column	Datatype	Description
TARGET_NAME	VARCHAR2(256)	Target Name in Enterprise Manager
TARGET_TYPE	VARCHAR2(64)	Target Type in Enterprise Manager
KEY_VALUE	RAW(20)	Unique Key Value for the Storage Entity
NAME	VARCHAR2(256)	Name of the Storage Entity
PATH	VARCHAR2(256)	OS path to the Storage Entity

Table 9–162 (Cont.) MGMT$STORAGE_REPORT_PATHS

Column	Datatype	Description
FILE_TYPE	VARCHAR2(256)	Type of file Examples: - _BLOCKSPECIAL - _DIRECTORY - _REGULAR
STORAGE_LAYER	VARCHAR2(32)	Storage layer of the Storage Entity. Examples: - OS_DISK - VOLUME_MANAGER - LOCAL_FILESYSTEM - NFS
ENTITY_TYPE	VARCHAR2(64)	Indicates the type of Entity. Value is vendor specific. Examples: Plex, Sub Disk, Diskgroup, Volume group, Metadevice etc.

9.16.4 MGMT$STORAGE_REPORT_ISSUES

MGMT$STORAGE_REPORT_ISSUES displays the consistency issues encountered when analyzing the instrumented storage metrics.

Table 9–163 MGMT$STORAGE_REPORT_ISSUES

Column	Datatype	Description
TARGET_NAME	VARCHAR2(256)	Target Name in Enterprise Manager
TARGET_TYPE	VARCHAR2(64)	Target Type in Enterprise Manager
TYPE	VARCHAR2(32)	Type of inconsistency. Value can be one of - MAPPING_ISSUE - MAPPING_WARNING
MESSAGE_COUNT	NUMBER	Count of the number of messages

9.16.5 MGMT$STORAGE_REPORT_DISK

MGMT$STORAGE_REPORT_DISK displays Additional Storage Data Metric Attributes for all Physical Disk Device Storage Entities.

Table 9–164 MGMT$STORAGE_REPORT_DISK

Column	Datatype	Description
TARGET_NAME	VARCHAR2(256)	Target Name in Enterprise Manager
TARGET_TYPE	VARCHAR2(64)	Target Type in Enterprise Manager
ENTITY_TYPE	VARCHAR2(64)	Indicates the type of Disk Device such as Disk or Disk Partition
USED_PATH	VARCHAR2(256)	The OS path to the disk or partition. If the disk or partition is allocated, then this is the path that is in use.

Table 9–164 (Cont.) MGMT$STORAGE_REPORT_DISK

Column	Datatype	Description
FILE_TYPE	VARCHAR2(256)	Type of file Examples: - _BLOCKSPECIAL - _REGULAR
SIZEB	NUMBER	Size in bytes
USEDB	NUMBER	Used size in bytes
FREEB	NUMBER	Free size in bytes
VENDOR	VARCHAR2(256)	Name of the disk vendor; detected through SCSI enquiry
PRODUCT	VARCHAR2(256)	Product family from the vendor; detected through SCSI enquiry

9.16.6 MGMT$STORAGE_REPORT_VOLUME

MGMT$STORAGE_REPORT_VOLUME displays Additional Storage Data Metric attributes for all Volume Manager Storage Entities.

Table 9–165 MGMT$STORAGE_REPORT_VOLUME

Column	Datatype	Description
TARGET_NAME	VARCHAR2(256)	Target Name in Enterprise Manager
TARGET_TYPE	VARCHAR2(64)	Target Type in Enterprise Manager
VENDOR	VARCHAR2(256)	Vendor name of the Volume or Software Raid Manager
PRODUCT	VARCHAR2(256)	Vendor name of the Volume or Software Raid Manager
TYPE	VARCHAR2(64)	ndicates the type of Volume entity. It can be vendor specific. In the case of Veritas Volume Manager for e.g. Volume, Plex, VM Disk, Diskgroup, Sub Disk, Metadevice, Metadevice Partition, Array, Raiddevice, Submirror, Diskset, Slice, raid-disk, spare-disk, Hot spare etc.
DISK_GROUP	VARCHAR2(256)	Disk group or Volume group name
NAME	VARCHAR2(256)	The name of the entity in the volume manager namespace
USED_PATH	VARCHAR2(256)	The OS path to the device. If the device is allocated, then this is the path that is in use.
FILE_TYPE	VARCHAR2(256)	Type of file Examples: - _BLOCKSPECIAL - _REGULAR
RAWSIZEB	NUMBER	In bytes. For a 2-way mirrored Veritas Volume. It is the sum of the size of each plex.
SIZEB	NUMBER	Size in bytes
USEDB	NUMBER	Used size in bytes
FREEB	NUMBER	Free size in bytes

Table 9–165 (Cont.) MGMT$STORAGE_REPORT_VOLUME

Column	Datatype	Description
CONFIGURATION	VARCHAR2(256)	A string describing the configuration of the Volume.

9.16.7 MGMT$STORAGE_REPORT_LOCALFS

MGMT$STORAGE_REPORT_LOCALFS displays Additional Storage Data Metric attributes for all Local Filesystem Storage Entities.

Table 9–166 MGMT$STORAGE_REPORT_LOCALFS

Column	Datatype	Description
TARGET_NAME	VARCHAR2(256)	Target Name in Enterprise Manager
TARGET_TYPE	VARCHAR2(64)	Target Type in Enterprise Manager
FILESYSTEM_TYPE	VARCHAR2(256)	The type of filesystem
FILESYSTEM	VARCHAR2(256)	The filesystem path on the operating system
MOUNTPOINT	VARCHAR2(256)	The mountpoint path on the operating system
SIZEB	NUMBER	NUMBER
USEDB	NUMBER	Used size in bytes
FREEB	NUMBER	Free size in bytes

9.16.8 MGMT$STORAGE_REPORT_NFS

MGMT$STORAGE_REPORT_NFS displays Additional Storage Data Metric attributes for all Network Filesystems.

Table 9–167 MGMT$STORAGE_REPORT_NFS

Column	Datatype	Description
TARGET_NAME	VARCHAR2(256)	Target Name in Enterprise Manager
TARGET_TYPE	VARCHAR2(64)	Target Type in Enterprise Manager
FILESYSTEM	VARCHAR2(256)	The filesystem name as seen on the operating system. For NFS filesystems the filesystem name should be in the format nfs_ server:/filesystem_name
MOUNTPOINT	VARCHAR2(256)	The mountpoint path on the operating system
SIZEB	NUMBER	Size in bytes
USEDB	NUMBER	Used size in bytes,
FREEB	NUMBER	Free size in bytes
NFS_SERVER	VARCHAR2(256)	The server name for the NFS Server.
NFS_SERVER_IP_ADDRESS	VARCHAR2(256)	The IP address of the NFS Server
NFS_VENDOR	VARCHAR2(256)	The NFS Server vendor
MOUNT_PRIVILEGE	VARCHAR2(256)	{READ | WRITE} This is the mount privilege of the filesystem.

10 Fetchlets

Enterprise Manager data retrieval is handled through predefined "fetchlets." A fetchlet is a parametrized data access mechanism that takes arguments (For example, a script, a SQL statement, a target instance's properties) as input and returns formatted data. Each fetchlet handles a specific type of data access. The fetchlets supplied with Enterprise Manager provide data retrieval capability for the most common data access methods, such as SQL, SNMP (Simple Network Management Protocol), HTTP, and DMS (Dynamic Monitoring Service). To handle more complex data access requirements, Enterprise Manager also provides an OS command fetchlet that allows developers to implement custom metric collection methods.

The following fetchlets are supplied with Enterprise Manager:

- OS Command Fetchlets
- SQL Fetchlet
- SNMP Fetchlet
- URL Timing Fetchlet
- Dynamic Monitoring Service (DMS) Fetchlet
- HTTP Data Fetchlets
- URLXML Fetchlet
- WBEM Fetchlet
- JDBC Fetchlet
- OJMX/SOAP Fetchlet

10.1 OS Command Fetchlets

The operating system (OS) command fetchlets allow you to obtain metric data by executing OS commands (either individually or from scripts) that return a standard out (stdout) data stream.

Three OS command fetchlets are available:

- OS Fetchlet (raw)
- OSLines Fetchlet (split into lines)
- OSLineToken Fetchlet (tokenized lines)

10.1.1 OS Fetchlet

The OS Fetchlet executes a given OS command and returns the command's output in a single cell table.

Input Parameters

Table 10–1 OS Fetchlet Input Parameters

Parameter	Type	Description	Use
command	string	Operating system command to be executed.	required
ENV*name*	string	OSFetchlet parameters starting with "ENV" appear in the execution environment for the command as *name* environment variables	zero or more
errStartsWith	String	When defined, this property allows you to define a custom prefix for error messages. If this property is not defined, the OSFetchlet defaults to "em_error=" as the message prefix.	Optional
script	string	Specifies the script to be executed if *command* property only provides an interpretor. For example, *command* might consist of "perl." *script* is then used to specify the particular perl script to be run. Although scripts can be specified directly from the *command* parameter, using the *script* parameter adds to stylistic clarity and readability when defining a target type metadata file.	Optional
args	string	A property that is taken as one or more arguments to the command and script properties. Although args can be specified directly from the command parameter, using the args parameter adds to stylistic clarity and readability when defining a target type metadata file.	Optional
separateErrorStream	boolean	If an error occurs while executing a *command*, this property instructs the fetchlet whether to return both the stdout and stderr to the user as an error message. When set to TRUE, only stderr output is sent to the user as an error message when there is a *command* error. When set to FALSE (default value), both the stdout and the stderr are sent to the user as an error message upon *command* failure.	Optional. (TRUE/FALSE)
em_metric_timeout	integer	Metric timeout period (in seconds). After the timeout period has finished, the Management Agent returns a timeout exception and terminates any child processes that may have been created. The Management Agent DOES NOT kill any of the grandchild processes. Specify "-1" for no timout period.	Optional

Example

You want to obtain metric data by executing the UNIX *echo* command.

Executing the command from the shell environment, you would enter:

```
echo Line 1|some more|even more\nLine 2\n\nLine 4|a little more|\n|Line 5\n|Line
6|\n|Line 7|again|\nLine 8|the|end
```

The *echo* command produces the following standard output:

```
Line 1|some more|even more
Line 2

Line 4|a little more|
|Line 5
|Line 6|
|Line 7|again|
Line 8|the|end
```

Using the OS Fetchlet with the given example command.

The fetchlet returns the following 1 x 1 table:

Figure 10–1 Table Returned by the OS Fetchlet

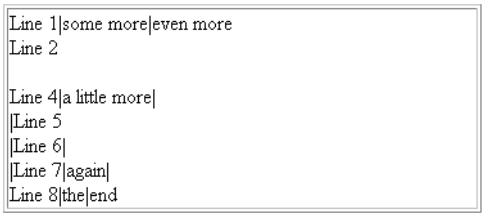


Table Returned by the OS Fetchlet

****

The raw output of the OS command is returned. Any standard error output is appended to the standard output.

Error Handling

Any problems encountered launching the *command* (For example, the *command* program no longer exists) results in an oracle.sysman.emSDK.emd.fetchlet.MetricSourceException wrapping a java.io.IOException. If the command exits with a non-zero exit value, the fetchlet throws an oracle.sysman.emSDK.emd.fetchlet.MetricSourceException wrapping an oracle.sysmand.emd.fetchlets.CommandFailedException.

Notes

Commands are NOT executed as if they are being run in a shell. This means that common shell symbols do not work, including piping, output redirection, and backgrounding.

Commands cannot read from standard input.

The fetchlet blocks and waits for the command to finish.

10.1.2 OSLines Fetchlet (split into lines)

The OS Lines Fetchlet executes a given OS command and tokenizes the OS command's output. The output is tokenized by lines. The fetchlet returns the tokens in a single

column table. The nth row in the table represents the nth line in the OS command's output.

To get the raw, untokenized output of an OS command, use the OS Fetchlet. To get the output of an OS command tokenized by lines and each line tokenized by a given delimiter, see the OS Line Token Fetchlet.

Input Parameters

Table 10–2 OSLines Fetchlet Input Parameters

Parameter	Type	Description	Use
command	string	Operating system command to be executed.	required
startsWith	string	Only lines starting with this string are included in the result.	optional; default = "" (all lines are included)
ENV*name*	string	Parameters starting with "ENV" appear in the execution environment for the command as *name* environment variables	zero or more of these
errStartsWith	string	When defined, this property allows you to define a custom prefix for error messages. If this property is not defined, the OSFetchlet defaults to "em_error=" as the message prefix.	Optional
script	string	Specifies the script to be executed if *command* property only provides an interpretor. For example, *command* might consist of "perl." *script* is then used to specify the particular perl script to be run. Although scripts can be specified directly from the *command* parameter, using the *script* parameter adds to stylistic clarity and readability when defining a target type metadata file.	Optional
args	string	A property that is taken as one or more arguments to the *command* and *script* properties. Although args can be specified directly from the *command* parameter, using the *args* parameter adds to stylistic clarity and readability when defining a target type metadata file.	Optional
separateErrorStream	boolean	If an error occurs while executing a command, this property instructs the fetchlet to whether to return both the stdout and stderr to the user as an error message. When set to TRUE, only stderr output is sent to the user as an error message when there is a command error. When set to FALSE (default value), both the stdout and the stderr are sent to the user as an error message upon command failure.	Optional. (TRUE/FALSE)

Table 10–2 (Cont.) OSLines Fetchlet Input Parameters

Parameter	Type	Description	Use
em_metric_timeout	integer	Metric timeout period (in seconds). After the timeout period has finished, the Management Agent returns a timeout exception and terminates any child processes that may have been created. The Management Agent DOES NOT kill any of the grandchild processes. Specify "-1" for no timout period.	Optional

Example

Let's take the following Unix command:

```
echo Line 1|some more|even more\nLine 2\n\nLine 4|a little more|\n|Line 5\n|Line 6|\n|Line 7|again|\nLine 8|the|end
```

It produces the following output:

```
Line 1|some more|even more
Line 2

Line 4|a little more|
|Line 5
|Line 6|
|Line 7|again|
Line 8|the|end
```

Running OSLinesFetchlet with the given example command produces the following single column table.

Figure 10–2 Table Returned by the OS LINES Fetchlet

Line 1|some more|even more
Line 2
Line 4|a little more|
|Line 5
|Line 6|
|Line 7|again|
Line 8|the|end

Table Returned by the OS LINES Fetchlet

****

Note that without content, "\n" results in a blank line inserted between Line 2 and Line 4.

Notes: Commands are NOT executed as if they are being run in a shell. This means that common shell symbols do not work, including piping, output redirection, and backgrounding.

Commands cannot read from standard input.

The fetchlet blocks and waits for the command to finish.

The standard output of the command is captured and the standard error is captured and appended to the standard output.

Lines are tokenized using "\n".

10.1.3 OSLineToken Fetchlet (tokenized lines)

The OS Line Token Fetchlet executes a given OS command and tokenizes the OS command's output. The output is tokenized first by lines, and then each line is tokenized by a given delimiter set. The fetchlet returns the tokens in a table. The nth row in the table represents the nth line in the OS command's output. The nth column in the table represents the nth token in a line as determined by the given delimiter set.

To get the raw, untokenized output of an OS command, see the OS Fetchlet.

Input Parameters

Table 10–3 OSLineToken Fetchlet Input Parameters

Parameter	Type	Description	Use
command	string	Operating system command to be executed.	required
delimiter	string	Set of characters that act as delimiters to tokenize the lines	optional; default = "" (just breaks output into lines)
startsWith	string	Only lines starting with this string are included in the result	optional; default = "" (all lines are included)
ENV*name*	string	Parameters starting with "ENV" appear in the execution environment for the command as *name* environment variables	zero or more of these
errStartsWith	String	When defined, this property allows you to define a custom prefix for error messages. If this property is not defined, the OSFetchlet defaults to "em_error=" as the message prefix.	Optional
script	string	Specifies the script to be executed if command property only provides an interpretor. For example, *command* might consist of "perl." The script is then used to specify the particular perl script to be run. Although scripts can be specified directly from the command parameter, using the script parameter adds to stylistic clarity and readability when defining a target type metadata file.	Optional

Table 10–3 (Cont.) OSLineToken Fetchlet Input Parameters

Parameter	Type	Description	Use
args	string	A property that is taken as one or more arguments to the *command* and *script* properties. Although args can be specified directly from the *command* parameter, using the args parameter adds to stylistic clarity and readability when defining a target type metadata file.	Optional
separateErrorStream	boolean	If an error occurs while executing a *command*, this property instructs the fetchlet to whether to return both the stdout and stderr to the user as an error message. When set to TRUE, only stderr output is sent to the user as an error message when there is a *command* error. When set to FALSE (default value), both the stdout and the stderr are sent to the user as an error message upon *command* failure.	Optional. (TRUE/FALSE)
em_metric_timeout	integer	Metric timeout period (in seconds). After the timeout period has finished, the Management Agent returns a timeout exception and terminates any child processes that may have been created. The Management Agent DOES NOT kill any of the grandchild processes. Specify "-1" for no timout period.	Optional

Example

Let's take the following Unix command:

```
echo Line 1|some more|even more\nLine 2\n\nLine 4|a little more|\n|Line 5\n|Line
6|\n|Line 7|again|\nLine 8|the|end
```

It produces the following output:

```
Line 1|some more|even more
Line 2

Line 4|a little more|
|Line 5
|Line 6|
|Line 7|again|
Line 8|the|end
```

Running OSLineTokenFetchlet with the given example command and a single character "|" for the delimiter generates the following table:

Figure 10–3 Table Returned by the OS Token Lines Fetchlet

Line 1	some more	even more
Line 2		
Line 4	a little more	
Line 5		
Line 6		
Line 7	again	
Line 8	the	end

Table Returned by the OS Token Lines Fetchlet

****

Error Handling

Any problem launching the command (unable to find the command program) results in a oracle.sysman.emSDK.emd.fetchlet.MetricSourceException wrapping a java.io.IOException.

If the command exits with a non-zero exit value, the fetchlet throws a oracle.sysman.emSDK.emd.fetchlet.MetricSourceException wrapping a oracle.sysmand.emd.fetchlets.CommandFailedException.

Notes

Commands are NOT executed as if they are being run in a shell. This means that common shell symbols do not work, including piping, output redirection, and backgrounding.

The fetchlet promptly closes the input stream to the running command.

The fetchlet blocks and waits for the command to finish.

Lines are tokenized using "\n".

The delimiter can be a single character or a set of characters. For example, it can be "|+_", if the line should be broken up by pipes, pluses, and underscores. If two or more delimiters are together in the output text, such as "||" or "+|+", then it is as if there are empty string tokens between them. These empty strings get columns in the result table. It is NOT considered that there are empty strings preceding a delimiter that starts a line or following a delimiter that ends a line.

In order to express non-printable characters in the delimiter set (such as tabs) in XML, use "&#xHH;" where H is the hexadecimal identifier for the character.

10.2 SQL Fetchlet

The SQL Fetchlet executes a given SQL statement on a given database as a given user and returns the table result.

Input Parameters

Table 10–4 SQL Fetchlet Input Parameters

Parameter	Type	Description	Use
Connection Information			

Table 10–4 (Cont.) SQL Fetchlet Input Parameters

Parameter	Type	Description	Use
MachineName	string	Database host	Required
Port	integer	Database port	Required
SID	string	Database SID	Required unless ServiceName is specified
ServiceName	string	Database ServiceName	Required unless SID is specified
OidRepSchemaName	string	Enterprise Manager Management Repository name. This parameter extracts database connection string and credential information (MachineName, Port, SID, ServiceName, UserName, password) from the Management Repository. Using this parameter eliminates the need to specify the connection and credential parameters in each target type metadata file, or having this target instance information exposed in the targets.xml file.	Required if connection and credential information parameters are not used.
OracleHome	string	Database's Oracle Home (used in conjunction with OidRepSchemaName).	Required when OidRepSchemaName is used.
Credential Information			
UserName	string	user name	Required
password	string	user password	Optional; default is ""
Role	string	Role used when connecting to the database (e.g., SYSDBA)	optional; allowed choices are SYSDBA, SYSOPER, and NORMAL (default)
General			
STATEMENT	string	SQL statement or PL/SQL block	Required unless FILENAME is specified.
FILENAME	string	Full path of the file containing the SQL statement or PL/SQL block	Required unless STATEMENT is specified.
NUMROWS	integer	Maximum number of rows to output.	Required
Bind Parameters			
SQLINPARAM<position>	string	Value of input parameter at position <position>	Zero or more, one for each input parameter.

Table 10–4 (Cont.) SQL Fetchlet Input Parameters

Parameter	Type	Description	Use
SQLOUTPARAMPOS	integer	Position of output parameter, if it exists	There can be exactly one output parameter, if it exists.
SQLOUTPARAMTYPE	string	Type of the output parameter, if it exists.	There can be exactly one output parameter type, if it exists.
transpose	TRUE/FALE	If TRUE, the result is transposed: rows to columns and columns to rows.	

Notes

The SQL Fetchlet uses the Oracle Call Interface (OCI) from libclntsh.so.

The SQL statement or PL/SQL block can be specified either through the STATEMENT property, or via a file whose name is provided through the FILENAME property.

The SQL fetchlet supports input parameters. Input and output parameters are indicated in the SQL/PLSQL text in the usual way, by using ":<number>". Input parameters can be used to bind values to both SQL queries and PL/SQL blocks.

Input parameter values are specified as properties of the form SQLINPARAM<position>. There can be any number of input parameters. The input parameters need to be scalar: input parameters of type ARRAY and STRUCT are not supported.

The SQL fetchlet supports the execution of anonymous PL/SQL blocks (which may call other functions or procedures) to retrieve data. When executing a block of PL/SQL, data is returned to the fetchlet by means of an OUT parameter. There can be exactly one out parameter. It must be of type SQL_CURSOR (a PL/SQL REF CURSOR), or it must be a named type that represents an array of objects. In the latter case, each field of the object represents one column of the table; and each object instance in the array represents one complete row in the table. The OUT parameter position and type are specified by means of the properties SQLOUTPARAMPOS and SQLOUTPARAMTYPE. If an OUT parameter is specified, then the fetchlet assumes it is executing PL/SQL and treats the STATEMENT property as an anonymous PL/SQL block.

Note: When using a SQLOUTPARAMTYPE of type 'ARRAY', you must identify the array as follows:

- If you create the array type specified in the SQLOUTPARAMTYPE from SQL*Plus or any utility ***without*** using double quotation marks to surround the array name, then you must specify the array name using all upper-case letters in the target metadata file for this property. The reason for this because the RDBMS automatically changes the array name to all upper-case.
- If you create the array type specified in the SQLOUTPARAMTYPE from SQL*Plus or any utility using double-quotation marks to surround the array name, then the RDBMS retains the case specified. For this reason, users must specify the array name using the same case used in the target metadata file.

If no OUT parameter is specified, the fetchlet assumes that it is executing a SQL query.

Note that all input parameters to the SQL fetchlet are strings. This means that all other datatypes will have to be converted to strings. This is straightforward for datatypes such as numbers, but not, for example, dates and timestamps. You can pass an absolute date or timestamp by passing a character representation of the value (using a DateFormat class). There is no way currently to pass in a date function, such as SYSDATE or SYSDATE+1. In such case, you could embed the date argument directly in the SQL, for example:

```
begin func1(:1, :2, SYSDATE); end;
```

The other caveat is passing null arguments to a procedure. Consider the following SQL:

```
STATEMENT=begin func1(:1,:2); end;
SQLINPARAM1=null
SQLOUTPARAMPOS=2
SQLOUTPARAMTYPE=fooret
```

Assume that the first argument is intended to be a varchar2. By parameterizing it and passing 'null' as the first argument, what we are really doing is passing the *string* 'null' to the argument, and not a null value. If you intend to pass a null value, do the following:

```
STATEMENT=begin func1(null, :1); end;
SQLOUTPARAMPOS=1
SQLOUTPARAMTYPE=fooret
```

Examples

The following properties execute a query (get all users) with no parameters:

```
MachineName=skini-pc
Port=1521
SID=o817
UserName=scott
password=tiger
STATEMENT=select * from all_users;
NUMROWS=30
```

The following properties execute a query (get the first few objects of a specified type owned by a specified user) with input parameters:

```
MachineName=skini-pc
Port=1521
SID=o817
UserName=scott
password=tiger
STATEMENT=select * from all_objects where owner=:1 and object_type=:2 and
rownum<:3tt>
NUMROWS=30
SQLINPARAM1=SYSTEM
SQLINPARAM2=INDEX
SQLINPARAM3=10
```

The following example executes a PL/SQL procedure that returns a cursor and has input parameters:

```
MachineName=skini-pc
Port=1521
SID=o817
UserName=scott
password=tiger
STATEMENT=begin :1 := skini_junk.func1(:2); end;
NUMROWS=30
SQLINPARAM2=SYSTEM
SQLOUTPARAMPOS=1
SQLOUTPARAMTYPE=sql_cursor
```

The following example specifies a PL/SQL procedure that returns an array of strings:

```
MachineName=skini-pc
Port=1521
SID=o817
UserName=scott
password=tiger
STATEMENT=begin skini_junk.newproc(:1,:2); end;
NUMROWS=30
SQLINPARAM1=SYSTEM
SQLOUTPARAMPOS=2
SQLOUTPARAMTYPE=my_string_array
```

The following example specifies a PL/SQL package that returns an array of structures:

```
MachineName=skini-pc
Port=1521
SID=o817
UserName=scott
password=tiger
STATEMENT=begin :1 := skini_junk.func2(:2,:3,:4,:5,:6); end;
NUMROWS=30
SQLINPARAM2=somename
SQLINPARAM3=someplace
SQLINPARAM4=someanimal
SQLINPARAM5=something
SQLINPARAM6=22
SQLOUTPARAMPOS=1
SQLOUTPARAMTYPE=my_struct_array
```

The PL/SQL used in the examples above is provided below for reference:

```
create or replace type my_type as Object (
```

```
    name varchar2(128),
    place varchar2(128),
    animal integer,
    thing number,
    thing2 number);
/
create or replace type my_struct_array as table of my_type;
/

create or replace type my_string_array as table of varchar2(3000);
/

create or replace type my_int_array as table of integer;
/

create or replace package skini_junk as

type Jcr is ref cursor;

function func1(username in varchar2) return Jcr;
function func2(name varchar2, place varchar2, animal integer,
             thing number, thing2 number) return my_struct_array;
procedure newproc(name varchar2, outArray OUT my_string_array);
procedure newproc2(numrows in varchar2, outArray OUT my_int_array);

end skini_junk;
/


create or replace package body skini_junk as

function func1(username in varchar2) return Jcr is
cr Jcr;
begin
    open cr for select object_name, object_type, status from all_objects where
         owner=upper(username);

     return cr;
end;

function func2(name varchar2, place varchar2, animal integer,
             thing number, thing2 number) return my_struct_array IS
ret my_struct_array := my_struct_array();

begin
    ret.extend(50);

    for i in 1..50 loop
        ret(i) := my_type(name || i,
                          place || i,
                          animal+i,
                          thing+i,
                          thing2+i);
    end loop;
    return ret;
end;

procedure newproc(name varchar2, outArray OUT my_string_array) IS
begin
    outArray := my_string_array();
```

```
    outArray.extend(100);

    for i in 1..100 loop
        outArray(i) := name || i;
    end loop;
end;

procedure newproc2(numrows in varchar2, outArray OUT my_int_array) IS
begin
    outArray := my_int_array();
    outArray.extend(numrows);
    for i in 1..numrows loop
        outArray(i) := i;
    end loop;
end;

end skini_junk;
/
```

10.3 SNMP Fetchlet

An *object identifier* (OID) corresponds to either a MIB variable instance or a MIB variable with multiple instances. Given a list of (OIDs), the SNMP Fetchlet polls an *SNMP agent* on a given host for corresponding instances.

Input Parameters

Table 10–5 SNMP Fetchlet Input Parameters

Parameter	Type	Description	Use
hostname	string	Host name of the SNMP agent	Required examples: "bigip.us.oracle.com" "148.87.10.5"
PORT	string	Port of the SNMP agent	Optional. Default is "161"
COMMUNITY	string	SNMP community string	Optional; Default is "public"
TIMEOUT	STRING	SNMP timeout.	Optional. Default is five seconds.
OIDS	string	A list of substrings separated by delimiters. Each substring starts with an OID (in numerical dot notation), and can be optionally ended with *PlacementOID. (See notes for details.)	Required; examples: "1.3.6.1.2.1.2.1.1.1.0,1.3.6.1.2.1.2.1.1.3.0,1.3.6.1.2.1.2.1.1.5.0" "1.3.6.1.2.1.2.1.2.2.1.2 1.3.6.1.2.1.2.1.2.2.1.10 1.3.6.1.2.1.2.1.2.2.1.16" "1.3.6.1.2.1.2.2.1.3 1.3.6.1.2.1.2.2.1.5 1.3.6.1.2.1.4.20.1.1*1.3.6.1.2.1.4.20.1.2 1.3.6.1.2.1.4.20.1.3*1.3.6.1.2.1.4.20.1.2"
DELIM	string	A delimiter to separate individual substrings in OIDS.	Optional; default is whitespace characters .(dot), *(star) and 0-9 (digits) cannot be used as delimiters
TABLE	string	Each OID in OIDS corresponds to a variable with multiple instances if this parameter is "TRUE" and to a single variable instance if it is "FALSE".	Optional; default is "FALSE"

Error Handling

MissingParameterException is thrown if either hostname or OIDS is not given. FetchletException is thrown if TABLE is not equal to either TRUE or FALSE, an I/O

error occurs while sending/receiving SNMP messages to/from the agent, or the agent responds with an SNMP error.

