Oracle® Enterprise Manager

Cloud Control Extensibility Programmer's Guide

13c Release 4
F23289-01
May 2020

ORACLE

Oracle Enterprise Manager Cloud Control Extensibility Programmer's Guide, 13c Release 4

F23289-01

Contents

3 About Managed Targets

4 About Enterprise Configuration Management

5 Using Derived Associations

10 Understanding Compliance Standards

11 Understanding Software Library

Index

Preface

This document provides a brief overview of the Enterprise Manager product and its architecture and then describes each of the subsystems that provide interfaces for extending the product features for a new or customized target type.

Each section provides an overview of the particular subsystem and describes the features it exposes to Enterprise Manager end-users, and the ways in which plug-in developers may leverage the provided interfaces to support those features for new target types being added to Enterprise Manager.

This document is not a reference guide and as such does not include details of these interfaces including API details. Refer to the *Oracle Enterprise Manager Cloud Control Extensibility Reference Guide* and individual API Documentation for those details.

> **Note:**
>
> For the most current version of this document, go to the **Extensibility** page of the Oracle Enterprise Manager Online Documentation set: https://docs.oracle.com/en/enterprise-manager/cloud-control/enterprise-manager-cloud-control/13.4/index.html

Audience

This document is intended for developers that want to extend Oracle Enterprise Manager to support the ability to manage custom target types or extend the manageability of out-of-box target types.This document assumes basic knowledge of Enterprise Manager and its core features and concepts.

Documentation Accessibility

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

Access to Oracle Support

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

Related Documents

For more information about Oracle Enterprise Manager, see the Oracle Enterprise Manager Online Documentation set: https://docs.oracle.com/en/enterprise-manager/index.html.

Conventions

The following text conventions are used in this document:

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

1

Introduction to Extending Enterprise Manager

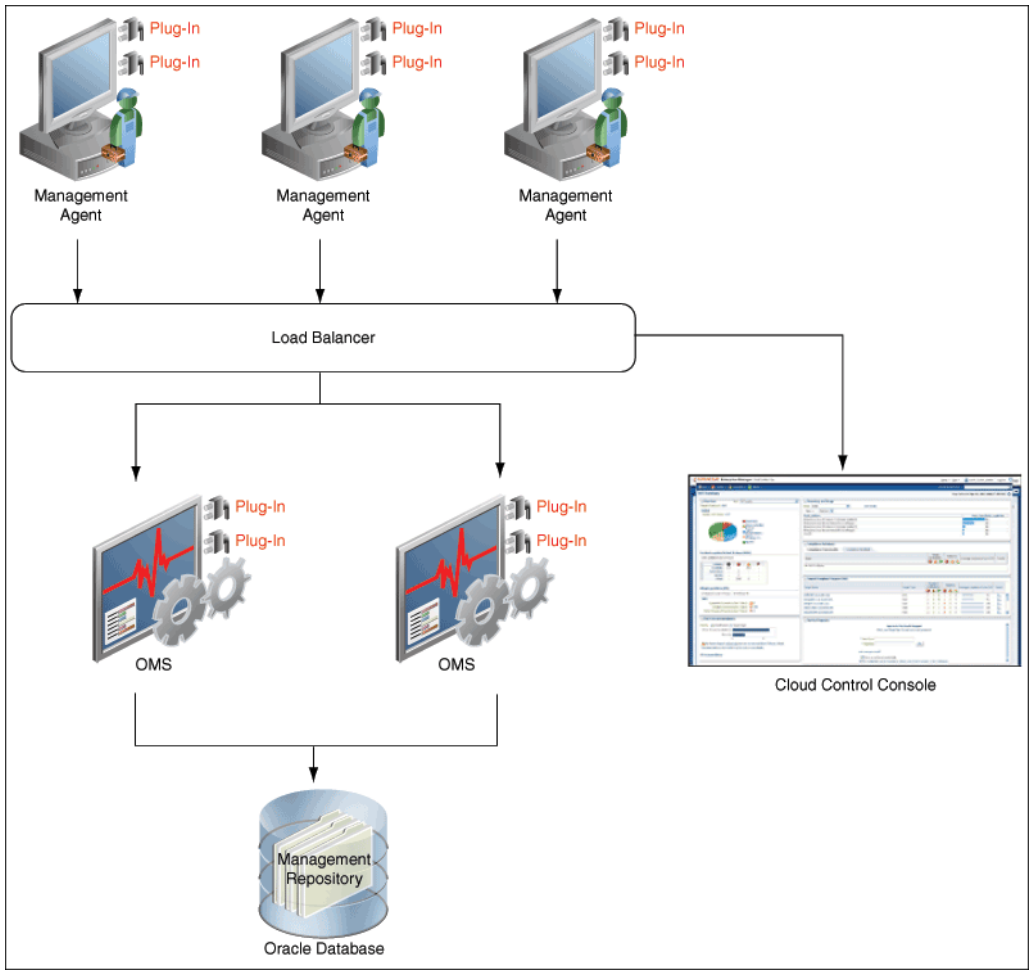
Oracle Enterprise Manager is Oracle's suite of management products for managing various Oracle and non-Oracle technologies. The name Enterprise Manager actually refers not to a single product, but to a portfolio of products. At the top level of the product portfolio, Oracle Enterprise Manager is made up of a set of consoles for managing various technologies including Oracle Database, Oracle Fusion Middleware and Oracle Fusion Applications.