Notes

The table returned by the fetchlet will contain a column for every OID in OIDS. If input OIDs correspond to single variable instances, the table will have just one row with those instances. On the other hand, if the OIDs correspond to variables with multiple instances, each column in the table will contain instances for its OID and each row will correspond to a different *subidentifier*. (A subidentifier is an OID extension that uniquely identifies a particular variable instance for some MIB variable.) OIDS must contain either all OIDs with subidentifiers or all OIDs without the subidentifiers.

For example, to request the variable instances for the three OIDs: sysDescr, sysUpTime, and sysName, OIDS would have to be "1.3.6.1.2.1.2.1.1.1.0 1.3.6.1.2.1.2.1.1.3.0 1.3.6.1.2.1.2.1.1.5.0". In this case, all OIDs contain the instance subidentifier, ".0". The return table would appear as follows (the actual values may be different):

Figure 10–4 SNMP Fetchlet

Sun SNMP Agent, Ultra-4	32504340	nedc-view3

This figure shows the output of the SNMP fetchlet

****

Alternatively, assume that some MIB contains the following 3 columns and 4 instances:

Figure 10–5 SNMP Fetchlet: Columns 3 and 4 Content

ifDescr (network interface description)	ifInOctets (bytes into an interface)	ifOutOctets (bytes out of an interface)
OID: 1.3.6.1.2.1.2.1.2.2.1.2	OID: 1.3.6.1.2.1.2.2.1.10	OID:1.3.6.1.2.1.2.2.1.16
subidentifier : variable instance	subidentifier : variable instance	subidentifier : variable instance
1: wx1	1: 26150844	1:29368527
2: wx2	2: 2763185941	2:3023812977
3: wx3	3:123615396	3:2055140730
4: wx 4	4: 2068257723	4:3212899913

SNMP Fetchlet: Columns 3 and 4 Content

****

To construct a table with 3 columns corresponding to ifDescr, ifInOctets, and ifOutOctets, OIDS would have to be defined as follows

"1.3.6.1.2.1.2.1.2.2.1.2 1.3.6.1.2.1.2.1.2.2.1.10 1.3.6.1.2.1.2.1.2.2.1.16"

The fetchlet would return:

Figure 10–6 SNMP Fetchlet:ifDescr, ifInOctets, and ifOutOctets OIDS

wx1	26150844	29368527
wx2	2763185941	3023812977
wx3	123615396	2055140730
wx 4	2068257723	3212899913

SNMP Fetchlet:ifDescr, ifInOctets, and ifOutOctets OIDS

****

The rows correspond to subidentifiers 1,2,3,4 respectively.

Any OID in OIDS can be appended with another *placement* OID. The variable instances for the placement OID do not appear in the returned table. Instead, they determine the place for the variable instances of the original OID within a column. In particular, for every variable instance I with subidentifier S in the set of instances for the original OID, (a) there must exist a variable instance X with subidentifier S in the set of instances corresponding to the placement OID, and (b) X is used as the subidentifier for the instance I.

For example, consider a MIB containing the following 3 columns, each with 4 variable instances:

Figure 10–7 SNMP Fetchlet: MIB Content with 4 Variable Instances

ifDescr (network interface description)	ipAdEntNetMask (netmask)	ipAdEntIfIndex (network interface index)
OID: 1.3.6.1.2.1.2.1.2.2.1.2	OID: 1.3.6.1.2.1.2.1.4.20.1.3	OID: 1.3.6.1.2.1.2.1.4.20.1.2
subidentifier : variable instance	subidentifier : variable instance	subidentifier : variable instance
1: wx1	IP1: 255.255.255.0	IP1: 3
2: wx2	IP2: 255.255.128.0	IP2: 1
3: wx3	IP3: 255.255.255.240	IP3: 4
4: wx 4	IP4: 255.255.254.0	IP4: 2

SNMP Fetchlet: MIB Content with 4 Variable Instances

****

To construct a table containing ifDescr and ipAdEntNetMask, OID of ipAdEntIfIndex would have to be used as the placement OID to "align" the columns. Thus, the OIDS input to the fetchlet would be "1.3.6.1.2.1.2.1.2.2.1.2 1.3.6.1.2.1.2.1.4.20.1.3*1.3.6.1.2.1.2.1.4.20.1.2". The fetchlet output will be as follows:

Figure 10–8 SNMP Fetchlet: Table Containing ifDescr and ipAdEntNetMask

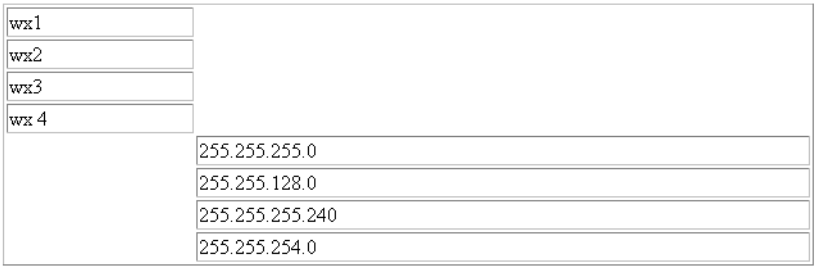
wx1	255.255.128.0
wx2	255.255.254.0
wx3	255.255.255.0
wx 4	255.255.255.240

SNMP Fetchlet: Table Containing ifDescr and ipAdEntNetMask

****

If OIDS were "1.3.6.1.2.1.2.1.2.2.1.2 1.3.6.1.2.1.2.1.4.20.1.3" for the previous example, the output would be as follows:

Figure 10–9 SNMP Fetchlet: Alternate OID



SNMP Fetchlet: Alternate OID

****

10.4 URL Timing Fetchlet

The URL Timing Fetchlet gets the contents of a given URL, timing not only the base page source but any frames or images in the page as well.

Input Parameters

Table 10–6 URL Timing Fetchlet Input Parameters

Parameter	Description	Use
url#	URL(s) to download. "url0" is required but any number of URLs can be specified beyond url0 that following the convention: url0, url1, url2, url3.	Required.
proxy_host	Proxy host used to make a URL connection.	Optional. Specifies the proxy to be used for accessing URLs. If the proxy_host_override value is provided, then that value will be used instead.
proxy_port	Port used by the proxy host used make the URL connection.	Optional.
dont_proxy_for	Domains for which the proxy will not be used.	Optional. For example, .us.oracle.com, .uk.oracle.com
use_proxy	When used in conjunction with the proxy override input parameters, use_proxy specifies a proxy to be used in lieu of the original proxy. When set to false without the proxy override parameters set, no proxy is used.	Optional. Parameter can be set to true or false.
proxy_host_ override	Alternate proxy host used to make the URL connection.	Optional. Overrides proxy_host.

Table 10–6 (Cont.) URL Timing Fetchlet Input Parameters

Parameter	Description	Use
proxy_port_ override	Alternate proxy port used to make the URL connection.	Optional. Overrides proxy_port.
dont_proxy_ override	Do not use the proxy for domains.	Optional. Parameter can be set to true or false.
internet_cert_loc	Path pointing to the location of a certificate to be used to access a secure (HTTPS) URL.	Optional.
auth_realm	Realm for the Basic Authentication log on. If the realm is not specified for the authentication, authentication does not occur and the download of the page fails with a 401 response code.	Optional.
auth_user	Username for Basic Authentication.	Optional.
auth_password	Password for Basic Authentication.	Optional.
retries	Number of times to retry the url if it initially fails.	Optional. Default = 1
connection_ timeout	Wait time (in milliseconds) allowed to establish a connection to a server. This time also includes time required for name resolution.	Optional. Default= 60000 milliseconds (1 minute)
read_timeout	Idle time in the read waiting for the server to respond. For example, if no data is received from the server during the specified timeout period, the operation is considered failure.	Optional. Default = 12000 milliseconds (2 minutes)
timeout	Number of milliseconds after which the page download is considered a failure. This will detect if the site is functional but is extremely slow.	Optional. Default = 300000 ms (5 minutes)
status_ comparator	When collating the rows to make a single row, the status_comparator parameter will indicate whether all URLs should have been a success (and) or any URLs should have been a success (or).	Optional. Default = and
cache	Indicates whether to use a cache when accessing an URL. Set the parameter to "n" to specify that no cache be used.	Optional. Default = y Note: The scope of the cache is per request. There is no persistent cache across multiple get metric requests.
output_format	Specifies the output format to be used: summary, detailed, repeat_column. For more information on output formats, see Metric Columns and Output Modes on page 10-19.	Required. summary : gives a default set of metrics in a single row for all urls detailed: gives a default set of metrics for each url. repeat_column : gives a single row of metric with timing for each of the url.

Table 10–6 (Cont.) URL Timing Fetchlet Input Parameters

Parameter	Description	Use
metrics	Specifies which metric columns need to be returned. For more information on metrics columns returned for each output format, see Table 10–8, " URLTIMING Fetchlet: Metric Columns"	Optional. Allows you to specify of what needs to be returned from the fetchlet and in which order. Example: status, status_description, total_response_time

Metric Columns and Output Modes

The format of information and specific metric information returned are controlled by the "output_format" and "metrics" input parameters. The following table lists the format categories and the metrics (columns) returned by each. For a description of available metric columns, see Table 10–8, " URLTIMING Fetchlet: Metric Columns"

Table 10–7 URLTIMING Fetchlet: Output Formats

Output Format	Description	Metric Columns
summary	Returns a default set of metrics in a single row for all URLs If the metrics input parameter is specified, then only the columns specified will be returned.	computed_response_time, status, status_description, dns_time, connect_time, redirect_time, first_byte_time, html_time, content_time, total_response_time, rate, max_response_time, avg_response_time, avg_connect_time, avg_first_byte_time, broken_count, broken_content
detailed	Returns a default set of metrics for each url. If the metrics input parameter is specified, then only the columns specified will be returned.	url, computed_response_time, status, status_description, dns_time, connect_time, redirect_time, first_byte_time, html_time, content_time, total_response_time, rate, redirect_count, html_bytes, content_bytes, total_bytes, avg_connect_time, avg_first_byte_time, broken_count, broken_details
repeat_column	Returns a single row of metrics with timing for each of the URLs. If the metrics input parameter is specified, then those columns will be returned for each url followed by overall status and status_description. (Note the output will always be single row).	total_response_time repeated for each URL followed by overall status and status_description.

Metric Columns

The following table shows the metric columns returned by the URLTIMING fetchlet.

Table 10–8 URLTIMING Fetchlet: Metric Columns

Column Name	Description
status	The overall status of all URLs. By default AND is used to compute overall status but this can be changed using the status_ comparator input parameter.
connect_time	The time to connect to the server and send the request.
first_byte_time	Time taken between sending the request and reading the first byte from the response.
total_response_time	Time taken for fetching ALL urls and associated content (gif, css, javascript etc).
max_response_time	Also referred as Slowest page time. This the time taken by the slowest URL.
avg_response_time	Average response time for URL. Computed as total response time / number of pages (urls).
rate	Kilo Bytes per second. This is computed by total bytes received / total time taken to receive them.
html_time	Total time taken to download the html part of all pages. This time excludes time to fetch images and other contents. (Includes time to fetch frame html).
content_time	Time taken to download the page content (gif, javascripts, css etc.).
redirect_time	Total time taken for all redirects occurring while fetching the set of urls specified.
redirect_count	Number of redirects.
total_bytes	Total number of bytes.
html_bytes	Total number of HTML bytes. (Includes bytes for frame html).
content_bytes	Total number of content bytes.
status_description	This is present only when the status is down. Corresponds to HTTP Status description.
request_count	Number of request made. (Includes all html as well as content requests).
broken_count	Number errors while fetching images or other content elements.
broken_details	List of images or other content elements that could not be fetched. This has format of url[broken list], url[broken list...
computed_response_time	This time approximates the time it would have taken for a client (like browser) to fetch all the pages in the transaction. This number is computed as if the contents of every page (gifs, css etc) were fetched using multiple threads.
avg_connect_time	Total connect_time / total number of connections made.
avg_first_byte_time	Total First Byte Time / Number of requests made (either to fetch HTML or content).
dns_time	Time to resolve host name (not implemented, always returns zero).
url	Returns the url, can only be used in 'detailed' output_format.

Example

Let's take the following URL:

```
url0=http://www.oracle.com/
```

With the input parameter output_format=summary, the fetchlet returns the following table (minus the headers on the columns):

Figure 10–10 Summary Output Format

computed response time	status	status description	dns time	connect time	redirect time	first byte time	html time	content time	total response time	rate (Kbytes per second)	max response time	avg response time	avg connect time	avg first byte time	broken count	broken content
540	1		0	548	0	149.0	1.0	7.0	705	95.16	705	705	54.80	14.90	0	

Graphic shows summary output format for the urltiming fetchlet.

****

With output_format = summary and metrics = total_response_time, status, status_description the fetchlet returns the following table (minus the headers on the columns):

Figure 10–11 Summary Output Format with Specified Metric Columns

total response time	Status	Status description
705.0	1	

Graphcic shows summary output format with specified metric columns.

****

With output_format = summary and metrics = total_response_time, status, status_description the fetchlet returns the following table (minus the headers on the columns) and the server is giving error:

Figure 10–12 Summary Output Format with Specified Metric Columns and Internal Server Error

total response time	status	status description
	0	Internal Server Error -- http://www.oracle.com

Graphic shows summary output format with specified metric columns and internal server error.

****

Let's take the following URL:

```
url0=http://www.oracle.com/
```

```
url1=http://nedc.us.oracle.com/
```

With the output_format=summary, the fetchlet returns the following table (minus the headers on the columns). Here the numbers are time taken for fetching both the urls.

Figure 10–13 Summary Output Format for Two URLs

computed response time	status	status description	dns time	connect time	redirect time	first byte time	html time	content time	total response time	rate (Kbytes per second)	max response time	avg response time	avg connect time	avg first byte time	broken count	broken content
4344	1		0	603	0	7277	438	318.0	8636	16.92	8098	4318	22.33	269.52	0	

Graphic shows summary output format for two urls.

****

With the output_format=detailed, the fetchlet returns the following table (minus the headers on the columns):

Figure 10–14 Detailed Output for Two URLs

url	uri	computed response time	status	status description	dns time	connect time	redirect time	first byte time	html time	content time	total response time	rate	redirect count	html bytes	content bytes	total bytes	avg connect time	avg first byte time	broken count	broken content
http://www.oracle.com/	not supported	531.0	1			548	0	149.0	1	7	705	95.16	0	18207	48883	67090	54.80	14.90	0	
http://nedc.us.oracle.com/	not supported	4424.0	1			480	0	6227.0	442	230	7379	10.71	0	24627	54427	79054	28.24	366.29	0	

Graphic shows detailed output for two urls.

****

With the output_format=repeat_column, the fetchlet returns the following table (minus the headers on the columns):

Figure 10–15 Repeat Column Output Format

total response time(oracle.com)	total response time (nedc)	status	status description
705.0	7379	1	

Graphic shows repeat column output format.

****

Error Handling

Metric error if the URL parameter is missing, malformed, or if the metric cannot be computed.

Notes

The time required to perform a retry will be added on to the total time of the page. For example, if two retries are performed and then a success occurs, the total page time

will be the time of the page that succeeded plus the time it took for the two retries to fail.

Proposed usage:

- For basic monitoring:

  Use url0=<URL to be monitored> , output_mode=summary and specify metrics=status, computed_response_time, status_description

- For getting all columns:

  Use url0=<url to be monitored> , output_mode=summary

10.5 Dynamic Monitoring Service (DMS) Fetchlet

The Dynamic Monitoring Service (DMS) fetchlet contacts an Application Server (AS) and then collects the metrics instrumented by the DMS.

The DMS allows application and system developers to measure and export customized, component-specific performance metrics. The Oracle Management Agent allows software components to import runtime performance data into Oracle Enterprise Manager Grid Control.

The DMS Fetchlet is an Oracle Management Agent plug-in module that allows the Management Agent to import the performance data that is exported by the DMS. Using the DMS fetchlet, any component that is instrumented using DMS API calls may share its performance data with Enterprise Manager Grid Control.

10.5.1 Advantages to Using DMS for Oracle Management Agent Integration

With DMS, a component can insulate itself from the operational details of the Management Agent. A component would not need to deploy (or maintain) its own fetchlet or deploy (or maintain) a Tcl script or shell script to plug into one of the existing fetchlets. A component would not need to devise its own new way of measuring or exporting performance metrics. Performance metrics can be measured and reported in a consistent way across components. The DMS fetchlet contacts the remote DMS runtime directly with no need for forking shell scripts or Tcl scripts. Most importantly, DMS automatically produces the long, complicated metadata document for you and thereby saves many hours of tedious and error-prone hand editing.

Input Parameters

Table 10–9 DMS Fetchlet Input Parameters

Name	Type	Description	Use
oraclehome	String	Top directory under which the monitored IAS instance is installed. It is used only for monitoring local IAS processes. For monitoring remote IAS processes, users should give it an empty value and specify property "opmnremoteport" and/or "machine" instead.	Required. Example: "/private/oracle/ias"
version	String	AS Version number of the target. It is used to distinguish the version of monitored AS instance.	Optional Example: "9.0.4"

Table 10–9 (Cont.) DMS Fetchlet Input Parameters

Name	Type	Description	Use
opmnport	Integer	Oracle Process Monitoring and Notification (OPMN) port. It is used primarily for monitoring remote AS processes. It should be specified together with property "machine". If it is present and valid , property "oraclehome" and "httpport" are ignored.	Optional Example: "6200"
httpport	Integer	HTTP port is used primaryly for monitoring stand-alone processes. It should be specified together with property "machine". It will be ignored, if property "opmnport" is present. If it is present and valid, property "oraclehome" is ignored.	Optional Example: "7777"
machine	String	Host name where the Internet Application Server (AS) instance runs. It should be specified together with property "opmnport". If it is not present, the local host is assumed.	Optional Example: "my-sun.us.oracle.com"
metric	String	Name of the table-type metric.	Required Example: "Servlets"
columnOrder	String	A list of metric column names separated by ";". The column names must be specified in same order as they appear in the target type metadata file. Do not include "name", "host", "process" and "fullname" columns.	Required Example: "processTimes;totalRequest;requestRate"
usecache	String	Whether to cache this metric. The legal values are "true", "false" and "refreshall" with "true" being the default. The "refreshall" value tells the DMS to delete its cache data and retrieve the most recent data from all targets.	Optional. Example: "false" Setting "usecache" to "false" will bypass DMS caching
proxyHost	String	Proxy host through which to make the HTTP connection	Optional Example: "proxy.us.oracle.com"
proxyPort	Integer	Proxy port through which to make the HTTP connection	Optional Example: "80"
dontProxyFor	String	Domains for which the proxy will not be used.	Optional Example: ".us.oracle.com" or "18.219.0"

Table 10–9 (Cont.) DMS Fetchlet Input Parameters

Name	Type	Description	Use
useDefaultProxy	String	When used in conjunction with the proxy override parameters, this variable specifies a proxy other than the original one. When set to false without the proxy override parameters set, no proxy at all is used.	Optional Example: "true" or "false"
proxyHostOverride	String	proxy host through which to make the HTTP connection	Optional Example: "www-proxy.us.oracle.com"
proxyPortOverride	Integer	proxy port through which to make the HTTP connection	Optional Example: "80"
authrealm	String	Realm for the Basic Authentication logon. If the realm is not specified for the authentication, authentication does not occur and the download of the page fails with a 401 response code.	Optional Example: "Please input your flex account login:"
authuser	String	Username for Basic Authentication	Optional "superuser"
authpwd	String	Password for Basic Authentication	Optional Example: "welcome"

Error Handling

DMS Fetchlet throws MissingParameterException if any of the properties "oraclehome", "metric", "columnOrder", "opmnport", or "httpport" is missing. It throws FetchletException if any of the ports given is not valid.

Notes

The first four columns of the metric table returned are always column "name", "fullname", "host" and "process". Therefore, do not include them in columnOrder string. Property "machine" should be specified together with either properties "opmnport" or "httpport". In this case, the property "oraclehome" is ignored.

10.5.2 DMS Fetchlet/Oracle Management Agent Integration Instructions

DMS has been used in several components (such as Apache, JServ, OSE, and Portal) to provide a consistent performance monitoring infrastructure for Oracle 9i Application Server. The Sensors are easy to use and save most of the work related to performance measurement because they hide most of the details related to timing, counting, and categorization. Finally, DMS hides many Management Agent details from component developers and much of the Management Agent integration effort.

10.5.2.1 Integrating DMS Data with the Management Agent

As mentioned earlier, DMS allows application and system developers to measure and export customized, component-specific performance metrics. The Oracle Management Agent enables software components to import runtime performance data into Enterprise Manager Grid Control. This section describes how to integrate DMS performance metrics with the Management Agent.

Step 1: Install AS

Step 2: Install Enterprise Manager Grid Control

Step 3: Instrument your Component with DMS

To enable DMS metrics for Enterprise Manager Grid Control, you must follow two additional rules:

- **Rule 1**: ***All Nouns exported to the Management Agent must have types***
  Noun types can be set either by specifying the "type" parameter in the Noun.create() methods or by using the Noun.setType(String) method. The idea is that every Noun type will be converted automatically to a management repository table. Every Noun of a given type will become a row in the type's corresponding management repository table. The metrics contained by a Noun become columns in the repository table metric. Any Noun without a type will not be exported to Management Agent.
- **Rule 2**: ***All Nouns of a given type must contain a consistent set of Sensor names***
  Because the metrics contained by a Noun become columns in a management repository table, you must make sure that all Nouns of a given type contain the same Sensors. This ensures that each row of the corresponding repository table has the same set of columns. DMS does not check this constraint for you.

For example, the following Java snippet shows how to create typed Nouns that contain a consistent set of Sensors. DMS will automatically convert these into a repository table named "MyType":

```
/* first create the nouns*/
Noun n1 = Noun.create("/myExample/myComponent/noun1", "MyType");
Noun n2 = Noun.create("/myExample/myComponent/noun2", "MyType");

/* next, create the Sensors */
PhaseEvent pe1 = PhaseEvent.create(n1, "criticalPhase", "a critical interval");
PhaseEvent pe2 = PhaseEvent.create(n2, "criticalPhase", "a critical interval");
Event e1 = Event.create(n1, "importantEvt", "an important event");
Event e2 = Event.create(n2, "importantEvt", "an important event");

/* here is a third set that shows the use of Noun.setType(String) */
PhaseEvent pe3 = PhaseEvent.create(
                              "/myExample/myComponent/noun3/criticalPhase",
                              "a critical interval");
Event e3 = PhaseEvent.create(
                              "/myExample/myComponent/noun3/importantEvt",
                              "an important event");
Noun n3 = Noun.get("/myExample/myComponent/noun3");
n3.setType("MyType");
```

For this example, the "MyType" table will contain three rows and four columns. Besides the columns corresponding to the two Sensors, there will be a "name" column and a "path" column that will contain the DMS path name including the process name and "/myExample/myCom...".

If these Nouns/Sensors are created in several servlet engines within the AS site, then the AggreSpy will find each of the servlet engines and will aggregate all of the Nouns/Sensors into a single MyType table.

Step 4: Generate your Target Metadata Document

You can generate the Target Metadata Document using your browser. Point your browser to your AS site that you want to monitor using the following URL:

```
http://YOUR_AS_HOST:YOUR_AS_PORT/YOUR_SERVLET_PATH/AggreSpy?format=targetmetadata
```

You should use the actual host, port and servlet path of your AS installation in the above URL. The servlet path usually defaults to "servlet". The XML document you get is the Target Metadata Document for your AS site. The first comment of the XML document explains where you can obtain the Target Metadata Document and instructions telling you what needs to be done to this document.

Step 5: Install the Target Metadata Document

Follow the steps described in the first comment of the XML document. Save the XML document to a file called "oracle_dms.xml" under the "metadata" directory of your Enterprise Manager installation (OMS_ORACLE_HOME/sysman/admin/metadata/). If you want to monitor a subset of the metrics or merge the metrics with the ones in the existing "oracle_dms.xml" file, you should save this new defintion to a separate file called target_name.xml. You will also need to change the Target Type entry in the generated metadata document.

Next, you should add the target instance information of your AS site to your "targets.xml" file residing under the top directory of your Enterprise Manager installation. You can find a block of XML tags in the comment you read. They look like:

```
<Target Type='oracle_dms' NAME='DMS_YOUR-IAS-HOST_YOUR-IAS-PORT' VERSION='2.0'>
 <Property NAME='host' VALUE='YOUR_IAS_HOST' />
 <Property NAME='port' VALUE='YOUR_IAS_PORT' />
 <Property NAME='dmsPath' VALUE='YOUR_SERVLET_PATH' />
</Target>
```

Copy this block and paste it to the "targets.xml" file between <targets> and </targets> tags.

Finally, to add the new target metadata file and target instance information from the targets.xml file to Enterprise Manager Grid Control, you must run the following command:

```
>$ORACLE_HOME/bin/emctl reload
```

Step 6: View your metrics

You are ready to view your metrics using Enterprise Manager's Metric Browser. See "Activating the Metric Browser" on page 2-10 for information on setting up the Metric Browser. First, make sure that AS and your component are still running. Next, restart the Oracle Management Agent. Finally, point your browser to your Management Agent installation using the following URL:

```
http://<YOUR_AGENT_HOST>:<YOUR_AGENT_PORT>/emd/browser/main
```

The Management Agent port information can be found in the $AGENT_HOME/sysman/config/emd.properties file at the EMD_URL line.

You should use the actual host and port of your Management Agent installation in the above URL. You will find your AS site listed as the target "DMS_YOUR-AS-HOST_ YOUR-AS-PORT". If you click on the link, you will see a list of metric IDs. You can browse your metrics by clicking on the respective metric IDs.

10.6 HTTP Data Fetchlets

The HTTP data fetchlets obtain the contents of a URL and returns the contents of the URL as data. Three fetchlets are available:

- URL Fetchlet
- URL Lines
- URL Lines Token

10.6.1 URL Fetchlet (raw)

The URL Fetchlet gets the contents of a given URL and returns the contents of the URL in a single cell table.

To get the output of a URL tokenized by lines and each line tokenized by a given delimiter, see the URL Line Token Fetchlet.

Input Parameters

Table 10–10 URL Fetchlet Input Parameters

Name	Description	Use
url	URL to retrieve the contents of	required
proxyHost	proxy host through which to make the URL connection.	optional
proxyPort	proxy port through which to make the URL connection.	optional

Example

Let's take the following URL:

```
http://localhost/nhcities.txt
```

It has the following contents:

Line 1: Nashua, Keene,

Line 2: Concord

Line 3: , Conway, Manchester, Milford, Brookline,

Line 4:

Line 5: Hollis, Meredith

Now let's run the URL Fetchlet with the given URL.
The fetchlet returns the following one-by-one table:

Figure 10–16 URL Fetchet Output

URL Fetchet Output

****

The raw contents of the URL is returned.

Error Handling

MissingParameterException if URL parameter is missing.
FetchletException if the URL is malformed or an I/O error occurs in retrieving the content of the URL.

10.6.2 URL Lines Fetchlet (split into lines)

The URL Fetchlet gets the contents of a given URL and tokenizes the contents of the URL. The output is tokenized by lines. The fetchlet returns the tokens in a single column table. The nth row in the table represents the nth line of the URL contents.

Note: To get the raw, untokenized contents of a URL, see the URL Fetchlet. To get the contents of a URL tokenized by lines and each line tokenized by a given delimiter, see the URL Line Token Fetchlet.

Table 10–11 URL Lines Fetchlet Input Parameters

Name	Description	Use
url	URL to retrieve the contents of	required
proxyHost	proxy host through which to make the URL connection.	optional
proxyPort	proxy port through which to make the URL connection.	optional
startsWith	only lines starting with this string are included in the result	optional; default = "" (all lines are included)

Example

Let's take the following URL:

```
http://localhost/nhcities.txt
```

It has the following contents:

Line 1: Nashua, Keene,

Line 2: Concord

Line 3: , Conway, Manchester, Milford, Brookline,

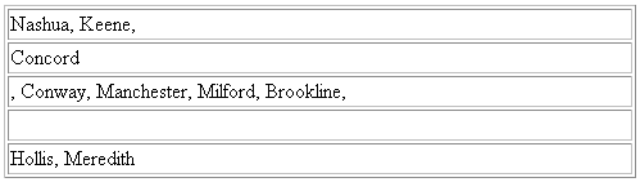
Line 4:

Line 5: Hollis, Meredith

Now let's run the URL Fetchlet with the given URL.

The fetchlet returns the following table:

Figure 10–17 URL LInes Fetchlet Output



Nashua, Keene,
Concord
, Conway, Manchester, Milford, Brookline,
Hollis, Meredith

URL LInes Fetchlet Output

****

Error Handling

MissingParameterException if URL parameter is missing.

FetchletException if the URL is malformed or an I/O error occurs in retrieving the content of the URL.

Notes

Lines are tokenized using "\n".

10.6.3 URL Line Token Fetchlet (tokenized lines)

The URL Fetchlet gets the contents of a given URL and tokenizes the contents of the URL. The output is tokenized first by lines, and then each line is tokenized by a given delimiter set. The fetchlet returns the tokens in a table. The nth row in the table represents the nth line of the URL content. The *n*th column in the table represents the nth token in a line as determined by the given delimiter set.

To get the raw, untokenized contents of a URL, see the URL Fetchlet.

Table 10–12 URL Line Token Fetchlet Input Parameters

Name	Description	Use
url	URL to retrieve the contents of	required
delimiter	set of characters that act as delimiters to tokenize the lines	optional; default = "" (just breaks output into lines)
proxyHost	proxy host through which to make the URL connection.	optional
proxyPort	proxy port through which to make the URL connection.	optional
startsWith	only lines starting with this string are included in the result	optional; default = "" (all lines are included)

Example

Let's take the following URL:

```
http://localhost/nhcities.txt
```

It has the following contents:

Line 1: Nashua, Keene,

Line 2: Concord

Line 3: , Conway, Manchester, Milford, Brookline,

Line 4:

Line 5: Hollis, Meredith

Now let's run the URL Fetchlet with the given URL and a single character "," for the delimiter.

The fetchlet returns the following table:

Figure 10–18 URL Token Lines Output

Nashua	Keene		
Concord			
Conway	Manchester	Milford	Brookline
Hollis	Meredith		

URL Token Lines Output

****

Error Handling

MissingParameterException if URL parameter is missing.

FetchletException if the URL is malformed or an I/O error occurs in retrieving the content of the URL.

Notes

Lines are tokenized using "\n".

The delimiter can be a single character or a set of characters. For example, it can be "|+_", if the line should be broken up by pipes, pluses, and underscores. If two or more delimiters are together in the output text, such as "||" or "+|+", then it is as if there are empty string tokens between them. These empty strings get columns in the result table. It is NOT considered that there are empty strings preceding a delimiter that starts a line or following a delimiter that ends a line.

In order to express non-printable characters in the delimiter set (such as tabs) in XML, use "&#xHH;" where H is the hexadecimal identifier for the character.

10.7 URLXML Fetchlet

The URL XML Fetchlet obtains the XML content of a given URL, and extracts information based on a given pattern. A pattern is a list of "chunks" of XML to match against. The return table is a table with a column for each grabber (*) in the pattern in order and a row each time the pattern chunk matches in the XML content.

Input Parameters

Table 10–13 URLXML Fetchlet Input Parameters

Name	Description	Use
url	URL to retrieve the contents of	Required.
pattern	The pattern used to extract information from XML; this is a list of XML chunks that that is compared against the XML content of the URL. Each chunk contains one or more "grabbers" (*) in the text portion of the elements that define what should the flattened into text and extracted.	Required.
proxyHost	The proxy host through which to make the URL connection.	Optional.
proxyPort	The proxy port through which to make the URL connection.	Optional.
ignoreDtd	If set to TRUE, specifies that the DTD file referenced by the content XML should be ignored. This is useful in cases where the DTD file cannot be accessed.	Optional.
generateKey	If set to true, a unique key will be generated for each row. The key will occupy the first column of the result, and will be numeric.	Optional.
throwConnException	If set to TRUE, a java.net.ConnectException will be thrown. Otherwise, it will be caught and an empty result set will be returned. Setting this property to FALSE provides behavior which is consistent with the DMSFetchlet.	Optional. The default value is TRUE.

Example

Let's take the following URL:

```
http://localhost/urlxmltestfile.xml
```

It has the following content:

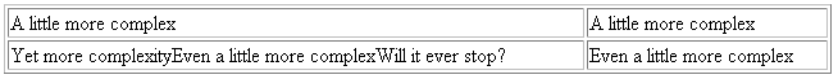
```
<?xml version="1.0"?>
<testfile>
    <test>Simple text</test>
    <test><level>A little more complex</level></test>
    <test></test>
    <notatest></notatest>
    <test>Yet more complexity<level>Even a little more complex</level>Will it ever
stop?</test>
    <test1>must match<level>extract me!</level></test1>
    <test1>must match here<level>extract me, too!</level></test1>
</testfile>
```

Running the URL XML Fetchlet with the given URL and the pattern:

```
<testfile><test>*<level>*</level></test></testfile>
```

returns the following table:

Figure 10–19 URL XML Fetchlet Output



A little more complex	A little more complex
Yet more complexityEven a little more complexWill it ever stop?	Even a little more complex

URL XML Fetchlet Output

****

Error Handling

MissingParameterException if URL or pattern parameters are missing.

A FetchletException is generated if:

- The URL is malformed.
- An I/O error occurs in retrieving the content of the URL.
- The URL contents or pattern contains invalid XML.

Notes

Setting the proxy host and/or port changes these settings for the java.net package for the whole Java environment and is not thread-safe if the proxy settings are changing.

10.8 WBEM Fetchlet

The WBEM fetchlet accesses a CIMOM and retrieves requested information using the specified CIM class. The CIM class is mapped to a Management Repository table metric. The name of the CIM class is the name of the table metric that is returned, and the properties defined for the CIM class are used to name the table columns for the metric. The properties of interest must be specified during metric definition.

The fetchlet returns the instances that have been instantiated for the CIM class as rows of the Management Repository table metric.

Input Parameters

Table 10–14 WBEM Fetchlet Input Parameters

Name	Type	Description	Use
hostname	String	Host name of the CIMOM	Optional; default is "localhost"
port	Integer	Port for the CIMOM	Optional; default is 5988
namespace	String	CIM Namespace	Optional; default is "root/cimv2"
username	String	Username to use for CIMOM authorization on the host where the CIMOM is running	Required
password	String	Password to use for CIMOM authorization on the host where the CIMOM is running	Required

Table 10–14 (Cont.) WBEM Fetchlet Input Parameters

Name	Type	Description	Use
CIMclassname	String	Name of the CIM class whose instances will be returned	Required for all operations except STATUS. STATUS operations just check whether the CIMOM is running, so a class name is not needed.
operation	String	Operation to be performed. Supported operations include COUNT, which returns a count of the number of instances in the class, VALUES, which returns the values of the specified properties for each instance of the class, or STATUS, which provides status information about the CIMOM.	Optional, default is VALUES
properties	String	The property names from the CIM class definition that we are interested in collecting.	Required for VALUES operation. If the operation is VALUES, we can have 1 to N of these, separated by a semicolon. If the operation is VALUES, and no properties are provided, an error is returned. Properties are handed to the EMD in the order that they are specified.

Error Handling

The following types of errors have been identified for the WBEM fetchlet.

MissingParameterException occurs when:

- No CIM Class parameters match.

Fetchlet exception occurs when:

- The class name is not found in the CIMOM namespace.
- The namespace is not found.
- The connection to the CIMOM does not have valid credentials.
- The connection to the CIMOM failed because the CIMOM was not running.
- The CIM class property does not exist
- An unsupported operation was specified
- No properties were specified.

Notes

Ports: Some CIMOM client interfaces expose the port that the CIMOM is listening on while some clients do not. To cover both cases, the port is exposed as an optional input parameter that defaults to port 5988. This is the default Pegasus CIMOM listener port. The Java API that is provided through Sun's Wbem Services does not expose the CIMOM port.

Protocols: Most CIMOMs support either an RMI or HTTP protocol for communicating with the CIMOM. The testing that has been done shows that the HTTP protocol is not as stable, and in some cases, not fully implemented in the CIMOM. Because of this, the protocol currently defaults to RMI. The actual parameters for the WBEM Services CIMOM for the protocol are: CIMClient.CIM_RMI or CIMClient.CIM_XML.

Fetchlet Operations: The WBEM APIs are very flexible at allowing clients to traverse the class hierarchies that are defined and their associations. At this point in time, the options on accessing CIM data from an EMD are restricted to counting, getting the properties of classes, and CIMOM status. These are the more important operations that need to be performed for monitoring. As additional requirements come in, we can add new operations to support them if necessary. For the prototype, only the count operation has been implemented.

Authentication: Most CIMOMs provide APIs to support authentication through a user identity mechanism. The majority of the CIMOMs have not implemented the API, so this capability is really a no-op. In any case, we've supplied the capability in the fetchlet so that as CIMOM implementations catch up with the standard, we'll have the necessary support in place.

Examples

The Wbem fetchlet supports three basic operations. At this point, the fetchlet only handles one operation at a time, so you cannot mix count, status, and value operations within a single fetchlet call. The following example shows how to write the metadata for a COUNT operation:

Example 10–1 COUNT Operation Metadata

```
<Metric NAME="Load" TYPE="TABLE">
    <Display>
       <Label NLSID="wbem_cimom_load">Load</Label>
     </Display>
     <TableDescriptor>
       <ColumnDescriptor NAME="Active Clients" TYPE="NUMBER" IS_KEY="FALSE">
           <Display>
             <Label NLSID="wbem_cimom_active_clients">Active CIMOM Clients</Label>
           </Display>
         </ColumnDescriptor>
     </TableDescriptor>
    <QueryDescriptor FETCHLET_ID="Wbem">
       <Property NAME="username" SCOPE="GLOBAL">guest</Property>
       <Property NAME="password" SCOPE="GLOBAL">guest</Property>
       <Property NAME="CIMClassname" SCOPE="GLOBAL">EX_SFLProvider</Property>
       <Property NAME="operation" SCOPE="GLOBAL">COUNT</Property>
    </QueryDescriptor>
  </Metric>
```

The FETCHLET_ID is identified as Wbem. Property names are passed to the fetchlet for the required parameters username, password, and CIMClassname. The operation is identified as COUNT.

The following example shows how to implement a Response Status metric to determine whether the CIMOM is running or not. It returns a value of 1 if the connection to the CIMOM is successful, otherwise 0.

Example 10–2 Response Status Metric

```
  <Metric NAME="Response" TYPE="TABLE">
     <Display>
       <Label NLSID="wbem_cimon_response">Response</Label>
```

```
    </Display>
    <TableDescriptor>
      <ColumnDescriptor NAME="Status" TYPE="NUMBER" IS_KEY="FALSE">
          <Display>
            <Label NLSID="wbem_cimom_response_status">Status</Label>
          </Display>
        </ColumnDescriptor>
   </TableDescriptor>
   <QueryDescriptor FETCHLET_ID="Wbem">
      <Property NAME="username" SCOPE="GLOBAL">guest</Property>
      <Property NAME="password" SCOPE="GLOBAL">guest</Property>
      <Property NAME="operation" SCOPE="GLOBAL">STATUS</Property>
    </QueryDescriptor>
  </Metric>
```

The default operation is the VALUES operation. It is used to fetch the values of a class that is defined in the CIMOM.

In the final example, the EX_Teacher class is accessed and fetches the name column. Name is the key of the class and of the new metric being defined, so the IS_KEY property is set to true. The CIM class properties will be mapped to the Enterprise Manager columns in the order that they are specified in the properties property. In this case, there is only 1 property - Name.

Example 10–3 Single Property Fetched for a Class

```
<Metric NAME="EX_Teacher" TYPE="TABLE">
     <Display>
       <Label NLSID="wbem_EX_Teacher">EX_Teacher Class</Label>
     </Display>
     <TableDescriptor>
       <ColumnDescriptor NAME="Name" TYPE="STRING" IS_KEY="TRUE">
           <Display>
             <Label NLSID="wbem_ex_teacher_name">Name</Label>
           </Display>
         </ColumnDescriptor>
     </TableDescriptor>
     <QueryDescriptor FETCHLET_ID="Wbem">
       <Property NAME="username" SCOPE="GLOBAL">guest</Property>
       <Property NAME="password" SCOPE="GLOBAL">guest</Property>
       <Property NAME="CIMClassname" SCOPE="GLOBAL">EX_Teacher</Property>
       <Property NAME="properties" SCOPE="GLOBAL">Name</Property>
     </QueryDescriptor>
   </Metric>
```

If multiple properties are fetched for a class, semi-colons should separate them. The properties should be provided in the order that the column descriptors are specified for the metric table definition.

Example 10–4 Multiple Properties Fetched for a Class

```
<Metric NAME="EX_SFLProvider" TYPE="TABLE">
     <Display>
       <Label NLSID="wbem_EX_SFLProvider">EX_SFLProvider Class</Label>
     </Display>
     <TableDescriptor>
       <ColumnDescriptor NAME="Name" TYPE="STRING" IS_KEY="TRUE">
           <Display>
             <Label NLSID="wbem_ex_sfl_name">Name</Label>
           </Display>
```

```
      </ColumnDescriptor>
    <ColumnDescriptor NAME="Win" TYPE="NUMBER" IS_KEY="FALSE">
        <Display>
          <Label NLSID="wbem_ex_sfl_win">Win</Label>
        </Display>
      </ColumnDescriptor>
    <ColumnDescriptor NAME="Lost" TYPE="NUMBER" IS_KEY="FALSE">
        <Display>
          <Label NLSID="wbem_ex_sfl_lost">Lost</Label>
        </Display>
      </ColumnDescriptor>
  </TableDescriptor>
  <QueryDescriptor FETCHLET_ID="Wbem">
    <Property NAME="username" SCOPE="GLOBAL">guest</Property>
    <Property NAME="password" SCOPE="GLOBAL">guest</Property>
    <Property NAME="CIMClassname" SCOPE="GLOBAL">EX_SFLProvider</Property>
    <Property NAME="properties" SCOPE="GLOBAL">Name;Win;Lost</Property>
  </QueryDescriptor>
</Metric>
```

10.9 JDBC Fetchlet

Call-level interfaces such as JDBC permit external access to SQL database manipulation and update commands. The Java Database Connectivity (JDBC) fetchlet allows you to execute common JDBC commands and obtain their response time for any type of database.

Input Parameters

Table 10–15 JDBC Fetchlet Input Parameters

Name	Description	Use
Transaction Name	(Standard)	Required.
Beacon Name	(Standard)	Required.

Connect String	Connection string provided by the user. The Connect String must comply with the URL format specified by the vendor of the database to which the user is trying to connect.
	Examples:
	Format required by Oracle:
	`jdbc:oracle:thin:@hostname:port`
	Format required by MySQL:
	`jdbc:mysql://hostname:port`
		Required.

Table 10–15 (Cont.) JDBC Fetchlet Input Parameters

Name	Description	Use
Class Name String	The driver class name to be used for connections. Example: oracle.jdbc.driver.OracleDriver You have two options for configuring the Agent to use the .jar file containing the driver: 1. Place the .jar file in $JAVA_HOME/jre/lib/ext. CLASSPATH does not need to be modified. 2. Place the .jar file anywhere and update CLASSPATH in emd.properties file with the path to jar. Bounce Agent. This should be scripted and be transparent to user.	Required.
Username	Username to be used when connecting to the database.	Required.
Password	Password to be used when connecting to the database.	Required.
Role	User Role	Required.
Statement	SQL statement to be executed. Use of PL/SQL is possible by using prepareCall() API.	Required.

Table 10–16 Metric Columns Collected

Column	Description
Status	Status of the test. Status is 'down' if there is a SQLException generated by the fetchlet.
Total Time	Time required for the fetchlet to execute the test.
Connect Time	Time required for DriverManager.getConnection() to complete.
Prepare Time	Time required for conn.prepareStatement() to complete.
Execute Time	Time required for stmt.executeQuery() to complete.
Fetch Time	Time required for while(rs.next()) { rs.getRow() } to complete.
Close Time	Time required for closing resultset, statement, connection to complete.
Number of rows	Number of rows fetched.
Total time per row	
Fetch time per row	

10.10 OJMX/SOAP Fetchlet

The OJMX fetchlet communicates with the JMX agent on the managed J2EE server and performs the specified operations.

Input Parameters

Table 10–17 JMX Fetchlet Input Parameters

Name	Type	Description
metricType	String	Tells the fetchlet that this metric is of type Web Service (GWS).
requestBodyElement	String	Provides the name of the Web Service operation as used in the Web Service request body.
documentType	String	Specifies SOAP encoding. For example: rpc/encoded or doc/literal.
soapVersion	String	Version of SOAP that the fetchlet should use to communicate with the Web Service (default is 1.2)
targetNamespace	String	Target namespace of Web Service.
ColumnOrder	String []	Comma separated list of XPaths to pick the column for a resultant Web Service response.
rowData	String []	Comma separated list of XPath prefixes which, when appended to corresponding columnOrder Xpaths, provide the value of the metric column for each row when arrays are returned in the response from a Web Service.
URI	String	URI of Web Service.
soapAction	String	SOAP action for web service from WSDL.
returnType	String	Type of web service invocation return value.
arguments	XML	Values to pass to the Web Service invocation.

Output

The OJMX fetchlet returns an object of type MetricResult that contains the information retrieved.

11 Receivelets

Receivelet is a library that allows Enterprise Manager to receive external notifications sent by third-party network elements. These are notifications that are asynchronously sent and without any requests from Oracle Management Agent.

Usually, Oracle Management Agent data retrieval mechanism is based on a polling model, that is, modular libraries, called Fetchlets. Fetchlets collect values of various metrics from their managed targets on a regular basis. Oracle Management Agent then compares the gathered data with user-defined thresholds and generates events when the thresholds were met.

Receivelets are a more efficient way of dealing with these metrics. It depends on the ability of the managed target to detect the condition for its own events, and then communicate with Enterprise Manager only when an event occurs. When this communication happens, Oracle Management Agent uses receivelets to receive the information.

You can use Receivelets as a quicker way to get alerts on data that will be eventually collected via Fetchlets. You can also use Receivelets as a way to send both alerts and data, or just alerts for cases where there is no real data chart associated with the alert.

Receivelet is not a substitute for Fetchlet, but it is another way of collecting data. It is more for immediacy of notification compared to periodic polling that Fetchlet offers. Therefore, if you can fetch data, then use Fetchlets to get that data. However, if your server is capable of sending you events or data at a cost lower than that associated with Fetchlets, then use Receivelets.

A receivelet may be tightly coupled to a particular type of managed target, or may be useful to a broad range of potential targets.

The following receivelets are offered with Enterprise Manager:

- SNMP Receivelets
- Advanced Queue Receivelets
- HTTP Receivelets

11.1 SNMP Receivelets

SNMP Receivelets allow you to receive SNMP Traps notifications from third-party network elements, and translate them into a form compatible with Oracle Management Service.

While monitoring third-party entities in your managed environment, if the status of a third-party network element turns unavailable or if its metric severity conditions (metric thresholds) are met or exceeded, the SNMP Agent of that third-party network

element sends a notification to Oracle Management Agent. These notifications may be in the form of SNMP Traps that get triggered asynchronously and without any requests from Oracle Management Agent.

Since these traps are based on SNMP, Oracle Management Agent uses SNMP Receivelets to receive and translate these SNMP Traps into a form compatible with Oracle Management Service.

Once the SNMP Traps are received, the SNMP Receivelet extracts information pertaining to only those object identifiers (OIDs) that are defined in the *push descriptor* section of the metadata.xml configuration file (that is, only for those third-party network elements that need to be monitored by Enterprise Manager). For information about the locations of configuration file and Document Type Declaration (DTD) files, see Chapter 2: 2 Developing a Management Plug-In of this guide.

Whenever an SNMP Trap is sent by the SNMP agent, the SNMP Receivelet receives those traps based on the SNMP Agent configuration, translates them to an Enterprise Manager understandable format (like event or datapoint based on the *push descriptor* information), and stores that information (in XML files) in the upload directory. The Upload Manager checks for such new files in the upload directory, and then uploads those files onto Oracle Management Service. Enterprise Manager, then, accesses the Oracle Management Service to extract the collected information and display it to the user.

Configuration Required for Receiving SNMP Traps

To receive SNMP traps, you have to make some configuration settings at Oracle Management Agent side and at SNMP target agent side.

This will enable the SNMP targets to send SNMP traps to Oracle Management Agent's SNMP Receivelet. Once the SNMP traps is received, the SNMP Receivelet uses the *Push Descriptor* properties, such as MatchAgentAddr, MatchEnterprise, and so on, to identify the target and metric for which the traps belongs. Then the SNMP Receivelet uses the *Push Descriptor* properties, such as Event<metric-column> or Event<metric-column>OID, SeverityCode, and so on, to convert the traps into an event. Once this happens, the SNMP Receivelet uploads the converted event to the Management Repository and in turn is displayed in the console.

Configuration Settings Required at Oracle Management Agent Side

1. Define SNMP target in the `$ORACLE_HOME/sysman/emd/targets.xml` file.
2. Identify the metrics of the target that you want to monitor using the SNMP Receivelet (SNMP Traps).
3. Define the identified metrics in the target type metadata file ($ORACLE_HOME/sysman/admin/metadata/<target-type.xml) with the SNMP *Push Descriptor* as shown in the examples.
4. Identify the columns of the metric for which you want to generate EM events from the SNMP Target's trap.
5. Define the identified event columns as one of the *Push Descriptor* properties. For example, Event<metric-column> or Event<metric-column>OID.
6. After making these configuration settings, the emctl reload agent command reloads the modified configuration to Oracle Management Agent and SNMP Receivelet starts to listen on a UDP port.

   By default, the SNMP Receivelet listens over UDP on the same port as that of Oracle Management Agent. However, if you want to use a different listening port

for the SNMP Receivelet, then add the `SnmpRecvletListenNIC(=8002)` in the `emd.properties` file.

Configuration Settings Required at SNMP Target Agent Side

1. Add or modify entries in the 'manager' table of the SNMP target agent for manager IP/host name and manager port.

   Manager host name is the name of the computer where Oracle Management Agent is running. Manager port is the port on which SNMP Receivelet receives SNMP traps.

Input Parameters

Table 11–1 SNMP Receivelet Input Parameters

Parameter	Type	Description	Use
MatchEnterprise	String	OID used to define the trap being sent.	Required
MatchGenericTrap	String	Code for a generic SNMP trap.	Required
MatchSpecificTrap	String	Trap defined in a MIB (not one of the generic traps), the ID assigned in that MIB.	Required
MatchAgentAddr	String	IP address of the generating SNMP agent, as sent in the trap.	Required
Event<metric-column>	String	Specifies that, on receiving this trap, the recvlet should generate a severity on this metric column. The value of the metric-column should be the value of this parameter. (This case is useful where the expected values of the EM metric are not the same as the triggering SNMP variable.)	Required, if events have to be generated. However, if Event<metric-column>OID is provided, then this is not required.
Event<metric-column>OID	String	Specifies that, on receiving this trap, the recvlet should generate a severity on this metric column. The value of the metric column should be taken from the varbind in the trap with OID equal to the value of this parameter.	Required, if events have to be generated. However, if Event<metric-column> is provided, then this is not required.
SeverityCode	String	Specifies the level at which the severity should be generated. The value of this parameter should be one of 'CRITICAL', 'WARNING', or 'CLEAR'.	Required. However, if SeverityCodeOID is provided, then this is not required.
SeverityCodeOID	String	Specifies the level at which the severity should be generated. If the varbind in the trap with OID equal to the value of this parameter is one of the strings 'CRITICAL', 'WARNING', or 'CLEAR', the severity should be generated at that level; otherwise, no severity should be generated. (This parameter would only be used if the integrator were designing a trap exclusively for use with EM, but may be useful in this case.)	Required. However, if SeverityCode is provided, then this is not required.

Table 11–1 (Cont.) SNMP Receivelet Input Parameters

Parameter	Type	Description	Use
Data<metric-column>OID	String	Specifies that, on receiving this trap, the recvlet should generate a datapoint on the metric, for which the value of this metric column should be taken from the varbind in the trap with OID equal to the value of this parameter. (An SNMP Push Descriptor may have many Data* parameters, in which case a single row will be returned, with all specified columns populated from the appropriate varbind in the trap. An SNMP PushDescriptor may not have both a Data* parameter and a Severity* one, nor may it have multiple Severity* parameters.)	Required, if datapoints have to be generated.
Key<metric-column>OID	String	Severity or datapoint generated by this PushDescriptor should contain a key-value for this metric column. The key-value should be taken from the varbind in the trap with OID equal to the value of this parameter. For every key-column in the metric definition, there must be a Key* parameter in the PushDescriptor.	Optional
Context<metric-column>OID	String	If the PushDescriptor generates a severity, the severity should contain a value for this metric column in its event context. The value should be take from the varbind in the trap with OID equal to the value of this parameter. If the PushDescriptor generates a datapoint, this parameter is ignored.	Optional. This can be used only for datapoints.

Example

Example 11–1 shows how a trap from a vendor-specific router looks like.

Example 11–1 Trap from a Vendor-Specific Router

```
ascendLinkDown    TRAP-TYPE
   ENTERPRISE     ascend
   VARIABLES     { ifIndex, ifAdminStatus, ifOperStatus, ifType,
                   ifName }
   DESCRIPTION    "This trap is in addition to the generic linkDown
            trap defined in RFC1215.  This trap provides
            additional information such as ifAdminStatus,
            ifOerStatus, ifName, slotIfSlotIndex, slotIfItemIndex.
            This is an Alarm class trap and it can
            be enabled/disabled via alarmEnabled and/or
            ascendLinkDownTrapEnabled in trap profile.
            This trap is sent only if rfc1215 linkDown trap is generated."
   ::= 50
```

Example 11–2 show how the trap will be received by Oracle Management Agent. Note that <x> in this example is the value of *ifIndex* that identifies the particular interface that's having problems.

Example 11–2 Trap Received by Oracle Management Agent

```
Message:
    version: 0
    community: 'public'
```

```
    Trap-PDU:
        enterprise: enterprises.ascend (1.3.6.1.4.1.529)
        agent-addr: 138.2.204.10
        generic-trap: 6
        specific-trap: 50
        time-stamp: <timestamp from router's sysUptime>
        variable-bindings:
            Name: ifIndex.<x> (1.3.6.1.2.1.2.2.1.1.<x>)
            Type: INTEGER
            Value: <x>

            Name: ifAdminStatus.<x> (1.3.6.1.2.1.2.2.1.7.<x>)
            Type: INTEGER
            Value: up (1)

            Name: ifOperStatus.<x> (1.3.6.1.2.1.2.2.1.8.<x>)
            Type: INTEGER
            Value: down (2)

            Name: ifType.<x> (1.3.6.1.2.1.2.2.1.3.<x>)
            Type: INTEGER
            Value: iso88023-csmacd (7)

            Name: ifName.<x> (1.3.6.1.2.1.2.2.1.31.<x>)
            Type: DisplayString
            Value: 'eth0'
```

Example 11–3 shows how the metric can be defined in the metadata.xml file.

Example 11–3 Metric Defined in the metadata.xml File

```
<Metric NAME="interfaces" TYPE="TABLE">
    <TableDescriptor>
        <ColumnDescriptor NAME="name" TYPE="STRING" IS_KEY="TRUE"/>
        <ColumnDescriptor NAME="type" TYPE="NUMBER" IS_KEY="FALSE"/>
        <ColumnDescriptor NAME="status" TYPE="NUMBER" IS_KEY="FALSE"/>
        <ColumnDescriptor NAME="configured_status" TYPE="NUMBER" IS_KEY="FALSE"/>
    </TableDescriptor>
</Metric>
```

Example 11–4 shows how the push descriptor can be defined in the metadata.xml file to trigger a severity.

Example 11–4 Push Descriptor in the metadata.xml File For Triggering a Severity

```
<PushDescriptor RECVLET_ID="SNMPTrap">
   <Property NAME="MatchEnterprise" SCOPE="GLOBAL">1.3.6.1.4.1.529</Property>
   <Property NAME="MatchGenericTrap" SCOPE="GLOBAL">6</Property>
   <Property NAME="MatchSpecificTrap" SCOPE="GLOBAL">50</Property>
   <Property NAME="MatchAgentAddr" SCOPE="INSTANCE">AdminAddress</Property>
   <Property NAME="SeverityStatusOID"
SCOPE="GLOBAL">1.3.6.1.2.1.2.2.1.8</Property>
   <Property NAME="KeyNameOID" SCOPE="GLOBAL">1.3.6.1.2.1.2.2.1.31</Property>
   <Property NAME="ContextTypeOID" SCOPE="GLOBAL">1.3.6.1.2.1.2.2.1.3</Property>
   <Property NAME="ContextConfigured_statusOID"
SCOPE="GLOBAL">1.3.6.1.2.1.2.2.1.7</Property>
   <Property NAME="SeverityCode" SCOPE="GLOBAL">CRITICAL</Property>
</PushDescriptor>
```

Example 11–5 show how the push descriptor can be defined in the metadata.xml file to trigger a datapoint, which would specify the reporting of data on the same trap, with *ifName* as the key-column and the other three as data columns.

Example 11–5 Push Descriptor in the metadata.xml File For Triggering a Datapoint

```
<PushDescriptor RECVLET_ID="SNMPTrap">
   <Property NAME="MatchEnterprise" SCOPE="GLOBAL">1.3.6.1.4.1.529</Property>
   <Property NAME="MatchGenericTrap" SCOPE="GLOBAL">6</Property>
   <Property NAME="MatchSpecificTrap" SCOPE="GLOBAL">50</Property>
   <Property NAME="MatchAgentAddr" SCOPE="INSTANCE">AdminAddress</Property>
   <Property NAME="KeyNameOID" SCOPE="GLOBAL">1.3.6.1.2.1.2.2.1.31</Property>
   <Property NAME="DataStatusOID" SCOPE="GLOBAL">1.3.6.1.2.1.2.2.1.8</Property>
   <Property NAME="DataTypeOID" SCOPE="GLOBAL">1.3.6.1.2.1.2.2.1.3</Property>
   <Property NAME="DataConfigured_statusOID"
SCOPE="GLOBAL">1.3.6.1.2.1.2.2.1.7</Property>
</PushDescriptor>
```

Error Handling

There are no exceptions thrown from SNMP Receivelets.

Notes

- The metadata.xml file can have multiple push descriptor definitions.
- For event format, ensure that your push descriptor defines only one metric column. Similarly, for other metric columns, you can create separate, distinct push descriptor definitions.
- For datapoint format, ensure that your push descriptor defines only one metric column. However, you can use the *Context<column_name>OID* parameter to capture details of other metric columns, in the same push descriptor definition.

11.2 Advanced Queue Receivelets

Advanced Queue Receivelets allow you to receive notifications from Oracle Databases, and translate them into a form compatible with Oracle Management Service.

Oracle Databases allow you to monitor and apply thresholds to many of its own performance metrics internally; this Advanced Queue Receivelet is dedicated to receiving notifications of these metrics. Therefore, Advanced Queue Receivelets apply only to Oracle Database targets.

While monitoring Oracle Databases, every time one of its performance metrics exceeds its threshold, the database posts an alert to the Advanced Queue. When Oracle Management Agent connects to an Oracle Database, it registers itself as a subscriber to the Advanced Queue; thereafter, a copy of each threshold alert is preserved for Oracle Management Agent in the queue.

If Oracle Management Agent is running and connected to the database at the time the alert is enqueued, then the alert is immediately available and passed on to the repository tier. However, if Oracle Management Agent is not running, then the alert is preserved and made available to the receivelet only when they connect to the database next time.

For the Advanced Queue Receivelet to receive notifications, the oracle_database.xml file must be updated to include push descriptor definitions for different metrics. The oracle_database.xml file can be found at $EMDROOT/sysman/admin/metadata/. You can retain the query descriptors defined in this file to ensure that the SQL

Fetchlets continue to collect regular data, while the Advanced Queue Receivelets receive immediate notification of alerts as they are generated by Oracle Databases.

The push descriptors in the Advanced Queue Receivelet contain the properties such as QueueName (defined to the constant GLOBAL string 'ALERT_QUE'), MachineName, Port, SID, UserName, and password. The push descriptors may also define a KeyField property, the value of this property names the field in the Advanced Queue alert whose value should be used as the *keyvalue* parameter in any call to nmercm_RecvletManager_reportEvent().

Input Parameters

Table 11–2 AQ Receivelet Input Parameters

Parameter	Type	Description	Use
QueueName	String	Name of the advanced queue.	Required.
MachineName	String	Database host.	Required.
Port	Integer	Database port.	Required
SID	String	Database SID.	Required
UserName	String	User name.	Required
password	String	User password.	Optional; default is ""

Example

Example 11–6 shows how the push descriptor can be defined in the oracle_database.xml file for the *wait_bottlenecks* metric that does not have a key value.

Example 11–6 Push Descriptor in the oracle_database.xml File for a Metric without Key Value

```
<PushDescriptor RECVLET_ID="AQMetrics">
  <PushProperty NAME="QueueName" SCOPE="GLOBAL">ALERT_QUE</PushProperty>
  <PushProperty NAME="MachineName" SCOPE="INSTANCE">MachineName</PushProperty>
  <PushProperty NAME="Port" SCOPE="INSTANCE">Port</PushProperty>
  <PushProperty NAME="SID" SCOPE="INSTANCE">SID</PushProperty>
  <PushProperty NAME="UserName" SCOPE="INSTANCE">UserName</PushProperty>
  <PushProperty NAME="password" SCOPE="INSTANCE">password</PushProperty>
</PushDescriptor>
```

Example 11–7 shows how the push descriptor can be defined in the oracle_database.xml file for the *tbspFull* metric that has a key value. The key value is the name column. The value, which is the name of the tablespace an alert applies to, will be sent in the OBJECT_NAME field of the alert.

Example 11–7 Push Descriptor in the oracle_database.xml File for a Metric with Key Value

```
 <PushDescriptor RECVLET_ID="AQMetrics">
  <PushProperty NAME="QueueName" SCOPE="GLOBAL">ALERT_QUE</PushProperty>
  <PushProperty NAME="MachineName" SCOPE="INSTANCE">MachineName</PushProperty>
  <PushProperty NAME="Port" SCOPE="INSTANCE">Port</PushProperty>
  <PushProperty NAME="SID" SCOPE="INSTANCE">SID</PushProperty>
  <PushProperty NAME="UserName" SCOPE="INSTANCE">UserName</PushProperty>
  <PushProperty NAME="password" SCOPE="INSTANCE">password</PushProperty>
  <PushProperty Name="KeyField" SCOPE="GLOBAL">OBJECT_NAME</PushProperty>
</PushDescriptor>
```

Error Handling

There are no exceptions thrown from Advanced Queue Receivelets.

11.3 HTTP Receivelets

HTTP Receivelets function much like other receivelets, except that they are designed to receive notifications from targets that communicate over HTTP or HTTPS protocol.

For example, application servers running in your environment. These application server may have built-in mechanism to trigger notifications every time a threshold is reached or the status is down. HTTP Receivelets allow you to receive these notifications and store that information (in XML files) in the upload directory.

The following are the parameters used when data information is sent as a notification:

- TargetName
- TargetType
- MetricName
- CollTS

The following are the parameters used when alert information is sent as a notification:

- TargetName
- TargetType
- CollTS
- MetricName
- MetricColumn
- KeyValue
- Severity
- DataValue
- AlertMessage

Example 11–8 shows how a notification (without an access key) from an HTTP-based targets looks. These notifications do not need any translation, and the information passed is simply captured and stored as is in the XML files. Example 11–9 shows how a simple push descriptor can be defined in the metadata.xml file to receive this information.

However, you may choose to generate access keys that can be used for authenticating the metric information coming from an HTTP-based target. This is to ensure that notifications are parsed only if they are from authorized targets.

In general, communications to the Management Agent should be protected to avoid problems of spoofing and denial of service attacks. Understandably, one option is to use HTTPS protocol as the communication mechanism to encrypt the channel, and the other option is to keep the communication restricted to targets that are authorized by the Management Agent. This authorization mechanism involves generation of access keys.

Authorization or access keys can be generated on demand by the Management Agent and passed on to the HTTP-based targets so that they can use them while connecting to the HTTP Receivelet. The Management Agent stores these generated keys in its

memory and verifies if every notification has a valid key. If the key has already been generated for a request string, then that value is returned. Example 11–10 shows how a complex push descriptor can be defined in the metadata.xml file to generate access keys.

> **Note:** securing your communication is only an option. You can also receive notifications without authenticating the information.

Once the access keys are generated, they need to be sent to the HTTP-based targets so that they can use them while connecting to the HTTP Receivelet. This can be done using one of the following methods:

1. An optional *command* to run a script as part of the HTTP Receivelet push descriptor. Example 11–10 shows how a push descriptor can be used with this optional *command* parameter to run the script and send back the generated access keys.
2. Setting up a polled metric that will start up the process and provide it with the access key. Example 11–11 shows how a query descriptor can be used to poll the HTTP-based target at regular intervals and send back the generated access keys.

Input Parameters

Table 11–3 HTTP Receivelet Input Parameters

Parameter	Type	Description	Use
command	String	Name of the perl script used that will generate the access key for the metric information received from a particular server.	Optional. Used only when authentication using access keys is required.
STDIN	String	Controls the value that is passed on the stdin to the launched command. The value is used inside the launched command file for access key generation logic.	Optional
ENV	String	Controls the value that is passed via the environment. (Note that if security is not a concern, then integrators can also use a ENV prefix to request a value to be passed in the environment to the launched command). The value is used inside the launched command file for access key generation logic.	Optional

Example

Example 11–8 shows how a notification without an access key from an HTTP-based targets looks:

Example 11–8 Notification without an Access Key

```
 <!-- Connect -->
<HTTPRecvletConnect HOST="%%EMD_HOST" PORT="%%EMD_PORT" ACCESS_KEY_FILE="%%{T_
WORK}/tvmayh04.key">
<!-- get database metric and dump the metric data -->
<HTTPRecvletSendAlert TARGET_TYPE="tvmayh04" TARGET_NAME="tvmayh04" METRIC_
NAME="metric" METRIC_COLUMN="col2" SEVERITY="CLEAR" KEY_VALUE="1" VALUE="1"
```

```
MESSAGE="Severity from recvlet" TIMESTAMP="2000-01-01 00:00:00" />
<!-- Disconnect -->
</HTTPRecvletConnect>
```

Example 11–9 shows how a simple push descriptor in the metadata.xml file looks:

Example 11–9 Simple Push Descriptor in the metadata.xml File

```
<Metric NAME="metric" TYPE="TABLE">
   <TableDescriptor>
       <ColumnDescriptor NAME="col1" TYPE="NUMBER" IS_KEY="TRUE"/>
       <ColumnDescriptor NAME="col2" TYPE="NUMBER" IS_KEY="FALSE"/>
   </TableDescriptor>
   <PushDescriptor RECVLET_ID="HTTP">
   </PushDescriptor>
 </Metric>
```

Example 11–10 shows how a complex push descriptor in the metadata.xml file looks, and how the generated access keys can be sent back to the HTTP-based target:

Example 11–10 Complex Push Descriptor in the metadata.xml File

```
<PushDescriptor RECVLET_ID="HTTPRecvlet">
  <Property NAME="command" SCOPE="GLOBAL">%perlBin%/perl
%scriptsDir%/startHRClient.pl</Property>
  <Property NAME="STDINKey" SCOPE="SYSTEMGLOBAL">_hr_acc_key_%NAME%_%TYPE%_
TestMetric</Property>
  <Property NAME="STDINTName" SCOPE="INSTANCE">NAME</Property>
  <Property NAME="STDINTType" SCOPE="INSTANCE">TYPE</Property>
  <Property NAME="STDINMetric" SCOPE="GLOBAL">TestMetric</Property>
</PushDescriptor>
```

This results in the startHRClient.pl script being run and the following values being sent to stdin.

```
Key=<AccessKey>
TName=<TargetName>
TType=<TargetType>
Metric=TestMetric
```

The STDIN method of passing values helps to keep sensitive data out of the command line or in the process environment.

Example 11–11 Polling Method to Send Back the Generated Access Keys

```
<QueryDescriptor FETCHLET_ID="OSLineToken">
<Property NAME="command" SCOPE="GLOBAL">%perlBin%/perl
%scriptsDir%/checkHRClient.pl</Property>
<Property NAME="STDINKey" SCOPE="SYSTEMGLOBAL">_hr_acc_key_%NAME%_%TYPE%_
TestMetric</Property>
<Property NAME="ENVTName" SCOPE="INSTANCE">NAME</Property>
<Property NAME="ENVTType" SCOPE="INSTANCE">TYPE</Property>
<Property NAME="ENVMetric" SCOPE="GLOBAL">TestMetric</Property>
</QueryDescriptor>
```

Error Handling

There are no exceptions thrown from HTTP Receivelets.

12

Enterprise Manager DTD

A DTD provides the grammar for the XML files, thus describing what content is expected in each of its related XML files. When creating a new XML file, you need to carefully study its DTD to understand what content needs to be present in that file.

This chapter to provides a lookup of DTD elements to facilitate integration with Oracle Server Technology products.

12.1 Terminology

This chapter provides a lookup of DTD elements to facilitate integration with Oracle Server Technology products. Most of the examples in the document are snippets of an XML file.

Target: Target is a managed entity. A managed entity can be a hardware device or a software resource. Examples of a target are: host system, Oracle database, SMTP, or service.

Associated Target: Targets whose data depend on each other.

Metric: Collectable data.

Mid Tier: OMS - Oracle Management Server

Container: A container is a home that houses an Oracle installation. Currently 2 kinds of containers are possible - Oracle Home (database, Enterprise Manager, Application Server installs) and ApplTop (application installs).

```
Cluster Targets: Cluster targets span across many hosts. All cluster targets
represent the same target. The agents monitoring each of the cluster targets will
produce the same metric result, severities etc.

Cluster Interfaces: Standard interfaces that the agent uses to talk to the cluster
target.
```

12.2 Target Metadata DTD Elements

This section defines DTD elements used by Enterprise Manager.

12.2.1 TargetMetadata

The TargetMetadata describes the metadata for a target type. Metadata for a target describes its measurable characteristics, format of the collected data and the mechanism to collect or compute that data.

```
<!ATTLIST TargetMetadata
META_VER CDATA #REQUIRED
```

```
TYPE CDATA #REQUIRED
REQUIRED_AGENT_VERSION CDATA #IMPLIED
HELPID CDATA #IMPLIED
HELP CDATA #IMPLIED
CATEGORY_PROPERTIES CDATA #IMPLIED
RESOURCE_BUNDLE_PACKAGE CDATA #IMPLIED
```

Note: The maximum length allowed for the various attributes mentioned in the file must be mentioend in tagsize.properties. This will be used to truncate the length of the attributes in the metadata file while loading the metadata information into the repository.

12.2.1.1 Attributes

META_VER: Describes the version of metadata.

TYPE: Specifies the Target type.

HELPID: Not used.

CATEGORY_PROPERTIES: Semicolon separated list of properties, used as properties for ValidIf. Currently, each target type can have up to 5 properties used as category property. EMAgent evaluates the values of the category properties and makes it available within the metadata.

RESOURCE_BUNDLE_PACKAGE:

Attributes introduced in **Enterprise Manager Version 10.2**:

REQUIRED_AGENT_VERSION: This attribute indicates the minimum agent version for the metadata. TargetMetadata marked with this attribute will be valid on Agent versions greater than or equal to the specified version.

HELP: Not used.

12.2.1.2 Element

Display

TypeProperties (First introduced in Enterprise Manager version 10.2.)

AssocTarget

DiscoveryHelper (First introduced in Enterprise Manager version 10.2.)

MonitoringMode

AltSkipCondition

MetricClass (First introduced in Enterprise Manager version 10.2.)

Metric

CredentialInfo

InstanceProperties

12.2.1.3 Used In

`TargetMetadata` is a top-level element.

12.2.1.4 Examples

```
<TargetMetadata TYPE="example1" META_VER="2.0" REQ
<TargetMetadata TYPE="example1" META_VER="2.0" REQ
```

```
UIRED_AGENT_VERSION="10.2.0.1.0">
. . .
</TargetMetadata>
```

The Metadata in the above example has REQUIRED_AGENT_VERSION attribute set to "10.2.0.1.0". This metadata will be valid only on agent versions 10.2.0.1.0 and higher.

```
<Metric NAME="prop" TYPE="TABLE">
<TableDescriptor>
<ColumnDescriptor NAME="name" TYPE="STRING" IS_KEY="TRUE" /> <ColumnDescriptor
NAME="value" TYPE="STRING" />
</TableDescriptor>
<Property NAME="hostname" SCOPE="INSTANCE">NAME</Property>
</QueryDescriptor>
</Metric>
</TargetMetadata>
<QueryDescriptor FETCHLET_ID="OS">
```

This is a very simple example that describes a Target type, 'example1' having data () that needs to be collected in the following format. The quoted values are evaluated by the 'OS' Fetchlet in accordance with the scoping rules defined in.

Table 12–1 Metric Prop

Name	Value
Host Name	NAME

```
<TargetMetadata TYPE="example2" META_VER="2.0">
      <Metric NAME="perf" TYPE=”TABLE”>
<TableDescriptor>
<ColumnDescriptor NAME="char" TYPE="STRING" IS_KEY="TRUE" />
<ColumnDescriptor NAME="value" TYPE="STRING" />
      </TableDescriptor>
      <QueryDescriptor FETCHLET_ID="OSLineToken" >
           <Property NAME="command" SCOPE="GLOBAL">%perlBin%/perl
%scriptsDir%/example1/perf.pl %port% </Property>
           <Property NAME="delimiter" SCOPE="GLOBAL">=</Property>
           <Property NAME="port" SCOPE="INSTANCE">accessPort</Property>
      </QueryDescriptor>
   </Metric>
   <InstanceProperties>
      <InstanceProperty NAME="accessPort" />
   </InstanceProperties>
</TargetMetadata>
```

This sample illustrates the use of `InstanceProperties`. The `InstanceProperties` element associates ‘accessPort’ to be associated with the target instance. The metric ‘perf’ is collected as shown below. The quoted values are evaluated by the ‘OSLineToken’ Fetchlet in accordance with the scoping rules defined in Property.

Table 12–2 Metric: perf

Char	Value
Command	‘%perlBin%/perl %scriptsDir%/example1/perf.pl %port%’
Delimiter	=
Port	’accessPort’

```
<TargetMetadata TYPE="example3" META_VER="2.0" CATEGORY_PROPERTIES="OS">
  <Metric NAME=”prop” TYPE=”TABLE”>
   . . .
   </Metric>
<InstanceProperties>
<DynamicProperties NAME="VersionAndLocation" FORMAT="ROW" PROP_
LIST="OS;OracleHome;Version">
   <QueryDescriptor FETCHLET_ID="OSLineToken">
<Property NAME="scriptsDir" SCOPE="SYSTEMGLOBAL">scriptsDir</Property>
<Property NAME="perlBin" SCOPE="SYSTEMGLOBAL">perlBin</Property>
<Property NAME="ENVEmdOS" SCOPE="SYSTEMGLOBAL">_emdOS</Property>
<Property NAME="ENVVersion" SCOPE="SYSTEMGLOBAL">_emdVersion</Property>
<Property NAME="ENVORACLE_HOME" SCOPE="SYSTEMGLOBAL">emdRoot</Property>
<Property NAME="command" SCOPE="GLOBAL">%perlBin%/perl</Property>
<Property NAME="script" SCOPE="GLOBAL">%scriptsDir%/emdlocandver.pl </Property>
<Property NAME="startsWith" SCOPE="GLOBAL">em_result=</Property>
<Property NAME="delimiter" SCOPE="GLOBAL">|</Property>
   </QueryDescriptor>
</DynamicProperties>
   </InstanceProperties>
</TargetMetadata>
```

This sample uses a DynamicProperties element to return OS, OracleHome and Version properties. The scripts return results that are parsed to return the properties listed in the PROP_LIST attribute.

12.2.2 Display

Specifies the information to be used by the Grid Control Console for displaying the element that has this tag.

```
<!ELEMENT Display (ValidIf*, Label, ShortName?, Icon?, Description?, Unit?)>
<!ATTLIST Display
FOR_SUMMARY_UI (TRUE | FALSE) "FALSE">
```

12.2.2.1 Attributes:

FOR_SUMMARY_UI: Indicates whether a column is visible in the condensed UI. Condensed view is necessary when all the columns cannot fit in one UI page. In a condensed view, only columns whose FOR_SUMMARY_UI=TRUE will be displayed.

TRUE | FALSE (default)

12.2.2.2 Elements:

ValidIf

Label

ShortName

Icon

Description

Unit

12.2.2.3 Used In:

Target Metadata DTD Elements

Metric

ColumnDescriptor

CredentialType

CredentialTypeColumn

CredentialSet

CredentialSetColumn

InstanceProperty

12.2.2.4 Examples:

Display element must contain a Label element. The elements ShortName, Icon, Description and Unit are optional. If ValidIf element(s) are present then all the conditions in the ValidIf elements must be satisfied for the element to be displayed.

```
<Display FOR_SUMMARY_UI="TRUE">
<Label NLSID="emd_resp_stat">Status</Label>
</Display>
```

This example describes the display characteristics for the element that includes it. The 'FOR_SUMMARY_UI' set to TRUE forces the UI to display the element in the condensed view also.

```
<Display FOR_SUMMARY_UI="TRUE">
<ValidIf>
<CategoryProp NAME="OS" CHOICES="SunOS"/>
</ValidIf>
<Label NLSID="id_name_for_agent">Agent Name</Label>
<ShortName NLSID="id_short_name">Name</ShortName>
<Icon GIF="a.gif">Name</Icon>
<Description NLSID="id_for_description">Displays the Agent Name</Description>
</Display>
```

This sample describes the display characteristics for an element that are valid only for 'SunOS' OS.

12.2.3 TypeProperties

The TypeProperties holds TypeProperty elements for the target type. This element is introduced in **Enterprise Manager version 10.2**.

```
<!ELEMENT TypeProperties (TypeProperty*)>
```

12.2.3.1 Attributes:

None

12.2.3.2 Elements:

TypeProperty

12.2.3.3 Used In:

TargetMetadata

12.2.3.4 Examples:

```
<TypeProperties>
<TypeProperty PROPERTY_NAME="a_name" PROPERTY_VALUE="a_value"/>
</TypeProperties>
```

The property name-value pair, *a_name,a_value*, would apply to the target type.

12.2.4 TypeProperty

TypeProperty element contains the property name-value pair for a target type. This element is introduced in Enterprise Manager **Version 10.2**.

```
<!ELEMENT TypeProperty EMPTY>
<!ATTLIST TypeProperty
PROPERTY_NAME CDATA #REQUIRED
PROPERTY_VALUE CDATA #IMPLIED
>
```

12.2.4.1 Attributes:

PROPERTY_NAME: Name of the target type property.

PROPERTY_VALUE: Value of the target type property.

12.2.4.2 Elements:

None

12.2.4.3 Used In:

TypeProperties

12.2.4.4 Examples:

Please refer to the example for TypeProperties

12.2.5 AssocTarget

The AssocTarget describes how two targets are related to each other. Targets may be associated for a number of reasons some of them being: rendering topology maps, root cause analysis, determining availability of targets, minimizing redundancy in data collection and transmission and determining order for data collection or job execution amongst others. For instance, if a fault occurs on a target, its associated target might be affected too. This information would be valuable in root cause analysis.

```
<!ELEMENT AssocTarget (AssocPropDef*)>
<!ATTLIST AssocTarget
ASSOC_TARGET CDATA #IMPLIED
TYPE CDATA #IMPLIED
ASSOCIATION_NAME CDATA #IMPLIED
NAME_NLSID CDATA #IMPLIED
DESCRIPTION CDATA #IMPLIED
DESCRIPTION_NLSID CDATA #IMPLIED
SOURCE_TARGET_TYPE CDATA #IMPLIED
ASSOC_TARGET_TYPE CDATA #IMPLIED
CARDINALITY (OPTIONAL_SINGLE_CARDINAL |
REQUIRED_SINGLE_CARDINAL |
OPTIONAL_MULTI_CARDINAL |
REQUIRED_MULTI_CARDINAL) #IMPLIED
ASSOC_TYPE (RELATES_TO|DEPENDS_ON | CONNECTS_TO |
SERVICE_ACCESS_POINT|RUNS_ON|
CONTAINS|HOSTED_BY|MONITORED_BY|
OPTIONALLY_CONNECTS_TO) #IMPLIED>
```

12.2.5.1 Attributes:

ASSOC_TARGET: The name of the associated target.

TYPE: The target type of the associated target.

Attributes introduced in Enterprise Manager **version 10.2**:

ASSOCIATION_NAME: Deprecates ASSOC_TARGET. Specifies the name of the association.

ASSOC_TARGET_TYPE: Deprecates TYPE. Specifies the target type that is associated with this target. 'ANY' can be used to indicate that the association target could be *any* target type.

NAME_NLSID: NLSID for association name.

DESCRIPTION: Description

DESCRIPTION_NLSID: NLSID for description string.

SOURCE_TARGET_TYPE: If association starts from a target, other than the target itself, the target type of the source target is specified in this attribute. 'ANY' can be used to indicate that the source could be *any* target type.

CARDINALITY: Specifies the cardinality of the associated targets.

Supported values are:

a) OPTIONAL_SINGLE_CARDINAL: Zero or one targets as associated.

b) REQUIRED_SINGLE_CARDINAL: Exactly one associated target.

c) OPTIONAL_MULTI_CARDINAL: Zero or several associated targets.

d) REQUIRED_MULTI_CARDINAL: One or more associated targets.

ASSOC_TYPE: Describes the relation of the associated targets.

Supported values are:

a) RELATES_TO (default): Implies "some" generic relationship.

b) DEPENDS_ON: Dependency on associated target.

c) CONNECTS_TO: Source target connects to associated target.

d) SERVICE_ACCESS_POINT

e) RUNS_ON: Source target runs on (installed on) associated target.

f) CONTAINS: Source target contains associated target.

g) HOSTED_BY: Similar to runs on.

h) MONITORED_BY: Source target is monitored by associated target (Agent).

i) OPTIONALLY_CONNECTS_TO

12.2.5.2 Elements:

AssocPropDef (Not supported in Enterprise Manager version 10.2.)

12.2.5.3 Used In:

TargetMetadata

12.2.5.4 Examples:

AssocTarget element for versions prior to 10.2 MUST have the following attributes:

a) ASSOC_TARGET

b) TYPE

AssocTarget element for versions 10.2 and later MUST use the following attributes at least.

a) ASSOCIATION_NAME instead of ASSOC_TARGET

b) ASSOC_TARGET_TYPE instead of TYPE.

```
<TargetMetadata TYPE="oracle_email" META_VER="2.0">
<AssocTarget ASSOCIATION_NAME="IM"
SOURCE_TARGET_TYPE="oracle_email"
ASSOC_TARGET_TYPE="oracle_im"
ASSOCIATION_TYPE="DEPENDS_ON"
NAME_NLSID="im_assoc_name"
DESCRIPTION="This association captures Email-IM dependency"
DESCRIPTION_NLSID="im_assoc_description" />
. . .
</TargetMetadata>
```

This element would be defined in 'oracle_email' and would represent the following relation:

Oracle_email -----DependsOn----à oracle_im

12.2.6 AssocPropDef

The AssocPropDef describe the properties for an association. This element is not supported in Enterprise Manager **version 10.2**.

```
<!ELEMENT AssocPropDef EMPTY>
<!ATTLIST AssocPropDef
NAME CDATA #REQUIRED
REQUIRED (TRUE | FALSE) #REQUIRED>
```

12.2.6.1 Attributes:

NAME: Name of the property.

REQUIRED: Indicates whether the property is required.

TRUE | FALSE

12.2.6.2 Elements:

None

12.2.6.3 Used In:

AssocTarget

12.2.6.4 Examples:

This element is not supported.

12.2.7 DiscoveryHelper

The DiscoveryHelper helps the agent in its process of discovering the target type. This element is introduced in Enterprise Manager version 10.2.

```
<!ELEMENT DiscoveryHelper (DiscoveryHint*) >
<!ATTLIST DiscoveryHelper
CATEGORYNAME CDATA #REQUIRED
OUI_BASED (TRUE | FALSE) "TRUE"
>
```

12.2.7.1 Attributes:

CATEGORYNAME: name of the category in discover.lst which discovers targets for a given type.

OUI_BASED: boolean value to indicate if this discovery used OUI inventory info.

TRUE (default) | FALSE

12.2.7.2 Elements:

DiscoveryHint

12.2.7.3 Used In:

TargetMetadata

12.2.7.4 Examples:

None.

12.2.8 DiscoveryHint

The DiscoveryHint allows users to specify any hint which can be a guide to discovery. This element is introduced in **Enterprise Manager version 10.2**.

```
<!ELEMENT DiscoveryHint (Display?) >
<!ATTLIST DiscoveryHint
NAME CDATA #REQUIRED
>
```

12.2.8.1 Attributes:

NAME: name of the hint to guide discovery process

12.2.8.2 Elements:

Display

12.2.8.3 Used In:

DiscoveryHelper

12.2.8.4 Examples:

None.

12.2.9 MetricClass

MetricClass provides a means for classifying Metrics into categories. Metrics can be classified into categories based on multiple characteristics such as Function (Perf, Load, Config), EvaluationCost (Cheap, Medium, Expensive) and Applicability (Typical, Esoteric). This element was first introduced in **Enterprise Manager version 10.2**.

```
<!ELEMENT MetricClass (MetricCategory*)>
<!ATTLIST MetricClass
NAME CDATA #REQUIRED
NLSID CDATA #IMPLIED>
```

12.2.9.1 Attributes:

NAME: Is the name of the class (e.g. Functional)

NLSID: Is the translation ID. The naming convention is metric_class_<classname>

12.2.9.2 Elements:

MetricCategory

12.2.9.3 Used In:

TargetMetadata

12.2.9.4 Examples:

```
<TargetMetadata TYPE="example3" META_VER="2.0">
<MetricClass NAME="EvaluationCost" NLSID="id_for_eval_cost_class">
<MetricCategory NAME="CHEAP" NLSID="id_for_cheap_cat"/>
<MetricCategory NAME="MEDIUM" NLSID="id_for_medium_cat"/>
<MetricCategory NAME="EXPENSIVE" NLSID="id_for_expensive_cat"/>
</MetricClass>
<Metric NAME="metric1" TYPE="TABLE">
<CategoryValue Class="EvaluationCost" CATEGORY_NAME="CHEAP"/>
. . .
</Metric>
. . .
</TargetMetadata>
```

The example describes adding a 'EvaluationCost' MetricClass for a Target type 'example3'. EvaluationCost has 3 categories: CHEAP, MEDIUM, and EXPENSIVE. Metric, 'metric1' is a CHEAP metric to evaluate.

Please refer to the explanations of CategoryValue, Metric for more details.

12.2.10 MetricCategory

A MetricCategory element lists each choice within a classification of metrics. This element was first introduced in **Enterprise Manager version 10.2**.

```
<!ELEMENT MetricCategory EMPTY>
<!ATTLIST MetricCategory
NAME CDATA #REQUIRED
NLSID CDATA #IMPLIED>
```

12.2.10.1 Attributes:

NAME: The name of the category (e.g. Security)

NLSID: The NLSID of the category. The naming convention here is metric_cat_ <category_name>

12.2.10.2 Elements:

None

12.2.10.3 Used In:

MetricClass

12.2.10.4 Examples:

Please refer to the example for MetricClass

12.2.11 Metric

A metric element is used to declare the different measurable characteristics (performance, load, configuration etc.) of a target. The metric element describes the structure of the collected data as well as how to compute the information.

> **Note:** Oracle EM recommends that every target type have a special metric named "Response". This metric should have a column called "Status". The type creator should also set up a collection of this metric and set up a condition (see TargetCollection.dtd) on the Status column. The availability system (target up/down status over time) uses alerts on this metric column to provide up/down statistics over time.

QueryDescriptor, ExecutionDescriptor, PushDescriptor is optional only for REPOSITORY_TABLE, REPOSITORY_STRING, REPOSITORY_NUMBER and REPOSITORY_EVENT metric types. For all other metric types they are required.

```
<!ELEMENT Metric ((ValidIf | ValidMidTierVersions)*, Display?, CategoryValue*
,TableDescriptor?, ((QueryDescriptor | ExecutionDescriptor) | PushDescriptor)* )>
<!ATTLIST Metric
NAME CDATA #REQUIRED
TYPE (NUMBER | STRING | TABLE | RAW | EXTERNAL | REPOSITORY_TABLE | REPOSITORY_
NUMBER | REPOSITORY_STRING | REPOSITORY_EVENT) "NUMBER"
REPOS_PLSQL CDATA #IMPLIED
USAGE_TYPE (VIEW_COLLECT | REALTIME_ONLY | HIDDEN | HIDDEN_COLLECT | COLLECT_
UPLOAD) "VIEW_COLLECT"
KEYS_FROM_MULT_COLLS (TRUE | FALSE) "FALSE"
IS_TEST_METRIC (TRUE | FALSE) "FALSE"
KEYS_ONLY (TRUE | FALSE) "FALSE"
REMOTE (TRUE | FALSE) "FALSE"
IS_TRANSPOSED (TRUE | FALSE) "FALSE"
HELP CDATA #IMPLIED
IS_METRIC_LONG_RUNNING (TRUE|FALSE) "FALSE"
CONFIG (TRUE|FALSE) "FALSE"
FORCE_CACHE (TRUE | FALSE) "FALSE">
```

12.2.11.1 Attributes:

NAME: Specifies Metric name, it uniquely identifies it within the scope of its target type.

TYPE: Specifies the data type.

Supported metric types are

a) NUMBER (default): Deprecated - Instead please use a table with 1 column of type NUMBER.

b) STRING: Deprecated - Instead please use a table with 1 column of type STRING

c) TABLE: Tabular data

d) RAW: Tabular data

c) EXTERNAL: Data not parsed/fronted by EMD.

e) REPOSITORY_TABLE: computed elsewhere

f) REPOSITORY_NUMBER: computed elsewhere

g) REPOSITORY_STRING: computed elsewhere

h) REPOSITORY_EVENT: First introduced only in **Enterprise Manager version 10.2**.

REPOS_PLSQL: If type is REPOSITORY_TABLE, REPOSITORY_NUMBER or REPOSITORY_STRING, this attribute specifies the name of the PL/SQL proc to execute at the repository to evaluate the metric.

USAGE_TYPE: This defines the purpose of the metric.

Supported types are:

a) VIEW_COLLECT (default): These are metrics that are both viewable and collected.

b) REALTIME_ONLY: These are metrics that cannot be collected. The rules on key uniqueness are not applied to these metrics.

c) HIDDEN: Metrics are tagged hidden when they shouldn't be collected nor visible from the console. The data is not uploaded either. These are "temporary" metrics used to compute other metrics.

Values introduced in **Enterprise Manager version 10.2**:

d) HIDDEN_COLLECT: Metric can be collected. It will not be viewable. The data is not uploaded. It is similar to HIDDEN, but can have collection on it.

e) COLLECT_UPLOAD: Metric can be collected and uploaded, Metadata is uploaded to MGMT_METRICS but it is not viewable in 'All' metric page.

Mapping of old USAGE_TYPE values

DISPLAY_ONLY: REALTIME_ONLY

MULTI_KEY: VIEW_COLLECT

COLLECT_ONLY: VIEW_COLLECT

The metric browser automatically decides which metrics to not display.

KEYS_FROM_MULT_COLLS: If TRUE, the attribute indicates that there are multiple key columns. The combination of key columns uniquely identifies a row. If the value is TRUE only then can the metric be collected in multiple collection items.

TRUE | FALSE (default)

IS_TEST_METRIC: The agent can check some metrics to determine if a target has been correctly specified with valid instance properties. This attribute marks this metric as one of the test metrics.

TRUE or FALSE (default)

HELP: Help text - This attribute is not used.

KEYS_ONLY: It is used to tag special metrics that have only key columns. Note that in general, such metrics are not useful in collections (since no data is uploaded), but there may be special cases where the metric is used to just retrieve a set of keys.

TRUE or FALSE (default)

IS_METRIC_LONG_RUNNING: IF true, the metric is long running. This gives the metric engine, a hint that this query will take relatively longer to finish. A special property EM_IS_METRIC_LONG_RUNNING will be passed to fetchlet automatically.

TRUE or FALSE (default)

CONFIG: This is a special designation for CONFIG metrics that are uploaded differently by the EM framework.

TRUE or FALSE (default)

Attributes introduced in **Enterprise Manager version 10.2:**

REMOTE: It is used to tag metrics that can be evaluated from a remote location. These metrics could be evaluated from "beacon" nodes

IS_TRANSPOSED: It is used to tag metrics that generate data as name value pairs and the UI treats the names as "column headers". These are useful when the number of rows (or data categories) is not known at design time.

FORCE_CACHE: For collected metrics, this is a strong hint to the agent to cache the results of a metric collection. In the absence of this hint, the agent may only start caching the result of a metric after it realizes that someone will try to use the cached value.

12.2.11.2 Elements:

ValidIf

ValidMidTierVersions (First introduced in Enterprise Manager version 10.2.)Display

CategoryValue(First introduced in Enterprise Manager version 10.2.)

TableDescriptor

QueryDescriptor

ExecutionDescriptor

PushDescriptor

12.2.11.3 Used In:

TargetMetadata

12.2.11.4 Examples:

```
<TargetMetadata TYPE="example1" META_VER="2.0" CATEGORY_PROPERTIES="OS;Version">
<Metric NAME="prop" TYPE="TABLE">
<Display>
<Label NLSID="example1_metric">Example1 Metric</Label>
</Display>
<TableDescriptor>
<ColumnDescriptor NAME="name" TYPE="STRING" IS_KEY="TRUE" /> <ColumnDescriptor
NAME="value" TYPE="STRING" />
</TableDescriptor>
<QueryDescriptor FETCHLET_ID="OS">
<Property NAME="hostname" SCOPE="INSTANCE">NAME</Property>
</QueryDescriptor>
</Metric>
</TargetMetadata>
```

This is the most common form of a metric definition. The statement declares the metric, 'example1', to contain tabular data.

If a metric has a "TABLE" type, the value will be returned as a set of rows each containing a set of values (columns). A list will be a special case of the Table. A Table Metric must have a TableDescriptor defined.

```
<Metric NAME="Inventory" TYPE="EXTERNAL" >
<ValidIf>
<CategoryProp NAME="OS" CHOICES="SunOS"/>
</ValidIf>
```

```
<Display>
<Label NLSID="host_Inventory">Inventory</Label>
</Display>
<QueryDescriptor FETCHLET_ID="OS">
<Property NAME="emdRoot" SCOPE="SYSTEMGLOBAL">emdRoot</Property>
<Property NAME="emHome" SCOPE="SYSTEMGLOBAL">agentStateDir</Property>
<Property NAME="scriptsDir" SCOPE="SYSTEMGLOBAL">scriptsDir</Property>
<Property NAME="perlBin" SCOPE="SYSTEMGLOBAL">perlBin</Property>
<Property NAME="hostConfigClasspath"
SCOPE="SYSTEMGLOBAL">hostConfigClasspath</Property>
<Property NAME="hostname" SCOPE="INSTANCE">NAME</Property>
<Property NAME="type" SCOPE="INSTANCE">TYPE</Property>
<Property NAME="display_target_name" SCOPE="INSTANCE">DISPLAY_NAME</Property>
<Property NAME="display_target_type" SCOPE="INSTANCE">TYPE_DISPLAY_NAME</Property>
<Property NAME="command" SCOPE="GLOBAL">"%perlBin%/perl"
"%scriptsDir%/osm/ecmCollectInventory.pl" "%hostConfigClasspath%" "%perlBin%"
"%emdRoot%" "%emHome%" "%hostname%" "%loaderFile%"
"%emHome%/sysman/config/OUIinventories.add" "%type%" "%display_target_name%"
"%display_target_type%"</Property>
</QueryDescriptor>
</Metric>
```

This example declares 'Inventory' as EXTERNAL which implies that the metric will be evaluated in the correct format and will be placed in the upload directory. The EMAgent will not parse the metric result. The ValidIf element ensures that the metric will be evaluated only for 'SunOS' OS.

```
<Metric NAME="AddressMap" TYPE="TABLE" FORCE_CACHE="TRUE">
```

The contents would be similar to a TABLE metric

```
</Metric>
```

In this example, the agent is forced to cache the results for the AddressMap metric.

```
<Metric NAME="ICMPPing" TYPE="TABLE" IS_TEST_METRIC="TRUE" USAGE_TYPE="HIDDEN">
```

The contents would be similar to a TABLE metric

```
</Metric>
```

'ICMPPing' is identified as a test metric. The agent will use its value to verify that the target identified by the instance properties is correct. Since the purpose of this metric is for INTERNAL use only, it is marked as 'HIDDEN'.

USAGE_TYPE Summary:

USAGE_TYPE	KEY_ UNIQUE_ CHECK	COLLECTABLE	MGMT_ METRICS_ RAW	MGMT_ METRICS	VIEWABLE
VIEW_COLLECT	Y	Y	Y	Y	Y
REALTIME_ONLY	N	N	N	Y	N
HIDDEN	Y	N	N	Y	N
HIDDEN_ COLLECT	Y	Y	N	Y	N
COLLECT_ UPLOAD	Y	Y	Y	Y	Y

```
<Metric NAME="http_raw" TYPE="TABLE"KEYS_FROM_MULT_COLLS="TRUE" REMOTE="TRUE">
```

The contents would be similar to a TABLE metric

```
</Metric>
```

'http_raw' metric can be evaluated from a remote location and therefore is tagged as 'REMOTE'.

```
<Metric NAME="openPorts" TYPE="RAW" CONFIG="TRUE" KEYS_ONLY="TRUE" HELP="NO_HELP">
<ValidIf>
<CategoryProp NAME="OS" CHOICES="SunOS"/>
</ValidIf>
<Display>
<Label NLSID="host_open_ports_ESM">Open Ports</Label>
</Display>
<TableDescriptor TABLE_NAME="esm_collection">
<ColumnDescriptor NAME="property" COLUMN_NAME="property" TYPE="STRING" IS_
KEY="TRUE" HELP="NO_HELP"/>
<ColumnDescriptor NAME="value" COLUMN_NAME="value" TYPE="STRING" IS_KEY="TRUE"
HELP="NO_HELP"/>
</TableDescriptor>
<QueryDescriptor FETCHLET_ID="OSLineToken">
<Property NAME="scriptsDir" SCOPE="SYSTEMGLOBAL">scriptsDir</Property>
<Property NAME="perlBin" SCOPE="SYSTEMGLOBAL">perlBin</Property>
<Property NAME="command" SCOPE="GLOBAL">%perlBin%/perl</Property>
<Property NAME="script" SCOPE="GLOBAL">%scriptsDir%/openports.pl</Property>
<Property NAME="startsWith" SCOPE="GLOBAL">em_result=</Property>
<Property NAME="delimiter" SCOPE="GLOBAL">=</Property>
</QueryDescriptor>
</Metric>
```

'openPorts' is defined as a CONFIG metric. Since the type is RAW, EMAgent expects the values in the correct format.

```
<Metric NAME="storage_reporting_data" TYPE="RAW" CONFIG="TRUE" IS_METRIC_LONG_
RUNNING="TRUE">
```

The contents would be similar to a RAW metric

```
</Metric>
```

This specifies that storage_reporting_data takes a long time to execute.

12.2.12 ValidIf

The ValidIf element is used to create type definitions that apply to multiple flavors of a target. To do this, certain properties of the target (up to a max of 5) can be marked as category properties, and ValidIf elements can be placed in portions of the metadata to indicate that they are only applicable if the target's property values match the specified values. The CategoryProp elements within a ValidIf should all match for the containing element to be evaluated. A containing element may include multiple ValidIfs to indicate its applicability for different sets of conditions.

```
<!ELEMENT ValidIf (CategoryProp+)>
```

12.2.12.1 Attributes:

None

12.2.12.2 Elements:

CategoryProp

12.2.12.3 Used In:

Metric

QueryDescriptor

Display

MonitoringMode

InstanceProperty

DynamicProperties

ExecutionDescriptor

PushDescriptor

Collection Item

12.2.12.4 Examples:

```
<TargetMetadata TYPE="example1" META_VER="2.0" CATEGORY_PROPERTIES="OS;Version">
<Metric NAME="prop" TYPE="TABLE">
<ValidIf>
<CategoryProp NAME="OS" CHOICES="SunOS"/>
<CategoryProp NAME="Version" CHOICES="5.9"/>
</ValidIf>
<TableDescriptor>
<ColumnDescriptor NAME="name" TYPE="STRING" IS_KEY="TRUE" /> <ColumnDescriptor
NAME="value" TYPE="STRING" />
</TableDescriptor>
<QueryDescriptor FETCHLET_ID="OS">
<Property NAME="hostname" SCOPE="INSTANCE">NAME</Property>
</QueryDescriptor>
</Metric>
</TargetMetadata>
```

This example indicates that the category properties, 'OS' and 'Version' must have values, 'SunOS' and '5.9' for the metric, 'prop' to be evaluated. EMAgent allows the definition of upto 5 category properties which can be used in ValidIfs elements.

12.2.13 CategoryProp

The CategoryProp element is used to list the allowed values for a property for the ValidIf to match.

```
<!ELEMENT CategoryProp EMPTY>
<!ATTLIST CategoryProp
NAME CDATA #REQUIRED
CHOICES CDATA #REQUIRED>
```

12.2.13.1 Attributes:

NAME: Specifies the name of the category property.

CHOICES: Specifies the values the property can have. This may contain values separated by ";".

12.2.13.2 Elements:

None

12.2.13.3 Used In:

ValidIf

12.2.13.4 Examples:

```
<ValidIf>
<CategoryProp NAME="OS" CHOICES="SunOS"/>
<CategoryProp NAME="Version" CHOICES="5.8;5.9"/>
```

```
</ValidIf>
```

If the Example described in the ValidIf statement is modified such that the 'Version' CategoryProperty now has two choices 5.8 and 5.9, the metric would be evaluated for 'SunOS' versions 5.8 and 5.9.

```
<ValidIf>
<CategoryProp NAME="OS" CHOICES="SunOS"/>
</ValidIf>
<ValidIf>
<CategoryProp NAME="OS" CHOICES="AIX"/>
</ValidIf>
```

A metric that has ValidIfs as shown above, would be evaluated if OS is either SunOS or AIX.

> **Note:** If ValidIfs are used to differentiate between multiple definitions of the same metric, there should be no cases where multiple definitions of the metric get validated.

```
<Metric NAME="a_metric" TYPE="TABLE">
<ValidIf>
<CategoryProp NAME="C1" CHOICES="A;B">
</ValidIf>
. . .
</Metric>
<Metric NAME="a_metric" TYPE="TABLE">
<ValidIf>
<CategoryProp NAME="C1" CHOICES="A;C">
</ValidIf>
. . .
</Metric>
```

The above example demonstrates how NOT to use ValidIfs.

12.2.14 ValidMidTierVersions

This element was first introduced in **Enterprise Manager version 10.2**. The ValidMidTierVersions element is used in the mid tier based versioning support in the agent, introduced in Enterprise Manager version 10.2.

This element can be used either under a Metric or within a CustomTableMapper element. When present, it indicates to the agent that a Metric definition or a CustomTableMapper definition only applies for a certain set of mid-tier versions.

```
<!ELEMENT ValidMidTierVersions EMPTY>
<!ATTLIST ValidMidTierVersions
PLUG_IN CDATA #IMPLIED
START_VER CDATA #IMPLIED
END_VER CDATA #IMPLIED>
```

12.2.14.1 Attributes:

PLUG_IN: Optional attribute that allows a particular mid tier plug in to be referenced. If not specified, this tag applies to the core OMS version.

START_VER: Starting version (inclusive, optional) the element is applied from.

END_VER: Ending version (exclusion, optional) that the element is applicable to.

12.2.14.2 Elements:

None

12.2.14.3 Used In:

Metric

CustomTableMapper

12.2.14.4 Examples:

Every ValidMidTierVersions element needs to have at least one of START_VER or END_VER specified.

```
<Metric NAME="metric1" TYPE="RAW">
<TableDescriptor>
<CustomTableMapper REP_TABLE_NAME="table1">
<ValidMidTierVersions PLUG_IN="DB" START_VER="10.1" END_VER="10.3"/>
<ColumnMapper METRIC_COLUMN="col1" REP_TABLE_COLUMN="col1_rep"/>
</CustomTableMapper>
<ColumnDescriptor NAME="c1" COLUMN_NAME="col1" TYPE="STRING"/> <ColumnDescriptor
NAME="c2" COLUMN_NAME="col2" TYPE="STRING"/>
</TableDescriptor>
<QueryDescriptor>
. . .
</QueryDescriptor>
</Metric>
```

The CustomTableMapper element mapping 'metric1' to repository table, 'table1' is applicable only for DB Plugin versions between 10.1 (inclusive) and 10.3 (not inclusive).

<ValidMidTierVersionsSTART_VER="10.1.0.1"/>

This element if present in a Metric or CustomTableMapper would indicate to the EMAgent that the Metric or the CustomTableMapper is applicable only to OMS versions 10.1.0.1 and higher.

<ValidMidTierVersionsEND_VER="10.2"/>

This element if present in a Metric or CustomTableMapper would indicate to the EMAgent that the Metric or the CustomTableMapper is applicable only to OMS versions less than (not including) 10.2.

12.2.15 TableDescriptor

TableDescriptor describes the structure of the data for the metric of type TABLE.

```
<!ELEMENT TableDescriptor (ColumnDescriptor+, CustomTableMapper*)>
<!ATTLIST TableDescriptor
TABLE_NAME CDATA #IMPLIED
SKIP_TARGET_COLUMN (TRUE | FALSE) "FALSE"
SKIP_METRIC_COLUMN (TRUE | FALSE) "FALSE"
SKIP_COLLTIME_PK (TRUE | FALSE) "FALSE"
SKIP_COLLTIME_COLUMN (TRUE | FALSE) "FALSE">
```

Note: SKIP_COLLTIME_PK attribute is deprecated. This attribute specifies that the collection timestamp should not be a part of the primary key. This was used to indicate that the latest row should override any previous rows with the same primary key. Since this can now be done by simply altering the table definition in the repository, SKIP_COLLTIME_PK is not useful any more.

12.2.15.1 Attributes:

TABLE_NAME: This attribute specifies the repository database table into which the collected data will be loaded to. *Note: Only RAW metrics can define this attribute.If a TableDescriptor contains CustomTableMapper elements, it should not contain a TABLE_NAME attribute.*

SKIP_TARGET_COLUMN: This attribute is applicable for a RAW metric only. If set to TRUE, Target GUID column will not be generated.

TRUE | FALSE (default)

SKIP_METRIC_COLUMN: This attribute is applicable for a RAW metric only. If set to TRUE, the Metric Name column will not be generated.

TRUE | FALSE (default)

SKIP_COLLTIME_PK: Deprecated in **Enterprise Manager version 10.2**.The SKIP_COLLTIME_PK option can be used if the collection timestamp needs to be generated, but not added as a primary key.

TRUE | FALSE (default)

12.2.15.2 Elements:

ColumnDescriptor

CustomTableMapper

12.2.15.3 Used In:

Metric

12.2.15.4 Examples:

A Table Metric must have a TableDescriptor defined. It describes columns of the table.

```
<Metric NAME="prop" TYPE="TABLE">
<Display>
<Label NLSID="example1_metric">Example1 Metric</Label>
</Display>
<TableDescriptor>
<ColumnDescriptor NAME="name" TYPE="STRING" IS_KEY="TRUE" /> <ColumnDescriptor
NAME="value" TYPE="STRING" />
</TableDescriptor>
<QueryDescriptor FETCHLET_ID="OS">
<Property NAME="hostname" SCOPE="INSTANCE">NAME</Property>
</QueryDescriptor>
</Metric>
```

This is the most common usage for the TableDescriptor element that represents a metric of TYPE=TABLE. For mid-tier versioning support please refer to the example for CustomTableMapper.

```
<TableDescriptor TABLE_NAME="esm_collection">
```

This declaration is valid only for a RAW metric. This element contains the description of esm_collection table within a RAW metric.

```
<TableDescriptor TABLE_NAME="mgmt_db_compatibility"
SKIP_COLLTIME_PK="TRUE" SKIP_COLLTIME_COLUMN="TRUE"
SKIP_METRIC_COLUMN="TRUE" SKIP_TARGET_COLUMN="TRUE">
```

The SKIP attributes may be applied only to RAW metrics. The EMAgent automatically generates TARGET_GUID, METRIC_NAME and COLLECTION_TIMESTAMP columns for a raw metric unless explicitly indicated by setting SKIP attributes to TRUE. The TableDescriptor, in this sample explicitly requests the omission of the default columns.

12.2.16 ColumnDescriptor

The ColumnDescriptor elements describe each column in a table. The agent also supports one level of nesting of tables for RAQ metrics. This allows metrics to return a table of data in place of a column which is uploaded in the context of the containing row. For example: In a metric that returns a list of expensive SQL statements in a database, a column that returns the multi-row explain plan for the SQL statement could be returned in a nested raw metric column.

```
<!ELEMENT ColumnDescriptor (Display?, CategoryValue*, TableDescriptor?)>
<!ATTLIST ColumnDescriptor
NAME CDATA #REQUIRED
TYPE (NUMBER | STRING | RAW | CLOB | BLOB) "NUMBER"
IS_FILENAME (TRUE | FALSE) "FALSE"
IS_KEY (TRUE | FALSE) "FALSE"
TRANSIENT (TRUE | FALSE) "FALSE"
COMPUTE_EXPR CDATA #IMPLIED
COLUMN_NAME CDATA #IMPLIED
IS_LONG_TEXT (TRUE | FALSE) "FALSE"
IS_DATE (TRUE | FALSE) "FALSE"
STATELESS_ALERTS (TRUE|FALSE) "FALSE"
IS_TIMESTAMP (TRUE | FALSE) "FALSE"
NON_THRESHOLDED_ALERTS (TRUE | FALSE) "FALSE"
KEYONLY_THRESHOLDS (TRUE | FALSE) "FALSE"
RENDERABLE (TRUE | FALSE) "TRUE"
HELP CDATA #IMPLIED>
```

12.2.16.1 Attributes:

NAME: This is the metric column name.

TYPE: Specifies the data type.

Supported types are:

a) NUMBER (default)

b) STRING.

c) RAW: This is for nested table support. A column may be defined as RAW to indicate it is a nested-table. *Note: Only 1 level of nested table is allowed.*

Values introduced in **Enterprise Manager version 10.2.**

d) CLOB: CLOB holds large character data such as a log file.

e) BLOB: BLOB holds binary data (.zip files, .tar, files, etc.).

IS_KEY: is set to true if this column is the primary key (uniquely identifies the row in the returned rows). For any two rows returned, the value of the key column cannot be

the same, else there will be a primary key violation. Note: Upto 5 columns can be marked with IS_KEY=TRUE. If no column is defined as key, the default value for the key is null (therefore should only return 1 row at a time)

TRUE | FALSE (default)

TRANSIENT: This will not be uploaded to repository. Only used to calculate rate data.

TRUE | FALSE (default)

COMPUTE_EXPR: This attribute specifies the formula for calculating the value of the column. Columns previously defined in the Table descriptor can participate in the calculation. Attaching a '_' prefix to a column name denotes previous value of a column. Support for string expressions is introduced in **Enterprise Manager version 10.2**. Please refer to the example for details about the expression grammar and usage.

Predefined special values:

a) __interval: collect interval.

b) __sysdate: current system time.

c) __GMTdate: current GMT time.

d) __contains: tests a given string expression for presence of a string expression.

e) __beginswith: tests whether a given string expression begins with a specified string expression.

f) __endswith: testw whether a given string expression ends with the specified string expression.

g) __matches: tests whether a given string expression matches a specified string expression.

h) __delta: computes the difference between the current value and the previous value.

i) __leadingchars: returns the leading characters in the specified string.

j) __trailingchars: returns the trailing characters in the specified string.

k) __substringpos: returns the position of the occurrence of the pattern within a specified string.

l) __is_null: tests whether the expression is NULL

m) __length: returns the length of the string expression.

n) __to_upper: converts the string to upper case.

o) __to_lower: converst the string to lower_case.

p) __ceil: returns the smallest integral value not less than identifier.

q) __floor: returns the largest integral value not greater than the identifier.

r) __round: rounds to nearest integer, away from zero.

COLUMN_NAME: This value will be used if the metric type is RAW to identify the database column.

IS_LONG_TEXT: This value will only be used when the metric is RAW and the column will be in digested form. The agent has support for metrics that expect to return the same long string repeatedly in metric results. If a column is marked with IS_LONG_TEXT="TRUE", the agent sends a row mapping the string to a digest into the MGMT_LONG_TEXT table and thereafter only sends the digested value as the data to the repository.

TRUE | FALSE (default)

IS_DATE: This value will only be used when the metric is RAW and the column is date type.

TRUE | FALSE (default)

STATELESS_ALERTS: This attribute if set to TRUE indicates to EM that alerts on this column will not have corresponding clears. This allows the UI to decide whether to allow users to manually clear alerts on this column.

TRUE | FALSE (default)

IS_TIMESTAMP: The value in this column will be used as the collection time for this row. If set to true, the values for this column should be specified in the yyyy-MM-dd HH:mm:ss z format (For example: "2003-07-30 08:41:05 PST"). The list of valid time zones is listed in the$ORACLE_HOME/sysman/emd/supportedtzs.lst file.

HELP: Not used.

Attributes introduced in **Enterprise Manager version 10.2**.

IS_FILENAME: When set to TRUE, it indicates that the column value is a file name that contains the real content that needs to be sent. IS_FILENAME attribute is valid only for CLOB/BLOB column types.

NON_THRESHOLDED_ALERTS: This attribute is used to indicate that there might be alerts for the metric column without there being a thresholded condition for it (eg: through server generated alerts).

TRUE | FALSE (default)

KEYONLY_THRESHOLDS: If this attribute is set to TRUE, conditions cannot apply to all metric rows and all Condition elements for the column need a KeyColumn element.

TRUE | FALSE (default)

RENDERABLE: A FALSE value for this attribute indicates that the value for this column maybe generated by the engine and may be cryptic or random enough to be of any use to the user. The UI would not display this value and would not allow the user to set thresholds for this value.

TRUE (default) | FALSE

12.2.16.2 Elements:

Display

CategoryValue (First introduced in Enterprise Manager version 10.2.)

TableDescriptor

12.2.16.3 Used In:

TableDescriptor

12.2.16.4 Examples:

Each column must specify the name and the data type for the column. The column can also be tagged as a key column. These column values qualify the value returned in the non-key columns.

For example: In a metric for top 10 processes, the process name will be the key column while the residence memory size, cpu, time used will be the value columns.

```
<ColumnDescriptor NAME="ciscoMemoryPoolName" TYPE="STRING" IS_KEY="TRUE">
```

```
<Display>
<Label NLSID="cisco_mem_pool_name">Memory Pool Name</Label>
</Display>
</ColumnDescriptor>
```

This is the most common usage of the ColumnDescriptor element. 'ciscoMemoryPoolName' is a key column in a metric. The values of this column are of type, 'STRING'. The optional Display element when included in the ColumnDescriptor, associates a UI Label with the Column.

```
<ColumnDescriptor NAME="pgScan" TYPE="NUMBER" IS_KEY="FALSE" TRANSIENT="TRUE"
HELP="NO_HELP"/>
```

The attribute, 'TRANSIENT' indicates that the column is used for internal calculations only and should not be uploaded to the repository.

```
<ColumnDescriptor NAME="property" COLUMN_NAME="property" TYPE="STRING" IS_
KEY="TRUE" HELP="NO_HELP"/>
```

The 'COLUMN_NAME' attribute will be used in RAW metrics to identify a database column.

```
<ColumnDescriptor NAME="log_file_message" TYPE="STRING" IS_KEY="FALSE" IS_LONG_
TEXT="TRUE"/>
```

'IS_LONG_TEXT' attribute, when set to TRUE, is an indication to the EMAgent to expect long values.

```
<ColumnDescriptor NAME="load_timestamp" COLUMN_NAME="load_timestamp" TYPE="STRING"
IS_DATE="TRUE" IS_KEY="FALSE"/>
```

This example defines a column 'load_timestamp' for a RAW metric with values in the date format.

```
<ColumnDescriptor NAME="log_file_match_count" TYPE="NUMBER" IS_KEY="FALSE"
STATELESS_ALERTS="TRUE"/>
```

This definition indicates that the alerts on the column, 'log_file_match_count' will not have corresponding clears. The UI can provide the users the option to manually clear alerts for this data.

The example for TableDescriptor describes a table metric with ColumnDescriptor elements.

Note: It is invalid for a ColumnDescriptor to both be a key and a timestamp column.

CLOB/BLOB is only valid inside RAW metrics. When TYPE is set to CLOB or BLOB, the ColumnDescriptor can also have IS_FILENAME attribute set to TRUE, in which case, the column value is the name of the file whose content should be sent rather than the column value itself. For CLOB/BLOB columns, the destination columns in the repository table should also be of CLOB/BLOB type.

Compute Expression support:

Supported Grammar:

```
expression := (cond_expr | (cond_expr ? cond_expr : cond_expr)
cond_expr := (string_expr |
(string_expr == string_expr) |
(string_expr < string_expr) |
(string_expr > string_expr) |
(string_expr <= string_expr) |
(string_expr >= string_expr) |
```

```
(string_expr __contains string_expr) |
(string_expr __beginswith string_expr) |
(string_expr __endswith string_expr) |
(string_expr __matches string_expr) |
(string_expr __delta string_expr))
string_expr := (simple_expr |
(simple_expr __leadingchars simple_expr) |
(simple_expr __trailingchars simple_expr) |
(simple_expr __substringpos simple_expr))
simple_expr := (term |
(simple_expr + term ) |
(simple_expr - term) )
term := (unary_expr |
(term * unary_expr ) |
(term / unary_expr ) )
unary_expr := (factor |
(__is_null factor) |
(__length factor) |
(__to_upper factor) |
(__to_lower factor) |
(__ceil factor) |
(__floor factor) |
(__round factor) )
factor := ( identifier |
string_literal |
number |
'(' expression ')' )
string_literal := '\'' (character | "\\'" )* '\''
```

Usage:

```
<ColumnDescriptor NAME="pgScan" TYPE="NUMBER" />
<ColumnDescriptor NAME="pgScanRate" TYPE="NUMBER" IS_KEY="FALSE" COMPUTE_
EXPR="(pgScan-_pgScan)/__interval"/>
```

The value of the column is calculated using the given compute expression. The value of the column is calculated using the present value of the 'pgScan' column, the previous value of the same column ('_pgScan') and the collect interval. Note: 'pgScan' column should be defined before any column can use its value in the COMPUTE_ EXPR.

```
<ColumnDescriptor NAME="baseDir" TYPE="STRING" />
<ColumnDescriptor NAME="component" TYPE="STRING" COMPUTE_EXPR="'/httpd'" />
<ColumnDescriptor NAME="full_path" TYPE="STRING" COMPUTE_EXPR="baseDir + component
+ '/egs/log/' "/>
```

The value of the column, 'full_path' is "<baseDir>/httpd/egs/log/" where <baseDir> is the value of the column baseDir.

```
<ColumnDescriptor NAME="value1" TYPE="NUMBER" COMPUTE_EXPR="Col __contains 'ay'"
/>
<ColumnDescriptor NAME="value2" TYPE="NUMBER" COMPUTE_EXPR="Col __beginswith
'Mon'" />
<ColumnDescriptor NAME="value3" TYPE="NUMBER" COMPUTE_EXPR="Col_ _endswith 'day'"
/>
<ColumnDescriptor NAME="value4" TYPE="NUMBER" COMPUTE_EXPR="Col __matches 'Sun*'"
/>
```

If the value of Col, in the above metric, is "Sunday", the outcome of the *COMPUTE_ EXPR* will be as follows:

value1 = 1

value2 = 0

value3 = 1

value4 = 1

```
<TableDescriptor>
<ColumnDescriptor NAME="parse_str" TYPE="STRING" IS_KEY="TRUE" />
<ColumnDescriptor NAME="startpos" TYPE="NUMBER" COMPUTE_EXPR="parse_str __
substringpos '#D1'" />
<ColumnDescriptor NAME="num_trailing" TYPE="NUMBER" COMPUTE_EXPR="(__length parse_
str) - startpos" />
<ColumnDescriptor NAME="trim_str" TYPE="STRING" COMPUTE_EXPR="parse_str__
trailingchars num_trailing" />
<ColumnDescriptor NAME="endpos" TYPE="NUMBER" COMPUTE_EXPR="trim_str __
substringpos '#E1'" />
<ColumnDescriptor NAME="result_str" TYPE="STRING" COMPUTE_EXPR="trim_str __
leadingchars endpos" />
</TableDescriptor>
```

The sample TableDescriptor describes a simple method for extracting a substring from a a given string. If the data represented by *"parse_str"* is of the form:

#A1 10

#B1 20

#C1 30

#D1 40

#E1 50

#F1 60

The *"result_str"* has the value "#D1=40".

12.2.17 CategoryValue

A CategoryValue element indicates the category for a metric, column or condition under a particular classification. If a CategoryValue is defined for a Metric element, it is valid for all the ColumnDescriptors in that Metric. If it is defined for a ColumnDescriptor, that column will have a category value that overrides the union of what it has and what is defined for the Metric.

The following MetricCategories are predefined for the MetricClass, 'FUNCTIONAL':

a) FAULT: metrics that can be used to indicate a breakdown in a component or occurrence of an error that indicates some component or user is unable to successfully complete processing. Example: AlertLog - Archiver hung

b) WORKLOAD_VOLUME: metrics that capture the workload on a system induced in proportion to the user's or batch jobs running against the system. It usually is an indication of how much work is done. Example: User calls (per second)

c) WORKLOAD_TYPE: metrics that capture the type of workload on a system independent of demand. It usually is an indication of what kind of work is done. Example: Logical Reads (per transaction)

d) PERFORMANCE: metrics that can be classified to measure the performance of a system. It usually is an indication of how well the system is doing. Example: Database response (per second)

e) CAPACITY: metrics that measure the usage of a fixed resource. Example: CPU Usage (per second)

f) CONFIGURATION: metrics that check the configuration of a target against a recommended *best-practice* configuration.

g) SECURITY: metrics that relate to the security aspects of the system.

```
<!ELEMENT CategoryValue EMPTY>
<!ATTLIST CategoryValue
CLASS CDATA #REQUIRED
CATEGORY_NAME CDATA #REQUIRED>
```

12.2.17.1 Attributes:

CLASS: Name of the metric class.

CATEGORY_NAME: Name of the metric category.

12.2.17.2 Elements:

Metric

ColumnDescriptor

Condition

12.2.17.3 Used In:

None

12.2.17.4 Examples:

```
<TargetMetadata TYPE="example3" META_VER="2.0">
<MetricClass NAME="EvaluationCost" NLSID="id_for_eval_cost_class">
<MetricCategory NAME="CHEAP" NLSID="id_for_cheap_cat"/>
<MetricCategory NAME="MEDIUM" NLSID="id_for_medium_cat"/>
<MetricCategory NAME="EXPENSIVE" NLSID="id_for_expensive_cat"/>
</MetricClass>
<Metric NAME="metric1" TYPE="TABLE">
<CategoryValue Class="EvaluationCost" CATEGORY_NAME="CHEAP"/>
. . .
</Metric>
. . .
</TargetMetadata>
```

This example illustrates the use of CategoryValue for a Metric. "metric1" is "CHEAP" to evaluate.

```
<Metric NAME="FileSystems" TYPE="TABLE" >
<CategoryValue CLASS="FUNCTIONAL" CATEGORY="WORKLOAD_VOLUME" />
<TableDescriptor>
<ColumnDescriptor NAME="FileSystem" TYPE="STRING" IS_KEY="TRUE" />
<ColumnDescriptor NAME="totalSpace" TYPE="NUMBER" >
<CategoryValue CLASS="FUNCTIONAL" CATEGORY="CAPACITY" />
</ColumnDescriptor>
<ColumnDescriptor NAME="diskUsedPct" TYPE="NUMBER /> <TableDescriptor> ....
</Metric>
```

In this sample, the column, "totalSpace" has a CategoryValue, "CAPACITY" which overrides the CategoryValue, "WORKLOAD_VOLUME" associated with the metric.

12.2.18 CustomTableMapper

The CustomTableMapper element was first introduced in **Enterprise Manager version 10.2** and is part of the mid tier based versioning project that allows custom (RAW) metrics to change their destination tables based on the version of the mid tier.

The TableDescriptor for a RAW metric can have multiple CustomTableMapper elements - one per set of mid tier versions - with each CustomTableMapper providing repository table and column mappings for the TableDescriptor's Columns.

```
<!ELEMENT CustomTableMapper (ValidMidTierVersions*, ColumnMapper*)>
<!ATTLIST CustomTableMapper
REP_TABLE_NAME CDATA #REQUIRED>
```

12.2.18.1 Attributes:

REP_TABLE_NAME: Indicates the table name that the content of the metric should be uploaded to.

12.2.18.2 Elements:

ValidMidTierVersions

ColumnMapper

12.2.18.3 Used In:

TableDescriptor

CustomTableMapper

12.2.18.4 Examples:

```
<TableDescriptor>
<ColumnDescriptor NAME="c1" COLUMN_NAME="col1" TYPE="STRING"/> <ColumnDescriptor
NAME="c2" COLUMN_NAME="col2" TYPE="STRING"/>
<CustomTableMapper REP_TABLE_NAME="table1">
<ValidMidTierVersions PLUG_IN="DB" START_VER="10.1" END_VER="10.3"/>
<ColumnMapper METRIC_COLUMN="col1" REP_TABLE_COLUMN="col1_rep"/>
</CustomTableMapper>
</TableDescriptor>
```

This sample illustrates Mid-tier based versioning. TableDescriptor element must not contain the TABLE_NAME attribute. The sample describes a mapping of the metric to the repository table, 'table1' for Mid-tier versions between 10.1 (inclusive) and 10.3 (not inclusive). Metric column, 'col1' maps to 'col1_rep' in the repository table, 'table1'.

Please refer to the example for ValidMidTierVersions also.

12.2.19 ColumnMapper

The ColumnMapper element was first introduced in **Enterprise Manager version 10.2** and is part of a CustomTableMapper element and describes the mapping between the ColumnDescriptor and the repository column its data should end up in.

The presence of a ColumnMapper provides the mapping for the column in a particular repository table, and indicates that the column is required in the table. To indicate that a column should not be uploaded to a particular version of the repository, there should not be a ColumnMapper for that column

```
<!ELEMENT ColumnMapper EMPTY>
<!ATTLIST ColumnMapper
METRIC_COLUMN CDATA #REQUIRED
REP_TABLE_COLUMN CDATA #REQUIRED>
```

12.2.19.1 Attributes:

METRIC_COLUMN: Name of the ColumnDescriptor this applies to

REP_TABLE_COLUMN: The database table column name this data should end up in.

12.2.19.2 Elements:

None

12.2.19.3 Used In:

CustomTableMapper

12.2.19.4 Examples:

Please refer to the example for CustomTableMapper.

12.2.20 QueryDescriptor

The query descriptor allows the framework to find the fetchlet as well as pass on the query information for obtaining performance data values from the target. The fetchlet can be identified by a well known id that is known to the EMAgent. It may also contain properties that will be passed to the fetchlet.

```
<!ELEMENT QueryDescriptor (ValidIf*, Property*) >
<!ATTLIST QueryDescriptor
FETCHLET_ID CDATA #REQUIRED
NEED_CHARSET_CONVERT (TRUE | FALSE) "TRUE">
```

12.2.20.1 Attributes:

FETCHLET_ID: Specifies the ID of the fetchlet to use that is known to the EMAgent. This attribute must point to an element from the $ORACLE_HOME/lib/fetchlets.reg file.

NEED_CHARSET_CONVERT: If the metric result is in correct "UTF8" encoding, this flag should be set to "FALSE" so that EMAgent will not do any character conversion.

TRUE (default) | FALSE

12.2.20.2 Elements:

ValidIf

Property

12.2.20.3 Used In:

Metric

DynamicProperties

12.2.20.4 Examples:

The query descriptor associated will contain the metadata that can be used to collect the value of the metric. For example: SQL Query.

```
<TargetMetadata TYPE="example1" META_VER="2.0">
<Metric NAME="prop" TYPE="TABLE">
<TableDescriptor>
<ColumnDescriptor NAME="name" TYPE="STRING" IS_KEY="TRUE" /> <ColumnDescriptor
NAME="value" TYPE="STRING" />
</TableDescriptor>
<QueryDescriptor FETCHLET_ID="OS">
<Property NAME="hostname" SCOPE="INSTANCE">NAME</Property>
</QueryDescriptor>
```

```
</Metric>
</TargetMetadata>
```

This simple example describes a query descriptor that is used in a Metric and relies on the 'OS' fetchlet to return the property 'hostname'.

```
<TargetMetadata TYPE="example1" META_VER="2.0">
<Metric NAME="prop" TYPE="TABLE">
. . .
</Metric>
<InstanceProperties>
<DynamicProperties NAME="VersionAndLocation" FORMAT="ROW" PROP_
LIST="OS;OracleHome;Version">
<QueryDescriptor FETCHLET_ID="OSLineToken">
<Property NAME="ENVEmdOS" SCOPE="SYSTEMGLOBAL">_emdOS</Property>
<Property NAME="ENVVersion" SCOPE="SYSTEMGLOBAL">_emdVersion</Property>
<Property NAME="ENVORACLE_HOME" SCOPE="SYSTEMGLOBAL">emdRoot</Property>
</QueryDescriptor>
</DynamicProperties>
</InstanceProperties>
</TargetMetadata>
```

This example illustrates the use of a QueryDescriptor to evaluate a DynamicProperties element.

12.2.21 Property

Describes the information to be passed to the fetchlets.

```
<!ELEMENT Property (#PCDATA)>
<!ATTLIST Property
NAME CDATA #REQUIRED
SCOPE (GLOBAL | INSTANCE | USER | SYSTEMGLOBAL | ENV | HOST | CACHE) "GLOBAL"
OPTIONAL (TRUE | FALSE) "FALSE">
```

Property values are resolved as follows:

1. The value is looked up in the specified scope

2. For each potential instantiation (ie %<varname>%) in the looked up value, varname is looked up as follows:

a. In the property values itself (ie in one of the earlier properties).

b. In the instance properties

c. In the systemglobal scope (emd.properties)

d. The value is checked for automatic property.

The following Automatic Properties are defined for a target:

1. NAME - substitutes Target Name

2. TYPE - substitutes Target type

3. DISPLAY_NAME - substitutes display name for the target

4. TYPE_DISPLAY_NAME - substitutes display name for the type

5. GUID - substitutes the guid

Note: All lookups are case-sensitive.

12.2.21.1 Attributes:

NAME: Name of the property.

SCOPE: Defines how the value of the property is to be resolved.

Supported values for Scope are:

a) GLOBAL (default): The property needs to be resolved in the Target Type Definition XML file.

b) INSTANCE: The property will be resolved by discovery. The PCDATA in that case should be the NAME of the property set in the discovery XML file.

c) USER: The property will be resolved by the caller(collector or the interactive end-user). The PCDATA in that case should be the name of the Property to be used when prompting the caller (in the case of interactive user).

d) SYSTEMGLOBAL: Use emd.properties to resolve the property.

e) ENV: Use environment variable to resolve the property.

f) HOST: The property must be resolved as an instance variable of the 'host' target on that EMAgent. For example, the OS property

g) CACHE: The value must be obtained from the previous evaluation of the metric. Any column returned in the previous evaluation can be specified, and this is only applicable to single row OR non-key metrics.

OPTIONAL: is meant to call out those properties that need NOT be available when provided to the fetchlet. The EMAgent will validate that it can find valid values for all non-optional properties before calling through to a fetchlet.

TRUE | FALSE (default)

12.2.21.2 Elements:

Contains character data representing the value for the property.

12.2.21.3 Used In:

QueryDescriptor

PushDescriptor

12.2.21.4 Examples:

```
<Property NAME="perlBin" SCOPE="SYSTEMGLOBAL">perlBin</Property>
```

'perlBin' Property has 'SYSTEMGLOBAL' scope which implies that emd.properties file is used to resolve the property

```
<Property NAME="delimiter" SCOPE="GLOBAL">|</Property>
```

'delimiter' property has 'GLOBAL' scope.

```
<Property NAME="hostname" SCOPE="INSTANCE">NAME</Property>
```

'hostname' has 'INSTANCE' scope which implies that the value will be resolved by discovery. The value 'NAME' must match the field in the discovery XML file.

```
<Property NAME="SNAPSHOT_TYPE" SCOPE="USER">SNAPSHOT_TYPE</Property>
```

This property will be resolved by the caller. 'SNAPSHOT_TYPE' is the name of the property when prompting the caller.

```
<Property NAME="ENVNMUPM_TIMEOUT" OPTIONAL="TRUE" SCOPE="SYSTEMGLOBAL">NMUPM_
TIMEOUT</Property>
```

The property, 'ENVNMUPM_TIMEOUT' is identified as an OPTIONAL property. All properties that are not OPTIONAL have to be validated by the EMAgent before the fetchlet is called.

Use of the property elements is illustrated in the following examples:

QueryDescriptor, ValidIf, TargetMetadata.

12.2.22 Label

This represents the Label that will be displayed in UI.

```
<!ELEMENT Label (#PCDATA)>
<!ATTLIST Label
NLSID CDATA #REQUIRED>
```

12.2.22.1 Attributes:

NLSID: Will contain the ID used to lookup the string in to the resource bundle

12.2.22.2 Elements:

Character data.

12.2.22.3 Used In:

Display

12.2.22.4 Examples:

This element must be present in the Display element.

```
<Label NLSID="host_load_cpuLoad">Run Queue Length (5 minute average)</Label>
```

This element is defined in a Display element and represents the label to display.

Please refer to the example for Display.

12.2.23 ShortName

ShortName is the short representation of the Metric Display name it should be less than 12 characters in length.

```
<!ELEMENT ShortName (#PCDATA)>
<!ATTLIST ShortName
NLSID CDATA #REQUIRED>
```

12.2.23.1 Attributes:

NLSID: Will contain the ID used to lookup the string in to the resource bundle

12.2.23.2 Elements:

Character data.

12.2.23.3 Used In:

Display

12.2.23.4 Examples:

None.

12.2.24 Icon

This represents the icon to be used.

```
<!ELEMENT Icon (#PCDATA)>
<!ATTLIST Icon
GIF CDATA #IMPLIED>
```

12.2.24.1 Attributes:

GIF: Will contain the name and the filepath for the image to be displayed.

12.2.24.2 Elements:

Character data.

12.2.24.3 Used In:

Display

12.2.24.4 Examples:

Please refer to the example for Display.

12.2.25 Description

This holds the description of the displayed entity.

```
<!ELEMENT Description (#PCDATA)>
<!ATTLIST Description
NLSID CDATA #IMPLIED>
```

12.2.25.1 Attributes:

NLSID: Will contain the ID used to lookup the string in to the resource bundle

12.2.25.2 Elements:

Character data.

12.2.25.3 Used In:

Display

12.2.25.4 Examples:

Please refer to the example for Display.

12.2.26 Unit

This holds the Unit information for the displayed data.

There are some standard units and unit nls ids that are supported. Please use the appropriate nls ids and display names for these standard units mentioned below. The translation for these system supported units (nls ids that start with "em__sys__"), is done at the system level and does not need to be translated on a per target type basis.

Supported Units:

Standard Percent: used for metrics who values are between 0 and 100%

NLSID: "em__sys__standard_percent"

Display: "%"

Usage: <Unit NLSID="em__sys__standard_percent">%</Unit>

Generic Percent: used for metrics who values can be in +ve and -ve percentages as well - like -50% or 200%

NLSID: "em__sys__generic_percent"

Display: "%"

Usage: <Unit NLSID="em__sys__generic_percent">%</Unit>

```
<!ELEMENT Unit (#PCDATA)>
<!ATTLIST Unit
NLSID CDATA #IMPLIED>
```

12.2.26.1 Attributes:

NLSID: Will contain the ID used to lookup the string in to the resource bundle

12.2.26.2 Elements:

Character data.

12.2.26.3 Used In:

Display

12.2.26.4 Examples:

Please refer to the example for Display.

12.2.27 MonitoringMode

MonitoringMode element indicates the mediator for data collection. Presence of this element in TargetMetadata element, indicates that the target is of cluster type. Mediation is required for a cluster type target to provide data collection consistency across all cluster type target agents. Cluster targets can be OMS mediated or Agent mediated. MEDIATOR attribute specifies the mediation. CLUSTERDESCRIPTOR attribute points to the shared library that implements the cluster interfaces needed by the agent. This is applicable only for AgentMediated clusters.

```
<!ELEMENT MonitoringMode (ValidIf*)>
<!ATTLIST MonitoringMode
MEDIATOR (AgentMediated|OMSMediated) #REQUIRED
CLUSTERDESCRIPTOR CDATA #IMPLIED>
```

12.2.27.1 Attributes:

MEDIATOR: Specifies the mediator to use.

Supported values are:

a) AgentMediated

b) OMSMediated

CLUSTERDESCRIPTOR: Describes the type of the cluster. This is applicable for Agent mediation only.

12.2.27.2 Elements:

ValidIf

12.2.27.3 Used In:

TargetMetadata

12.2.27.4 Examples:

```
<TargetMetadata META_VER="2.0" TYPE="example1" CATEGORY_PROPERTIES="OS;OSVersion">
```

```
. . .
<MonitoringMode MEDIATOR="OMSMediated">
<ValidIf>
<CategoryProp NAME="OSVersion" CHOICES="5.8"/>
<CategoryProp NAME="OSVersion" CHOICES="5.9"/>
</ValidIf
</MonitoringMode>
. . .
</TargetMetadata>
```

This example indicates the target, 'example1' is of cluster type and is OMS Mediated only for the OSVersions 5.8 and 5.9. For the other versions it acts like a normal target. All the agents would monitor the target. Absence of this element makes the target a normal target.

12.2.28 AltSkipCondition

The agent has logic to skip evaluation of metrics for targets that are known to be down to reduce generation of metric errors due to connection failures. Metrics are skipped whenever there is an error in evaluating the Response metric or there is a non-clear severity on the Response, Status condition. If a target needs to have its metric evaluation stop on a condition other than the Response, Status column, this can be specified by creating an AltSkipCondition element.

```
<!ELEMENT AltSkipCondition EMPTY >
<!ATTLIST AltSkipCondition
METRIC CDATA #REQUIRED
COLUMN CDATA #REQUIRED
ASSOC_TARGET CDATA #IMPLIED>
```

12.2.28.1 Attributes:

METRIC: Name of the result metric

COLUMN: Name of the column

ASSOC_TARGET: may be used to point to the conditions in an associated target.

12.2.28.2 Elements:

None

12.2.28.3 Used In:

TargetMetadata

12.2.28.4 Examples:

```
<TargetMetadata TYPE="example1" META_VER="2.0">
<AltSkipCondition METRIC="metric1" COLUMN="Status"/>
<Metric NAME="Response" TYPE="TABLE">
<TableDescriptor>
<ColumnDescriptor NAME="Status" TYPE="NUMBER" IS_KEY="FALSE"/>
</TableDescriptor>
<QueryDescriptor FETCHLET_ID="OS">
<Property NAME="hostname" SCOPE="INSTANCE">NAME</Property>
</QueryDescriptor>
</Metric>
<Metric NAME="metric1" TYPE="TABLE">
<TableDescriptor>
<ColumnDescriptor NAME="Status" TYPE="NUMBER" IS_KEY="FALSE"/>
</TableDescriptor>
```

```
<QueryDescriptor FETCHLET_ID="OS">
<Property NAME="homedir" SCOPE="INSTANCE">home</Property>
</QueryDescriptor>
</Metric>
</TargetMetadata>
```

This example describes a target whose metric evaluation will be skipped whenever there is an error in evaluating the Response metric or there is a non-clear severity on the Response Status column. In addition to that, metric evaluation will be skipped if errors or non-clear severity are encountered on the Status column of the 'metric1' metric as well.

```
<AltSkipCondition METRIC="Response" COLUMN="State" ASSOC_TARGET="t2"/>
<AltSkipCondition METRIC="Response" COLUMN="State" ASSOC_TARGET="t2"/>
```

If the 'example1' target type in the above example was associated with another target type. The AltSkipCondition can be used to skip evaluating example1's metric based on Response metric's 'State' column in the other target type.

The association 't2' has to be defined in the targets.xml file for example1.

12.2.29 CredentialInfo

Credential types are metadata for sets of credentials. It describes the components of the credential (CredentialTypeColumn s), which is the key etc. In some cases, CredentialTypes may be composed of existing CredentialTypes (in this or other target types)

CredentialSets are the instances of CredentialTypes that apply to a particular target. Of particular importance are the monitoring credential sets whose values are mapped to the instance properties of the target.

```
<!ELEMENT CredentialInfo (CredentialType*, CredentialSet*)>
```

12.2.29.1 Attributes:

None

12.2.29.2 Elements:

CredentialType

CredentialSet

12.2.29.3 Used In:

TargetMetadata

12.2.29.4 Examples:

<TargetMetadata TYPE="example1" META_VER="2.0">

. . .

```
<CredentialInfo>
 <CredentialType NAME="DBCreds" >
 <CredentialTypeColumn NAME="DBUsername" IS_KEY="true" />
 <CredentialTypeColumn NAME="DBPassword" />
 <CredentialTypeColumn NAME="DBRole" />
 </CredentialType>
<CredentialSet NAME="DBCredsMonitoring" CREDENTIAL_TYPE="DBCreds"
USAGE="monitoring">
<CredentialSetColumn TYPE_COLUMN="DBUsername" SET_COLUMN="UserName" />
<CredentialSetColumn TYPE_COLUMN="DBPassword" SET_COLUMN="password" />
```

```
<CredentialSetColumn TYPE_COLUMN="DBRole" SET_COLUMN="role" />
</CredentialSet>
<CredentialSet NAME="DBCredsSysdba" CREDENTIAL_TYPE="DBCreds" USAGE="monitoring">
<CredentialSetColumn TYPE_COLUMN="DBUsername" SET_COLUMN="SYSDBAUserName" />
<CredentialSetColumn TYPE_COLUMN="DBPassword" SET_COLUMN="SYSDBApassword" />
<CredentialSetColumn TYPE_COLUMN="DBRole" SET_COLUMN="SYSDBArole" />
</CredentialSet>
</CredentialInfo>
. . .
</TargetMetadata>
```

CredentialInfo may have CredentialType and CredentialSet elements. This is illustrated in this example. Target type, 'example1' is associated with 'DBCreds', 'DBCredsMonitoring' and 'DBCredsSysdba' credentials.

12.2.30 CredentialType

CredentialType elements contain the description of a type as composed of component columns (one of which may be the key) or as a composite of other predefined credential types.

```
<!ELEMENT CredentialType (Display?, (CredentialTypeColumn|CredentialTypeRef)+)>
<!ATTLIST CredentialType
NAME CDATA #REQUIRED>
```

12.2.30.1 Attributes:

NAME: Unique Name of the CredentialType

12.2.30.2 Elements:

CredentialType

CredentialSet

12.2.30.3 Used In:

TargetMetadata

12.2.30.4 Examples:

CredentialType may contain an optional Display element specifying the display characteristics for the CrendentialType and one or more of either the CredentialTypeColumn or the CredentialTypeRef.

```
<CredentialType NAME="HostCreds" >
<CredentialTypeColumn NAME="HostUserName" IS_KEY="TRUE">
<CredentialTypeColumn NAME="HostPassword">
</CredentialType>
```

In this sample 'HostCreds' is declared as a CredentialType. Please refer to a more detailed in CredentialInfo.

12.2.31 CredentialTypeColumn

CredentialType is defined as a set of Credential Columns. Each CredentialTypeColumn may provide a list of values that are the only allowed values for this column.

```
<!ELEMENT CredentialTypeColumn (Display?, CredentialTypeColumnValue*)>
<!ATTLIST CredentialTypeColumn
NAME CDATA #REQUIRED
IS_KEY (TRUE|FALSE) "FALSE">
```

12.2.31.1 Attributes:

NAME: Name of the column

IS_KEY: If multiple sets are created of this credential type, is this the column by which one set is differentiated from another.

TRUE | FALSE (default)

12.2.31.2 Elements:

Display

CredentialTypeColumn

CredentialTypeRef

12.2.31.3 Used In:

CredentialInfo

12.2.31.4 Examples:

CredentialTypeColumn may contain an optional Display element specifying the display characteristics for the CredentialTypeColumn and optional CredentialTypeColumnValue element(s).

```
<CredentialTypeColumn NAME="HostUserName" IS_KEY="TRUE">
<Display FOR_SUMMARY_UI="TRUE">
<Label NLSID="host_username">UserName</Label>
</Display>
</CredentialTypeColumn>
```

The HostUserName column is the key in a (username,password) credential type. This column would be displayed in a *condensed* version of the UI and the label associated with this is 'UserName'. Please refer to a more detailed in CredentialInfo.

12.2.32 CredentialTypeColumnValue

CredentialTypeColumnValue holds the allowed values for a CredentialTypeColumn.

```
<!ELEMENT CredentialTypeColumnValue (#PCDATA)>
<!ATTLIST CredentialTypeColumnValue
IS_DEFAULT (TRUE|FALSE) "FALSE">
```

12.2.32.1 Attributes:

IS_DEFAULT: Is set to true if this is the default value in the list.

TRUE | FALSE (default)

12.2.32.2 Elements:

Character data representing the value.

12.2.32.3 Used In:

CredentialInfo

12.2.32.4 Examples:

```
<CredentialType NAME="DBCreds" >
 <CredentialTypeColumn NAME="DBUsername" IS_KEY="true" />
 <CredentialTypeColumn NAME="DBPassword" />
 <CredentialTypeColumn NAME="DBRole">
```

```
<CredentialTypeColumnValue IS_DEFAULT="true">normal</CredentialTypeColumnValue>
<CredentialTypeColumnValue>sysdba</CredentialTypeColumnValue>
</CredentialTypeColumn>
</CredentialType>
```

The 'DBRole' column in 'DBCreds' credential type may have the following values:

1. normal (default value)

2. sysdba

Please refer to the in CredentialInfo. The example shows the context for the CredentialType element

12.2.33 CredentialTypeRef

This element allows a credential type to refer to other predefined credential types. It contains mapping of the columns in the original credential type to columns of the credential type being defined.

```
<!ELEMENT CredentialTypeRef (CredentialTypeRefColumn*)>
<!ATTLIST CredentialTypeRef
REF_NAME CDATA #REQUIRED
REF_TYPE CDATA #REQUIRED
REF_TARGETTYPE CDATA #IMPLIED
ASSOCIATION CDATA #IMPLIED>
```

12.2.33.1 Attributes:

REF_NAME: Specifies the name for this CredentialTypeRef.

REF_TYPE: Credential type referred to.

REF_TARGETTYPE: The target type that contains the original credential type. Specify Null if this is the same target type.

ASSOCIATION: Refers to the association of this target with the other target for whom credentials are maintained here. Note that this value needs to be one of the AssocTarget elements above.

12.2.33.2 Elements:

CredentialTypeRefColumn

12.2.33.3 Used In:

CredentialType

12.2.33.4 Examples:

```
<CredentialType NAME="FMCreds" >
<CredentialTypeRef NAME="FMDBCreds1" REF_TYPE="DBCreds" REF_TARGET_TYPE="oracle_
database" ASSOCIATION="firstDB">
<CredentialTypeRefColumn NAME="FMUserName1" REF_TYPECOLUMN="DBUsername" />
<CredentialTypeRefColumn NAME="FMpassword1" REF_TYPECOLUMN="DBPassword" />
<CredentialTypeRefColumn NAME="FMRole1" REF_TYPECOLUMN="DBRole" />
</CredentialTypeRef>
</CredentialType>
```

FMCreds defines a credential type whose columns FMUserName1, Fmpassword1 and FMRole are mapped to DBCred's DBUsername, DBPassword and DBRole columns respectively.

Please refer to the example in CredentialInfo. The example shows the context for the CredentialType element

12.2.34 CredentialTypeRefColumn

This element maps the columns in the referred credential type to this credential type's columns.

```
<!ELEMENT CredentialTypeRefColumn EMPTY>
<!ATTLIST CredentialTypeRefColumn
NAME CDATA #REQUIRED
REF_TYPECOLUMN CDATA #REQUIRED>
```

12.2.34.1 Attributes:

NAME: Name of column in this credential type.

REF_TYPECOLUMN: Name of the column in the referred credential type

12.2.34.2 Elements:

None

12.2.34.3 Used In:

CredentialTypeRef

12.2.34.4 Examples:

Please refer to the example for CredentialTypeRef.

12.2.35 CredentialSet

This element defines a set of elements that form a named credential set for this target type. A credential set provides values for one of the credential types defined for this target type. The credential set may contain credentials for one of 3 usages: monitoring, preferred credentials or app specific functionality.

```
<!ELEMENT CredentialSet (Display?, CredentialSetColumn+)>
<!ATTLIST CredentialSet
NAME CDATA #REQUIRED
CREDENTIAL_TYPE CDATA #REQUIRED
USAGE (MONITORING|PREFERRED_CRED|SYSTEM) "MONITORING"
CONTEXT_TYPE (TARGET|CONTAINER|COLLECTION) "TARGET"
CONTEXT CDATA #IMPLIED>
```

12.2.35.1 Attributes:

NAME: Name of the credential set

CREDENTIAL_TYPE: The credential type that this set provides values for.

USAGE: Is this credential set used for monitoring, as preferred credentials or for app specific stuff?

Supported values are:

a) MONITORING (default): Specifies credentials that management applications can use to connect directly to the target.

b) PREFERRED_CRED: Specifies a user's preferred credentials

c) SYSTEM: Specifies a fixed set of credentials that are used by certain specialized applications (patching, cloning etc.)

Attributes introduced in **Enterprise Manager version 10.2:**

CONTEXT_TYPE: Specifies what kind of entity, the set pertains to.

Supported values are:

a) TARGET (default): These are the stored credentials for a target that could be used by applications such as job system, patch etc.

b) CONTAINER: These are the stored credentials for a container. These are always host credentials.

c) COLLECTION: These are credentials associated with user-defined metrics.

CONTEXT: Specifies the metric that this set is for. Refers to collection credentials only.

12.2.35.2 Elements:

Display

CredentialSetColumn

12.2.35.3 Used In:

CredentialInfo

12.2.35.4 Examples:

CredentialSet contains an optional Display element that specifies the display characteristics for the CredentialSet element and at least 1 CredentialSetColumn.

```
<CredentialSet NAME="HostPrefCreds" CREDENTIAL_TYPE="HostCreds" USAGE="PREFERRED_
CRED">
<CredentialSetColumn TYPE_COLUMN="HostUsername" SET_COLUMN="HostPrefUserName" />\
<CredentialSetColumn TYPE_COLUMN="HostPassword" SET_COLUMN="HostPrefPassword" />
</CredentialSet>
This sample illustrates usage of a user credential set.

<CredentialSet NAME="DBCredsMonitoring" CREDENTIAL_TYPE="DBCreds"
USAGE="monitoring">
<CredentialSetColumn TYPE_COLUMN="DBUsername" SET_COLUMN="UserName"/>
<CredentialSetColumn TYPE_COLUMN="DBPassword" SET_COLUMN="password"/>
<CredentialSetColumn TYPE_COLUMN="DBRole" SET_COLUMN="role"/>
</CredentialSet>
DBCredsMonitoring defines a credential set that may be used by the agent for
monitoring a database.
<CredentialSet NAME="HostSystemCreds" CREDENTIAL_TYPE="HostCreds" USAGE="SYSTEM">
<CredentialSetColumn TYPE_COLUMN="HostUsername" SET_COLUMN="HostPrefUserName"/>
<CredentialSetColumn TYPE_COLUMN="HostPassword" SET_COLUMN="HostPrefPassword"/>
</CredentialSet>
```

HostSystemCreds is an example of a System credential type.

Please refer to the example in CredentialInfo. The example shows the context for the CredentialType element

12.2.36 CredentialSetColumn

Credential set columns map the columns of a credential type to the source of their values. In the case of monitoring credential sets, the source is instance properties of the target. This element was first introduced in **Enterprise Manager version 10.2**.

```
<!ELEMENT CredentialSetColumn (Display?, CredentialSetColumnValue*) >
<!ATTLIST CredentialSetColumn
TYPE_COLUMN CDATA #REQUIRED
```

```
SET_COLUMN CDATA #REQUIRED>
```

12.2.36.1 Attributes:

TYPE_COLUMN: Name of the column in the CredentialType.

SET_COLUMN:

12.2.36.2 Elements:

Display

CredentialSetColumnValue

12.2.36.3 Used In:

CredentialSet

12.2.36.4 Examples:

CredentialSetColumn contains an optional Display element that specifies the display characteristics for the CredentialSetColumn element and optional CredentialSetColumnValue element(s).

Please refer to the example for CredentialSet.

12.2.37 CredentialSetColumnValue

This element holds the allowed values for a CredentialSetColumn. This was introduced in **Enterprise Manager version 10.2.**

```
<!ELEMENT CredentialSetColumnValue (#PCDATA)>
<!ATTLIST CredentialSetColumnValue
IS_DEFAULT (TRUE|FALSE) "FALSE">
```

12.2.37.1 Attributes:

IS_DEFAULT: Set this attribute to true if the element represents the default value in the list.

12.2.37.2 Elements:

Character data representing the value.

12.2.37.3 Used In:

CredentialSetColumn

12.2.37.4 Examples:

Please refer to the example for CredentialSet.

12.2.38 InstanceProperties

The InstanceProperties element declares the "properties" of a target type. Some properties are obtained from the targets.xml file, and may be optional or required, and others can be computed using DynamicProperties elements using the values of other properties. The agent uses the information in the InstanceProperties element to determine when a target has not been sufficiently configured, and to compute the dynamic properties for the target. The console UIs use the information about the

properties to create UIs where a target can be created from scratch or an existing target's properties are modified.

InstanceProperties holds target InstanceProperty(s) and DynamicProperties elements.

```
<!ELEMENT InstanceProperties ((InstanceProperty | DynamicProperties)*)>
```

12.2.38.1 Attributes:

None

12.2.38.2 Elements:

InstanceProperty

DynamicProperties

12.2.38.3 Used In:

TargetMetadata

12.2.38.4 Examples:

Please refer to the example for QueryDescriptor and TargetMetadata.

12.2.39 InstanceProperty

An InstanceProperty element contains the definition of an instance property.

```
<!ELEMENT InstanceProperty (ValidIf*, (PCDATA | Display)*)>
<!ATTLIST InstanceProperty
NAME CDATA #REQUIRED
OPTIONAL (TRUE | FALSE) "FALSE"
CREDENTIAL (TRUE | FALSE) "FALSE"
READONLY (TRUE | FALSE) "FALSE"
NEED_REENTER ( TRUE | FALSE) "FALSE"
HIDE_ENTRY ( TRUE | FALSE) "TRUE"
CHECK_ORIGINAL ( TRUE | FALSE) "FALSE"
IS_COMPUTED (TRUE|FALSE) "FALSE"
>
```

12.2.39.1 Attributes:

NAME: Name of the property

OPTIONAL: Is a value for the property required

TRUE | FALSE (default)

CREDENTIAL: Is the property sensitive in nature. Such properties are usually saved obfuscated in targets.xml

TRUE | FALSE (default)

READONLY: Marks this element as ReadOnly.

TRUE | FALSE (default)

NEED_REENTER: If TRUE, will require user to enter the value twice at command line.

TRUE | FALSE (default)

HIDE_ENTRY: If TRUE, will show the character user typed as '*'.

TRUE (default) | FALSE

CHECK_ORIGINAL: If TRUE, before modify, user need to type the original value.

Attributes introduced in **Enterprise Manager version 10.2**:

IS_COMPUTED: If TRUE, indicates that it describes a dynamic property.

12.2.39.2 Elements:

ValidIf

Display

12.2.39.3 Used In:

TargetMetadata

12.2.39.4 Examples:

If InstanceProperty element contains ValidIf elements, all the conditions must be met for the property to be evaluated. InstanceProperty also optionally contains either Display elements or character data.

```
<InstanceProperty NAME="password" OPTIONAL="FALSE" CREDENTIAL="TRUE">
```

Example: for an oracle_database target one InstanceProperty has the NAME "password", which is not OPTIONAL and which is a "CREDENTIAL".

Please refer to additional example for InstanceProperty described in TargetMetadata.

12.2.40 DynamicProperties

DynamicProperties elements allow a target to specify a query that will return a set of values corresponding to the instance properties of the target. The values are turned into target properties and are accessible to other query descriptors from the INSTANCE scope.

```
<!ELEMENT DynamicProperties (ValidIf*, (QueryDescriptor | ExecutionDescriptor)+) >
<!ATTLIST DynamicProperties
NAME CDATA #REQUIRED
PROP_LIST CDATA #IMPLIED
OPT_PROP_LIST CDATA #IMPLIED
FORMAT (TABLE | ROW) "TABLE">
```

> **Note:** If CategoryProperties are instantiated through DynamicProperty evaluation, such a failed DynamicProperty evaluation would cause the agent to reject the target unless the value of the category property is available in targets.xml.

12.2.40.1 Attributes:

NAME: attribute simply identifies the property collector, for error tracing etc.

PROP_LIST: Contains ';' separated values that specify a list of names of properties that can be returned by the query descriptor. The result MUST contain the properties listed here.

OPT_PROP_LIST: Contains ';' separated values that specify a list of names of properties that can be returned by the query descriptor. The result MAY contain the properties listed here.

FORMAT: Specifies the format for the return data.

Supported values are:

a) TABLE (default): If the FORMAT is "TABLE" (the default), the return value must be a table of instance property values. The table returned must be a two-column (NAME, VALUE) table.

b) ROW: If the FORMAT is "ROW", the contents of the one row are taken as the values of the properties in the same order they are listed in the PROP_LIST, and then in the OPT_PROP_LIST lists.

12.2.40.2 Elements:

ValidIf

QueryDescriptor

ExecutionDescriptor

12.2.40.3 Used In:

InstanceProperties

12.2.40.4 Examples:

If DynamicProperties element contains ValidIf elements, all the conditions must be met for the property to be evaluated. In addition to this it must also include at least 1 instance of either QueryDescriptor or ExecutionDescriptor.

The property names in the PROP_LIST and OPT_PROP_LIST are used in conjunction with InstanceProperty declarations while validating target type metadata.

```
<DynamicProperties NAME="VersionAndLocation" FORMAT="ROW" PROP_
LIST="OS;OracleHome;Version">
```

DynamicProperties, 'VersionAndLocation' has a ROW format and returns 'OS', 'OracleHome', 'Version' as properties.

Note that if a query descriptor returns a property value that is already available, the property is ignored. If multiple DynamicProperties queries return a property, the value from the first one is used.

The example in TargetMetadata includes the context for DynamicProperties element.

12.2.41 ExecutionDescriptor

ExecutionDescriptor specifies the execution plan for evaluating a metric. EMAgent executes each statement of the plan, in the order it is defined, to produce a Metric Result. The Metric Result generated as result of the evaluation of the last statement of the execution plan will be returned.

```
<!ELEMENT ExecutionDescriptor (ValidIf*, (GetTable | GetView | GroupBy | Union |
JoinTables)*)>
```

12.2.41.1 Attributes:

None

12.2.41.2 Elements:

ValidIf

GetTable

GetView

GroupBy

Union

JoinTables

12.2.41.3 Used In:

Metric

DynamicProperties

12.2.41.4 Examples:

If ExecutionDescriptor element contains ValidIf elements, all the conditions must be met for the element to be evaluated. In addition to this it must also include 0 or more instances of either 1 of the following elements: GetTable, GetView, GroupBy, Union, JoinTables

ExecutionDescriptor is used to compute aggregation metric.

```
<TargetMetadata META_VER="3.0" TYPE="host">
. . .
<Metric NAME="Load" TYPE="TABLE">
. . .
<ExecutionDescriptor>
<GetTable NAME="DiskActivity"/>
<GetView NAME="AvgSrvcTimeView" FROM_TABLE="DiskActivity">
<Column NAME="DiskActivityavserv"/>
</GetView>
<GroupBy NAME="DA_MaxAvServ" FROM_TABLE="AvgSrvcTimeView">
<AggregateColumn NAME="longestServ"COLUMN_NAME="DiskActivityavserv" OPERATOR="MAX"
/>
</GroupBy>
<GetTable NAME="_LoadInternal"/>
<JoinTables NAME="Load">
<Table NAME="_LoadInternal"/>
<Table NAME="DA_MaxAvServ"/>
</JoinTables>
</ExecutionDescriptor>
</Metric>
<Metric NAME="_LoadInternal" TYPE="TABLE" USAGE_TYPE="HIDDEN">
. . .
</Metric>
<Metric NAME="DiskActivity" TYPE="TABLE">
. . .
</Metric>
</TargetMetadata>
```

This ExecutionDescriptor in this sample generates the following intermediate Metric results after executing each statement:

§ GetTable: The metric result, 'DiskActivity' contains all the columns of 'DiskActivity' metric. The result of this operation is similar to the SQL statement,

"Select * from DiskActivity"

§ GetView: The metric result, 'AvgSrvcTimeView' contains only 'DiskActivityavserv' column of 'DiskActivity' metric.

§ GroupBy: Metric result, 'DA_MaxAvServ' is a grouping of the 'AvgSrvcTimeView' based on the 'longestServ' which is the max value of the 'DiskActivityavserv′ column.

§ GetTable: Metric result, '_LoadInternal' contains all the columns of the '_ LoadInternal' metric.

§ JoinTable: 'Load' metric contains a join of the '_LoadInternal' and 'DA_MaxAvServ' metrics.

The last metric is returned as the result for the 'Load' metric.

12.2.42 GetTable

This element is used within an ExecutionDescriptor element and is equivalent to the following SQL operation:

Select * from T

T is a metric.

```
<!ELEMENT GetTable EMPTY>
<!ATTLIST GetTable
NAME CDATA #REQUIRED
ASSOC_TARGET CDATA #IMPLIED
METRIC_NAME CDATA #IMPLIED
USE_CACHE (TRUE | FALSE | TRUE_IF_COLLECT) "FALSE">
```

12.2.42.1 Attributes:

NAME: Name of the metric.

ASSOC_TARGET: Target from which data is collected. This attribute is optional and when omitted, the METRIC_NAME points at a metric in the same target.

METRIC_NAME: Name of the metric that originates the request. If omitted, attribute NAME is used as METRIC_NAME.

USE_CACHE: Specifies whether the data can be fetched from the cache.

TRUE | FALSE (default) | TRUE_IF_COLLECT

12.2.42.2 Elements:

None

12.2.42.3 Used In:

ExecutionDescriptor

12.2.42.4 Examples:

```
<GetTable NAME="DiskActivity"/>
```

This statement gets all the columns of the 'DiskActivity' metric.

Please refer to the examples in ExecutionDescriptor.

12.2.43 GetView

GetView creates a sub-table from a table. The newly created table is identified by the NAME attribute. It must be unique in the ExecutionDescriptor. This element is equivalent to the following SQL statement:

Select column1, column2,. . . from T

T is a metric

```
<!ELEMENT GetView ((ComputeColumn | Column)*, (Filter | In)*)>
<!ATTLIST GetView
NAME CDATA #REQUIRED
FROM_TABLE CDATA #REQUIRED>
```

12.2.43.1 Attributes:

NAME: Name of the view

FROM_TABLE: Table from which to generate the view.

12.2.43.2 Elements:

ComputeColumn

Column

Filter

In

12.2.43.3 Used In:

ExecutionDescriptor

12.2.43.4 Examples:

GetView may contain 0 or more instances of either 'ComputeColumn' or 'Column' elements and 0 or more instances of either 'Filter' or 'In' elements.

```
<GetView NAME="AvgSrvcTimeView" FROM_TABLE="DiskActivity">
<Column NAME="DiskActivityavserv"/>
</GetView>
```

This is equivalent to the following SQL statement:

create view AvgSrvcTimeView as

select DiskActivityavserv from DiskActivity.

If no Column elements are present in GetView, all columns in the table are included.

Please refer to the example in ExecutionDescriptor.

12.2.44 Filter

Specifies the filter criteria. Filter is used to determine whether a row will be included in the new table. If a row does not satisfy any Filter criteria, it will be excluded.

```
<!ELEMENT Filter (#PCDATA)>
<!ATTLIST Filter
COLUMN_NAME CDATA #REQUIRED
SCOPE (GLOBAL | INSTANCE | SYSTEMGLOBAL) "GLOBAL"
OPERATOR (EQ | LT | GT | LE | GE | NE | CONTAINS | MATCH | ISNULL | ISNOTNULL)
"EQ">
```

12.2.44.1 Attributes:

COLUMN_NAME: Column name on which the filter criteria is to be applied.

SCOPE:

Supported values are:

a) GLOBAL (default)

b) INSTANCE

c) SYSTEMGLOBAL

OPERATOR: Specifies the operation to perform.

Supported operators are:

a) EQ (default): Equal

b) LT: Less than

c) GT: Greater than

d) LE: Less than or equal to

e) GE: Greater than or equal to

f) NE: Not equals

g) CONTAINS: contains

h) MATCH: matches

i) ISNULL: is NULL

j) ISNOTNULL: is not NULL

12.2.44.2 Elements:

Character data representing the filter criteria

12.2.44.3 Used In:

GetView

12.2.44.4 Examples:

```
<ExecutionDescriptor>
<GetTable NAME="Servlet_raw" USE_CACHE="TRUE_IF_COLLECT"/>
<GetView NAME="result" FROM_TABLE="Servlet_raw">
<Filter COLUMN_NAME="totalRequests1" OPERATOR="GT">0</Filter>
</GetView>
</ExecutionDescriptor>
```

The result of the ExecutionDescriptor is the metric 'result' that has totalRequests1 > 0

12.2.45 Column

Represents a column to include in the new table.

```
<!ELEMENT Column EMPTY>
<!ATTLIST Column
NAME CDATA #REQUIRED
COLUMN_NAME CDATA #IMPLIED
TABLE_NAME CDATA #IMPLIED>
```

12.2.45.1 Attributes:

NAME: Specifies the name of the column of a metric

COLUMN_NAME: Specifies the name of the column. It can be omitted if it is same as NAME.

TABLE_NAME: Specifies the name of the metric. If Column is a part of the GetView element, this attribute must be excluded.

12.2.45.2 Elements:

None

12.2.45.3 Used In:

GetView

JoinTables

12.2.45.4 Examples:

```
<Column NAME="DiskActivityavserv"/>
'DiskActivityavserv' is the metric column that is selected for the operation.
<Column NAME="responseTime" TABLE_NAME="groupbyapps"/>
```

Column, 'responseTime' from 'groupbyapps' metric is selected for the operation.

Please refer to the example in ExecutionDescriptor.

12.2.46 ComputeColumn

This element describes how to compute the values of a column.

```
<!ELEMENT ComputeColumn EMPTY>
<!ATTLIST ComputeColumn
NAME CDATA #REQUIRED
EXPR CDATA #REQUIRED
IS_VALUE (TRUE | FALSE) "FALSE"
DEFAULT_WHEN_EMPTY (TRUE | FALSE) "FALSE"
DEFAULT_VALUE CDATA #IMPLIED>
```

12.2.46.1 Attributes:

NAME: Name of the column

EXPR: Expression that is evaluated to calculate the value. Support for string expressions is introduced in **Enterprise Manager version 10.2**. Please refer to the example of ColumnDescriptor for details about the expression grammar and usage.

IS_VALUE: If set to TRUE, the EXPR points to the actual string value else EXPR is the expression used to calculate the column.

TRUE | FALSE (default)

DEFAULT_WHEN_EMPTY:

TRUE | FALSE (default)

DEFAULT_VALUE: Default value for the column. If not specified the default value will be set to "0"

12.2.46.2 Elements:

None

12.2.46.3 Used In:

GetView

GroupBy

12.2.46.4 Examples:

```
<GetView NAME="NEW_TABLE" FROM_TABLE="ORIG_TABLE">
<Column NAME="cn1" COLUMN_NAME="A" /><Column NAME="cn2" COLUMN_NAME="B" />
<ComputeColumn NAME="cn3" EXPR="cn1+cn2" />
<ComputeColumn NAME="cn4" EXPR="cn1/cn2" DEFAULT_WHEN_EMPTY="TRUE" DEFAULT_
VALUE="1"/>
<ComputeColumn NAME="isMultiThreaded.value"IS_VALUE="TRUE"EXPR="true" />
<Filter COLUMN_NAME="C" OPERATOR="EQ">3</Filter>
</GetView>
```

Please refer to the example of ColumnDescriptor for details about the expression grammar and additional examples of expressions that are acceptable in the EXPR attribute.

12.2.47 In

This element is equivalent of the SQL Statement

select * from from_table where

column_name in (select in_column_name from in_table_name)

```
<!ELEMENT In EMPTY>
<!ATTLIST In
COLUMN_NAME CDATA #REQUIRED
IN_TABLE_NAME CDATA #REQUIRED
IN_COLUMN_NAME CDATA #REQUIRED>
```

12.2.47.1 Attributes:

COLUMN_NAME: Column name to search for.

IN_TABLE_NAME: Table in which to search for.

IN_COLUMN_NAME: Column name of the table specified in IN_TABLE_NAME attribute.

12.2.47.2 Elements:

None

12.2.47.3 Used In:

GetView

12.2.47.4 Examples:

None.

12.2.48 GroupBy

GroupBy will perform aggregation operation on a table/view to create a new table. This element is equivalent to the SQL statement

Select sum(column_name) from table_name

```
<!ELEMENT GroupBy (By*, (AggregateColumn | ComputeColumn)*)>
<!ATTLIST GroupBy
NAME CDATA #REQUIRED
FROM_TABLE CDATA #REQUIRED>
```

12.2.48.1 Attributes:

NAME: Specifies the name of the grouping

FROM_TABLE: Specifies the name of the metric.

12.2.48.2 Elements:

By

AggregateColumn

ComputeColumn

12.2.48.3 Used In:

GetView

12.2.48.4 Examples:

This element may include 0 or more elements of 'By' and 0 or more elements of either AggregateColumn or ComputeColumn.

<GroupBy NAME="DA_MaxAvServ" FROM_TABLE="AvgSrvcTimeView">

This statement results in DA_MaxAvServ containing the result of the groupby operation applied to the AvgSrvcTimeView metric.

Please refer to the example in ExecutionDescriptor.

12.2.49 By

'By' element defines a column that constitutes a GroupBy clause.

Each 'By' element will be added to the result table as a column.

```
<!ELEMENT By EMPTY>
<!ATTLIST By
NAME CDATA #REQUIRED
COLUMN_NAME CDATA #IMPLIED>
```

12.2.49.1 Attributes:

NAME: Name of the new column.

COLUMN_NAME: name for the new column.

12.2.49.2 Elements:

None

12.2.49.3 Used In:

GroupBy

12.2.49.4 Examples:

NAME must be unique within the new table. If COLUMN_NAME is not given, will be the same as NAME

12.2.50 AggregateColumn

This element describes an operation to be performed on a column.

```
<!ELEMENT AggregateColumn EMPTY>
<!ATTLIST AggregateColumn
NAME CDATA #REQUIRED
COLUMN_NAME CDATA #REQUIRED
OPERATOR (MAX | MIN | SUM | AVG | COUNT) "SUM">
```

12.2.50.1 Attributes:

NAME: Specifies the name of the AggregateColumn

COLUMN_NAME: Specifies the column name of the metric on which the operation is to be performed.

OPERATOR: Specifies the operation to be performed.

Supported operators are:

a) MAX

b) MIN

c) SUM (default)

d) AVG

e) COUNT

12.2.50.2 Elements:

None

12.2.50.3 Used In:

GroupBy

12.2.50.4 Examples:

```
<AggregateColumn NAME="longestServ" COLUMN_NAME="DiskActivityavserv"
OPERATOR="MAX" />
```

This operation results in the calculation of the Maximum value from the 'DiskActivityavserv' column.

Please refer to the example in ExecutionDescriptor.

12.2.51 Union

Union element describes an operation to combine/merge two or more tables. Only tables with the same number of columns can be unioned together. This element is equivalent of the SQL statement:

Select * from T1 Union

Select * from T2

```
<!ELEMENT Union (Table+)>
<!ATTLIST Union
NAME CDATA #REQUIRED
DISTINCT (TRUE | FALSE) "FALSE">
```

12.2.51.1 Attributes:

NAME: Specifies the name for the union.

DISTINCT: If TRUE, any row that is exactly the same as previous row will be discarded.

TRUE | FALSE (default)

12.2.51.2 Elements:

Table

12.2.51.3 Used In:

ExecutionDescriptor

12.2.51.4 Examples:

Union element must include atleast 1 Table element.

```
<Union NAME="result4">
<Table NAME="result" />
<Table NAME="result1" />
<Table NAME="result2" />
<Table NAME="result3" />
</Union>
```

'result4' will be a union of the 4 Tables. The resulting table will have the same column name list as first table.

> **Note:** Only tables with the same number of columns can be unioned together. The newly created table is identified by NAME. It must be unique in the ExecutionDescriptor.

12.2.52 Table

This element describes the metrics that can take part in Table operations like Union and JoinTables.

```
<!ELEMENT Table EMPTY>
<!ATTLIST Table
NAME CDATA #REQUIRED>
```

12.2.52.1 Attributes:

NAME: Specifies the name of the metric.

12.2.52.2 Elements:

Table

12.2.52.3 Used In:

ExecutionDescriptor

12.2.52.4 Examples:

```
<Table NAME="result" />
```

This example describes 'result' as a metric that can participate in Table operations.

Please refer to the example of Union.

12.2.53 JoinTables

This element describes the join operation. It is equivalent of the SQL statement:

Select C1, C2 where . . .

```
<!ELEMENT JoinTables (Table, Table+, (Column | ComputeColumn)*, Where*)>
<!ATTLIST JoinTables
NAME CDATA #REQUIRED
OUTER (TRUE | FALSE) "FALSE"
BOTH_SIDE (TRUE | FALSE) "FALSE">
```

12.2.53.1 Attributes:

NAME: Specifies the name of the join.

OUTER: If true, specifies an OUTER join.

TRUE | FALSE (default).

BOTH_SIDE: If true, specifies BOTH_SIDE. The results would contain the same number of rows as a UNION.

TRUE | FALSE(default)

12.2.53.2 Elements:

Table

Column

ComputeColumn

Where

12.2.53.3 Used In:

ExecutionDescriptor

12.2.53.4 Examples:

JoinTables must include atleast 2 'Table' elements, 0 or more elements of either 'Column' or 'ComputeColumn' and 0 ore more elements of 'Where' element.

```
<JoinTables NAME="Load">
<Table NAME="_LoadInternal"/>
<Table NAME="DA_MaxAvServ"/>
</JoinTables>
```

This example defines 'Load' as a join of '_LoadInterval' and 'DA_MaxAvServ' metrics.

Please refer to the example in ExecutionDescriptor.

12.2.54 Where

```
<!ELEMENT Where EMPTY>
<!ATTLIST Where
FROM_TABLE CDATA #REQUIRED
FROM_KEY CDATA #REQUIRED
OPERATOR (EQ | GT | GE | LE | LT | NE | CONTAINS | MATCH) "EQ"
JOIN_TABLE CDATA #REQUIRED
JOIN_KEY CDATA #REQUIRED>
```

12.2.54.1 Attributes:

FROM_TABLE:

FROM_KEY:

OPERATOR: Specifies the operator in the where clause.

Supported operators are:

a) EQ (default): Equals

b) GT: Greater than

c) GE: Greater than or equal to

d) LE: Less than or equal to

e) LT: Less than

f) NE: Not equals

g) CONTAINS: Contains

h) MATCH: matches

JOIN_TABLE:

JOIN_KEY:

12.2.54.2 Elements:

None

12.2.54.3 Used In:

JoinTables

12.2.54.4 Examples:

```
<JoinTables NAME="all" OUTER="TRUE">
<Table NAME="groupbyapps" />
<Table NAME="groupbyappsejb" />
<Column NAME="Application" TABLE_NAME="groupbyapps"/>
<Column NAME="activeRequests" TABLE_NAME="groupbyapps"/>
<Column NAME="responseTime" TABLE_NAME="groupbyapps"/>
<Column NAME="activeMethods" TABLE_NAME="groupbyappsejb"/>
<Column NAME="avgMethodExecTime" TABLE_NAME="groupbyappsejb"/>
<Where FROM_TABLE="groupbyapps" JOIN_TABLE="groupbyappsejb"
FROM_KEY="Application" JOIN_KEY="ApplicationName"/>
</JoinTables>
```

12.2.55 PushDescriptor

The push descriptor identifies a recvlet that is responsible for supplying data and/or events for a metric, and specifies data to be passed to that recvlet for each target. The name used for the recvlet here should match the recvlet name used in recvlets.reg.

```
<!ELEMENT PushDescriptor (ValidIf*, Property*) >
<!ATTLIST PushDescriptor
RECVLET_ID CDATA #REQUIRED>
```

12.2.55.1 Attributes:

RECVLET_ID: Specifies the ID of the recvlet to use that is known to the EMAgent. This attribute must point to an element from the $ORACLE_HOME/lib/recvlets.reg file.

12.2.55.2 Elements:

ValidIf

Property

12.2.55.3 Used In:

Metric

12.2.55.4 Examples:

PushDescriptor may include 0 or more elements of 'ValidIf' and 'Property'.

```
<Metric . . .>
. . .
<PushDescriptor RECVLET_ID="AQMetrics">
<Property NAME="QueueName" SCOPE="GLOBAL">ALERT_QUE</Property>
<Property NAME="MachineName" SCOPE="INSTANCE">MachineName</Property>
<Property NAME="Port" SCOPE="INSTANCE">Port</Property>
```

```
<Property NAME="SID" SCOPE="INSTANCE">SID</Property>
<Property NAME="UserName" SCOPE="INSTANCE">UserName</Property>
<Property NAME="password" SCOPE="INSTANCE">password</Property>
<Property NAME="Role" SCOPE="INSTANCE">Role</Property>
<Property NAME="InstanceName" SCOPE="INSTANCE">InstanceName</Property>
<Property NAME="KeyField" SCOPE="GLOBAL">OBJECT_NAME</Property>
<Property NAME="KeyColumn" SCOPE="GLOBAL">name</Property>
</PushDescriptor>
</Metric>
```

The properties included in the PushDescriptor are passed to the 'AQMetrics' recvlet. This recvlet provides data and/or events for the metric.

12.3 Target Collection

Target Collection drives the background collection of metrics for the purposes of uploading their values in a central repository and/or the check of their values against specified conditions.

Note that the XML files conforming to this DDT will be generated by the system (Could be generated from Servlet Frontend or collector).

EMAgent can have more than one collection files each containing metrics that need to be collected for a particular target.

12.3.1 TargetCollection

It can have 0 or more CollectionItem and CollectionLevel elements. The CollectionLevel element applies only to the default collections.

```
<!ELEMENT TargetCollection ( CollectionLevel*, CollectionItem* ) >
<!ATTLIST TargetCollection
TYPE CDATA #REQUIRED
NAME CDATA #IMPLIED
LEVEL CDATA #IMPLIED
INCLUDE_DEFAULT (TRUE|FALSE) "TRUE">
```

12.3.1.1 Attributes:

TYPE: Specifies the target type.

NAME: Specifies the name of the target. If this is the top-level element, NAME must not be null. If this file is included, it could be null. This attribute applies only to instance specific collections.

LEVEL: Specifies the collection level. The default will be the minimum. This attribute applies only to instance specific collections.

INCLUDE_DEFAULT: If set to TRUE, will include default collection with the same target TYPE. This attribute applies only to instance specific collections.

TRUE (default) | FALSE

12.3.1.2 Elements:

CollectionLevel

CollectionItem

12.3.1.3 Used In:

TargetCollection is a top-level element.

12.3.1.4 Examples:

```
<TargetCollection TYPE="preferred" >
<CollectionItem NAME = "Load" UPLOAD_ON_FETCH="TRUE">
<Schedule OFFSET_TYPE="INCREMENTAL">
<IntervalSchedule INTERVAL = "15" TIME_UNIT = "Min"/>
</Schedule>
<MetricColl NAME = "NfsOperations">
<Filter COLUMN_NAME="NfsCallsRate" OPERATOR="LE">100000</Filter>
<Condition COLUMN_NAME="NfsServPercentBadCalls" CRITICAL="10" WARNING="5"
OPERATOR="GT" OCCURRENCES="3" MESSAGE="NFS Bad Calls are %value%%%." MESSAGE_
NLSID="netapp_filer_nfs_operations_nfs_per_bad_calls"/>
</MetricColl>
<MetricColl NAME = "CifsOperations">
<Filter COLUMN_NAME="CifsCallsRate" OPERATOR="LE">100000</Filter>
<Condition COLUMN_NAME="CifsPercentBadCalls" CRITICAL="10" WARNING="5"
OPERATOR="GT" OCCURRENCES="3" MESSAGE="CIFS Bad Calls %value%%%." MESSAGE_
NLSID="netapp_filer_cifs_operations_cifs_per_bad_calls"/>
</MetricColl>
<MetricColl NAME = "SystemLoad" />
</CollectionItem>
</TargetCollection>
```

This example illustrates the preferred method of declaring CollectionItems. One collection item specifies multiple metrics with their own filter criteria and Condition elements defined.

```
<TargetCollection TYPE="network" >
<CollectionItem NAME = "Response">
<Schedule>
<IntervalSchedule INTERVAL = "300" TIME_UNIT = "Sec"/>
</Schedule>
<Condition COLUMN_NAME="Status"
CRITICAL="ok"
OPERATOR="NE"
OCCURRENCES="3"
MESSAGE="%target% adaptor is inaccesible or is connected."
MESSAGE_NLSID="network_response_status"/>
</CollectionItem>
</TargetCollection>
```

This example describes a default collection for 'network' target type. The metric is provided as the NAME attribute for the CollectionItem. This method of providing metric NAME in the CollectionItem is for backward compatibility only. Please use the example with MetricColls instead.

TargetCollection includes optional 'CollectionLevel' element(s) and 'CollectionItem' element(s)

```
<TargetCollection TYPE="oracle_beacon" >
```

This is a typical usage for TargetCollection element. It describes TargetCollection for 'oracle_beacon'.

```
<TargetCollection TYPE="oracle_csa_collector" NAME="CSA_collector">
```

NAME attribute implies that this might not be in the default_collections.

12.3.2 CollectionLevel

This element represents Collection Level List. It applies only to the default collections. The order implies the 'contains' relationship.

```
<!ELEMENT CollectionLevel EMPTY>
<!ATTLIST CollectionLevel
NAME CDATA #REQUIRED>
```

12.3.2.1 Attributes

NAME: the name of the collection level

12.3.2.2 Elements

None

12.3.2.3 Used In

TargetCollection

12.3.2.4 Examples

```
<TargetCollection TYPE="example2">
<CollectionLevel NAME="LEVEL1"/>
<CollectionLevel NAME="LEVEL2"/>
<CollectionLevel NAME="LEVEL3"/>
. . .
<MetricColl NAME="metric1">
<ItemProperty NAME="prop1">foo</ItemProperty>
<Condition COLUMN_NAME="value" CRITICAL="bar" OPERATOR="EQ"/>
</MetricColl>
</TargetCollection>
```

The order for the collection level implies that LEVEL2 contains LEVEL1 collection items and LEVEL3 would include both LEVEL2 and LEVEL1 items.

Once the levels are declared using this element, the LEVEL attributes would refer to these levels.

12.3.3 CollectionItem

A CollectionItem defines the collection of one or more metrics. It has a schedule.

```
<!ELEMENT CollectionItem (ValidIf*, Schedule?, (MetricColl+ | (ItemProperty*,
Filter*, LimitRows?, Condition*) ) )>
<!ATTLIST CollectionItem
NAME CDATA #REQUIRED
LEVEL CDATA #IMPLIED
UPLOAD CDATA "YES"
UPLOAD_ON_FETCH (TRUE | FALSE) "FALSE"
ATOMIC_UPLOAD (TRUE | FALSE) "FALSE"
POSTLOAD_PROC CDATA #IMPLIED
PRELOAD_PROC CDATA #IMPLIED
CONFIG (TRUE | FALSE) "FALSE"
CONFIG_METADATA_VERSION CDATA #IMPLIED
TIMEOUT CDATA #IMPLIED
COLLECT_WHEN_DOWN (TRUE | FALSE) "FALSE"
COLLECT_WHEN_ALTSKIP (TRUE | FALSE) "FALSE"
COMBINE_WITH_OTHER_COLLECTION (TRUE | FALSE) "TRUE"
PROXY_TARGET_NAME CDATA #IMPLIED
PROXY_TARGET_TYPE CDATA #IMPLIED
PROXY_TARGET_TZ CDATA #IMPLIED
PROXY_TARGET_TZ_REGION CDATA #IMPLIED
INITIAL_UPLOADS CDATA #IMPLIED
DISABLED (TRUE | FALSE) "FALSE"
REQUIRED (TRUE | FALSE) "FALSE">
```

12.3.3.1 Attributes

NAME: Specifies the name of the collection.

LEVEL: The collection level.

UPLOAD: This attribute, if not specified, will be deemed as YES. NUMBER indicates how often the CollectionItem is uploaded. (Upload every 'n' collections).

YES (default) | NO | *NUMERIC*

UPLOAD_ON_FETCH: Collection Items marked as UPLOAD_ON_FETCH will have the same behavior as ATOMIC_UPLOAD with 1 difference - the upload occurs immediately.

TRUE | FALSE (default)

COLLECT_WHEN_DOWN: The default behavior is that the collection stops if the Response metric (if present for the target) indicates that the status of the target is down. The exception being the Response metric itself. But the behavior of not collecting when target is down can be overridden by setting this attribute to TRUE.

TRUE | FALSE (default)

COLLECT_WHEN_ALTSKIP: The default behavior is that the collection of metrics stops if an AltSkipCondition has been defined and there is a severity on the condition. Setting this attribute to TRUE allows collections to proceed even when this is the case. Note that a severity on the Response/Status condition is only overcome by using the COLLECT_WHEN_DOWN attribute.

TRUE | FALSE (default)

PROXY_TARGET_NAME: Used for proxy collection support, specifies Name of target data and severities should be uploaded for

PROXY_TARGET_TYPE: Used for proxy collection support, specifies Type.

PROXY_TARGET_TZ_REGION: Used for proxy collection support, specifies Timezone Region String (Eg "US/Pacific")

PROXY_TARGET_TZ: Used for proxy collection support, specifies Timezone (minutes from GMT: eg -420)

TIMEOUT: This is the time by which the metric evaluation is expected to finish. The time is provided in seconds. If the evaluation takes more than this time, the agent aborts the metric evaluation and returns a TIMEOUT exception. If this attribute is not provided or a value of zero, the agent defaults to twice the frequency of this metric evaluation in the collection file. Users can provide a time of less than zero to avoid any sort of timeout. A value less than 0 will force the agent to wait until the metric is evaluated completely.

POSTLOAD_PROC: Only applicable in UPLOAD_ON_FETCH situations. This attribute specifies an optional pl/sql procedures that the receiver should invoke when it receives the file with the contents of this collection.

PRELOAD_PROC: Only applicable in UPLOAD_ON_FETCH situations. This attribute specifies an optional pl/sql procedures that the receiver should invoke when it receives the file with the contents of this collection.

CONFIG: This attribute is used to tag collections of CONFIG metrics - these are handled specially by the EM framework. Note: Collection Items for *CONFIG* Metrics cannot specify ATOMIC_UPLOAD as FALSE. TRUE | FALSE (default)

INITIAL_UPLOADS: Defaults to 2, but can be set to a different number if more initial uploads need to be sent up before skipping uploads based on the UPLOAD parameter.

Attributes introduced in **Enterprise Manager version 10.2**:

ATOMIC_UPLOAD: Collection Items marked as *ATOMIC_UPLOAD* will be bundled into a single file which will be uploaded in the regular upload interval (5 minutes).

TRUE | FALSE (default)

CONFIG_METADATA_VERSION: This attribute is used to specify version of CONFIG metrics.

COMBINE_WITH_OTHER_COLLECTION: Agent typically combines collections and executes them in a single thread sequentially to save on threads based on the interval. This can cause some delay in the metric execution if a previous one is taking some time. However some metrics would require to be executed on time. Setting this flag to FALSE would ensure that this metric is executed in its own thread.

TRUE (default) | FALSE

DISABLED: If set to TRUE, the agent will ignore this collection item.

REQUIRED: If set to TRUE, the console will disallow users from disabling this item.

12.3.3.2 Elements

ValidIf

Schedule

MetricColl

ItemProperty

Filter (for TargetCollection)

LimitRows

Condition

12.3.3.3 Used In

TargetCollection

12.3.3.4 Examples

CollectionItem may contain 'ValidIf' element(s) which must all be satisfied for the CollectionItem to be evaluated. It may contain an optional 'Schedule' element and either a 'MetricColl' element or either 'ItemProperty', 'Filter', 'LimitRows' or 'Condition' elements.

For backward compatibility, a single metric can be specified using the NAME attribute, and its properties, filters and conditions can be provided as child elements.

The *NEW* preferred DTD has one or more Metric elements within a CollectionItem each indicating a metric to collect, and the filters, thresholds, etc. to associate with it.

```
<CollectionItem NAME="ProgramResourceUtilization">
```

This is an example of a simplest form of this element. 'ProgramResourceUtilization is a CollectionItem.

```
<CollectionItem NAME = "general_collection" UPLOAD="12" INITIAL_UPLOADS="4">
```

In this sample, 'general_collection' will be uploaded 4 times (number dictated by INITIAL_UPLOADS) initially and after that it will be uploaded once every 12 times (number dictated by UPLOAD attribute).

```
<CollectionItem NAME = "Inventory" UPLOAD_ON_FETCH = "TRUE" TIMEOUT = "3600">
```

'Inventory' collection item will be uploaded when the metrics are collected. Timeout specified is 1 hour.

```
<CollectionItem NAME = "oracle_security" UPLOAD_ON_FETCH = "TRUE" CONFIG = "TRUE">
```
'oracle_security' in this sample, involves collecting CONFIG metrics and should be indicated as such to the EMAgent.

```
<CollectionItem NAME = "UserResourceUsage" UPLOAD="YES" UPLOAD_ON_FETCH="FALSE"
COLLECT_WHEN_DOWN="FALSE">
```
The UPLOAD attribute can take a 'Yes', 'No' or a numeric value. 'UserResourceUsage' collection will be attempted even when the response metric indicates that the target is down.

The PROXY_TARGET_TZ_REGION takes precedence over PROXY_TARGET_TZ if both are specified

Please refer to the example in TargetCollection.

12.3.4 MetricColl

The MetricColl element refers to a metric that is being collected within a collection item.

```
<!ELEMENT MetricColl (ItemProperty*, Filter*, LimitRows?, Condition*)>
<!ATTLIST MetricColl
NAME CDATA #REQUIRED
TRANSIENT (TRUE|FALSE) "FALSE"
UPLOAD_IF_SEVERITY (WARNING|CRITICAL|CHANGE_ONLY) "CHANGE_ONLY">
```

12.3.4.1 Attributes

NAME: This is the name of the metric to collect. Attributes introduced in version 10.2.

TRANSIENT: If this attribute is set to TRUE would indicate that the data of this metric should be uploaded or is collected to refresh the cache used in the evaluation of other metrics.

UPLOAD_IF_SEVERITY: Only effective when UPLOAD=NO and UPLOAD=N>1

Supported values are:

a) CHANGE_ONLY (default): Upload data when there is severity change.

c) WARNING: Upload data when there is severity change and when any condition is in WARNING or CRITICAL

d) CRITICAL: Upload data when there is severity change and when any condition is in CRITICAL

12.3.4.2 Elements

ItemProperty

Filter

LimitRows

Condition

12.3.4.3 Used In

CollectionItem

12.3.4.4 Examples

MetricColl may include optional ItemProperty element(s), Filter element(s), a single optional LimitRows element and optional Condition element(s).

```
<MetricColl NAME = "WebServicesService"/>
```
This is the typical usage for MetricColl. 'WebServicesService' metric is associated with a CollectionItem.

Please refer to the examples in TargetCollection.

12.3.5 LimitRows

LimitRows is a filtering mechanism that can be applied to the collected data, before the data is sent to the repository via the Upload Manager. It limits the number of rows hat are uploaded.

```
<!ELEMENT LimitRows EMPTY>
<!ATTLIST LimitRows
COLUMN_NAME CDATA #IMPLIED
SORT_ORDER (ASCEND|DESCEND|NO_ORDER) "NO_ORDER"
LIMIT_TO CDATA #REQUIRED>
```

12.3.5.1 Attributes

COLUMN_NAME:

SORT_ORDER:

Supported values are:

a) ASCEND:

b) DESCEND:

c) NO_ORDER (default):

LIMIT_TO: Specifies the limit for the number of rows in the collection.

12.3.5.2 Elements

None

12.3.5.3 Used In

CollectionItem

MetricColl

12.3.5.4 Examples

```
<LimitRows LIMIT_TO="16" />
```
This example limits the collection results to 16 rows.

12.3.6 ItemProperty

This element describes a name value pair for a property.

```
<!ELEMENT ItemProperty (#PCDATA)>
<!ATTLIST ItemProperty
NAME CDATA #REQUIRED
ENCRYPTED (NA|FALSE|TRUE) "NA"
>
```

12.3.6.1 Attributes

NAME: Name of the Item property

ENCRYPTED: Indicates whether the property value will be encrypted.

The following values are defined for the attribute:

NA (default): Encryption is not available. The agent will not attempt to encrypt the value.

FALSE: Encryption is available. The agent will attempt to encrypt the value.

TRUE: Encryption is available. The agent has encrypted the value.

12.3.6.2 Elements

Character data

12.3.6.3 Used In

CollectionItem

MetricColl

12.3.6.4 Examples

```
<TargetCollection TYPE="example2">
. . .
<MetricColl NAME="metric1">
<ItemProperty NAME="prop1">foo</ItemProperty>
<Condition COLUMN_NAME="value" CRITICAL="bar" OPERATOR="EQ"/>
</MetricColl>
</TargetCollection>
```

'prop1', ItemProperty will be utilized to compute the value of 'prop1' property defined in 'USER' scope in 'metric1' in TargetMetadata for target type 'example2'.

12.3.7 Filter (for TargetCollection)

Filter defines a filtering mechanism that can be applied to the collected data before the data is sent to the repository via the Upload manager. If filtering is not applied, all the data that is collected through a Fetchlet is sent to the repository. As a result the repository can get filled quickly when uploading certain metrics. To alleviate this problem, filtering mechanism is applied to the data before uploading. The filter criteria are specified in collection xml.

> **Note:** Filter elements for TargetCollection and TargetMetadata are different.

```
<!ELEMENT Filter (#PCDATA)>
<!ATTLIST Filter
COLUMN_NAME CDATA #REQUIRED
OPERATOR (EQ|LT|GT|LE|GE|NE|CONTAINS|MATCH) "EQ"
AFTER_SEVERITY_CHECKING (TRUE|FALSE) "FALSE" >
```

12.3.7.1 Attributes

COLUMN_NAME: Name of the column in the metric to filter on.

OPERATOR: Specifies the operation

EQ (default): Equals

LT: Less than

GT: Greater than

LE: Less than or equal to

GE: Greater than or equal to

NE: Not equals

CONTAINS: Contains

MATCH: Matches

AFTER_SECURITY_CHECKING: If TRUE, filter will be applied after severity checking.

TRUE | FALSE (default)

12.3.7.2 Elements

Character data

12.3.7.3 Used In

CollectionItem

MetricColl

12.3.7.4 Examples

```
<Filter COLUMN_NAME="total_connections" OPERATOR="NE">0</Filter>
```

The result will include only those rows where 'total_connections' not equal to 0.

12.3.8 Condition

Condition element defines when a severity will be triggered.

```
<!ELEMENT Condition (CategoryValue*, KeyColumn*, FixitJob? )>
<!ATTLIST Condition
CRITICAL CDATA #IMPLIED
WARNING CDATA #IMPLIED
OPERATOR (EQ | LT | GT | LE | GE | NE | CONTAINS | MATCH ) "GT"
OCCURRENCES CDATA "1"
NO_CLEAR_ON_NULL (TRUE | FALSE) "FALSE"
MESSAGE CDATA #IMPLIED
MESSAGE_NLSID CDATA #IMPLIED
COLUMN_NAME CDATA #REQUIRED
PUSH (TRUE | FALSE) "FALSE"
GENERATE_INIT_CLEAR (TRUE | FALSE) "FALSE"
ALERT_CONTEXT CDATA #IMPLIED
CLEAR_MESSAGE CDATA #IMPLIED
CLEAR_MESSAGE_NLSID CDATA #IMPLIED>
```

If it is Table Metric, MetricColumn is used to define which column and key to use to identify the row and column.

If KEYONLY_THRESHOLDS is set to TRUE for a Metric column, the Condition element must include a KeyColumn element.

12.3.8.1 Attributes

CRITICAL: Threshold. A special value, *"NotDefined"* for the threshold ensures that the result of the operation specified by the OPERATOR will fail.

WARNING: Threshold. "NotDefined" may also be applied to WARNING.

OPERATOR: Specifies the operation to evaluate the condition.

EQ: Equals

LT: Less than

GT (default): Greater than

LE: Less than or equal to

GE: Greater than or equal to

NE: Not equals

CONTAINS: Contains

MATCH: Matches

OCCURRENCES: The default value is 1.

NO_CLEAR_ON_NULL: This attribute is used to control severity clearing when a null value is returned for a metric column. It defaults to FALSE with the behavior that a null value ends up clearing previous alert severities. With a TRUE value for this attribute, null values will be skipped in severity evaluations without clearing the severity.

MESSAGE: The message attribute is a message template that will be used to generate message(s)to be sent along with the event occurrence. This message can contain the following place holders.

a) %value%: The value of the metric (or column of metric)

b) %target%: name of the target

c) %metric_id%: metric id

d) %column_name% This will be the value of any column this can include value columns as well as key columns

e) %warning_threshold%: the warning threshold of the condition

f) %critical_threshold%: the critical threshold of the condition

g) %num_of_occur%: number of occurrences

MESSAGE_NLSID: Specifies the String ID of the ResourceBundle for the message.

COLUMN_NAME: For table metric, COLUMN_NAME defines which column will be checked. KeyColumn will be used to identify a row.

PUSH: This attribute is used to distinguish conditions created for push-based alerts from conditions that are evaluated over the collected data. The agent does not evaluate PUSH="TRUE" conditions.

TRUE | FALSE (default)

GENERATE_INIT_CLEAR: This attribute can be used to override the agent's behavior of not generating a severity the very first time a CLEAR is generated. Set this to TRUE if you do want the initial CLEAR.

TRUE | FALSE (default)

Attributes introduced in **Enterprise Manager version 10.2**:

ALERT_CONTEXT: This attribute will be used to pass the related alert context. This new attribute may contain a list of column names separated by ";".

CLEAR_MESSAGE: Specifies a different message when an alert is cleared. If this attribute is missing then the MESSAGE attribute is used when alerts are cleared.

CLEAR_MESSAGE_NLSID: Specifies the NLSID for the clear message. If absent, the MESSAGE_NLSID is used when alerts are cleared.

12.3.8.2 Elements

CategoryValue (First introduced in Enterprise Manager version 10.2.)

KeyColumn

FixitJob

12.3.8.3 Used In

CollectionItem

MetricColl

12.3.8.4 Examples

Condition element may include optional CategoryValue element(s), KeyColumn element(s) and an optional FixitJob element.

If the result after the keys are applied contains more than one row, the event occurrence generated will have content/message for each row that has crossed the threshold.

MATCH is used for regular expression.

For example:

OPERATIOR="MATCH" CRITICAL=".*ORA.*ERR.*"

This statement will match a string containing both ORA and ERR such as "ORA-ERR 345".

CategoryValue sub tags are used to classify the Condition along two axis, CLASS and CATEGORY. For e.g. CLASS=Fruits, CATEGORY=RedFruits Categorization of Conditions is useful for Root Cause Analysis among other things.

```
<Condition COLUMN_NAME="alertSeverity"
CRITICAL="NotDefined"
WARNING="NotDefined"
OPERATOR="LE"
OCCURRENCES="1"
MESSAGE="%alertConcatString%"
NO_CLEAR_ON_NULL="TRUE"
MESSAGE_NLSID="host_alertLog_alertSeverity_cond"/>
<Condition COLUMN_NAME="State"
CRITICAL="open"
OPERATOR="EQ"
MESSAGE="%Description%"
ALERT_CONTEXT="In-contextLaunchURL"
NO_CLEAR_ON_NULL="TRUE" />
```

These are examples of the Condition element.

```
<TargetCollection TYPE="examplec1" >
<CollectionItem NAME = "Response">
<Schedule>
<IntervalSchedule INTERVAL = "300" TIME_UNIT = "Sec"/>
</Schedule>
<Condition COLUMN_NAME="Status"
CRITICAL="ok"
OPERATOR="NE"
OCCURRENCES="3"
```

```
MESSAGE="%target% adaptor is inaccesible or is connected."
MESSAGE_NLSID="network_response_status"/>
</CollectionItem>
</TargetCollection>
```

This sample gives the context for the Condition element.

For additional examples please refer to TargetCollection.

12.3.9 KeyColumn

The KeyColumn element is used to specify the key column for a table. It identifies a row of a table. This element must be present in a Condition element if KEYONLY_ THRESHOLDS attribute is set for a Metric column.

```
<!ELEMENT KeyColumn (#PCDATA)>
<!ATTLIST KeyColumn
COLUMN_NAME CDATA #REQUIRED
OPERATOR (EQ | LIKE) "EQ">
```

12.3.9.1 Attributes

COLUMN_NAME: Specifies name of the key column

Attributes introduced in **Enterprise Manager version 10.2**:

OPERATOR:

Supported values are:

a) EQ (default): Equals

b) LIKE: like

12.3.9.2 Elements

Character data

12.3.9.3 Used In

Condition

12.3.9.4 Examples

```
<Condition COLUMN_NAME="col3" WARNING="def" OPERATOR="CONTAINS">
<KeyColumn COLUMN_NAME="col1">keyA</KeyColumn>
</Condition>
```

12.3.10 FixitJob

This element describes the action to be taken in response to an alert.

```
<!ELEMENT FixitJob (Property*) >
<!ATTLIST FixitJob
TYPE CDATA #IMPLIED >
```

12.3.10.1 Attributes

TYPE: Specifies the type of the job.

12.3.10.2 Elements

Property

12.3.10.3 Used In

Condition

12.3.10.4 Examples

```
<TargetCollection TYPE="examplec1" >
<CollectionItem NAME = "FixitExample" UPLOAD_ON_FETCH="TRUE">
<Schedule>
<IntervalSchedule INTERVAL="60" TIME_UNIT="Sec" />
</Schedule>
<MetricColl NAME="metric1"/>
<Condition COLUMN_NAME="col2" CRITICAL="0" OPERATOR="GT" OCCURRENCES="1">
<FixitJob TYPE="OSCommand">
<Property NAME="prop_emdurl" SCOPE="SYSTEMGLOBAL">EMD_URL</Property>
<Property NAME="prop_env" SCOPE="ENV">EMDROOT</Property>
<Property NAME="prop_loc" SCOPE="INSTANCE">FILE_LOC</Property>
<Property NAME="COMMAND" SCOPE="GLOBAL">rm %prop_loc %</Property>
</FixitJob>
</Condition>
</CollectionItem>
</TargetCollection>
```

This example illustrates a simple FixitJob that deletes files in response to a condition. The COMMAND property specifies the command that is executed when the value in col2 of the metric triggers the condition.

Index

A

C

D

E

F

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y