This chapter contains the following sections:

Overview of the Oracle Enterprise Manager Platform

The foundation of Cloud Control is a lightweight, multi-tiered, extensible platform for building management tools. The framework is built on the Fusion Middleware platform. The three main components of the Cloud Control platform are:

Figure 1-1 Enterprise Manager Architecture



Oracle Management Service

The Oracle Management Service (OMS) provides the services used to coordinate the storage of management information and the automation of management activities for all entities managed in the network. It also includes facilities for serving the web-based user interface, which is the Cloud Control console.

Management Repository

The Management Repository is the Oracle Database that stores all important management information for the entities managed by Enterprise Manager. The Oracle Management Service (OMS) uses the Management Repository to store and retrieve key information, such as monitoring data.

Management Agent

The Management Agent is the lightweight process that acts as a proxy for Enterprise Manager on the various hosts in the network where entities that Enterprise Manager manages are located. It communicates with the OMS to collect and deliver monitoring

information and to coordinate management activities executed against the management entities.

Interfaces to Enterprise Manager

The following tools and applications are available to access Enterprise Manager:

- Cloud Control Console

  The Cloud Control Console provides the user interface that presents management content to the user for monitoring, administration, or enterprise configuration.

  For more information, see Cloud Control Console.

- EM CLI

  The Enterprise Manager Command Line Interface (EM CLI) enables you to access Enterprise Manager Cloud Control functionality from text-based consoles (shells and command windows) for a variety of operating systems.

  For more information, see the *Oracle Enterprise Manager Command Line Interface* at the **Extensibility** page of the Oracle Enterprise Manager Online Documentation set.

- Management Repository Views

  The Management Repository views provide access to target, metric, and monitoring information stored in the Management Repository.

  For more information, see the "Using Management Repository Views" section of the *Oracle Enterprise Manager Cloud Control Management Repository Views Reference* at the **Extensibility** page of the Oracle Enterprise Manager Online Documentation set.

- Cloud Web Service APIs

  The Cloud APIs can be used to integrate Enterprise Manager with custom-built or 3rd party self service consoles and service desks.

  For more information, see Using the Cloud APIs from the *Oracle Enterprise Manager Cloud Administration Guide* of the Oracle Enterprise Manager Online Documentation set.

About the Oracle Management Service

The Oracle Management Service (OMS) tier is a Java EE (Enterprise Edition web applications that can be divided into several major components:

- A management console for performing management and administrative functions
- A Management Repository where information collected by the Management Agents from managed targets is consolidated
- Management administration and maintenance services

The management service tier can be further distributed in high-end environments for performance. For example, you can install the Management Repository on a separate host from the host running the Management Service. The framework can also be collapsed into the managed target tier to support a stand-alone deployment configuration (Management Repository, Management Service, and Management Agent residing on a single host).

In a typical configuration, the Oracle Management Service (OMS) resides on a separate host from the managed targets. The infrastructure tiers can be collapsed onto a single host for small deployments for enterprises where central management is not required.

The OMS tier of the management infrastructure includes the management consoles that are used for management operations such as monitoring, administration, configuration, central policy setting, and security.

Cloud Control Console

The Cloud Control Console provides the user interface that presents management content to the user for monitoring, administration, or enterprise configuration.

The Cloud Control Console uses the Enterprise Manager Services to display management content to the user. Administrators, managers, or developers can see views of the management information that is abstracted to best satisfy their requirements. Interface customization controls what information is displayed as well as the operations that may be performed by a particular user. For example, administration functions such as database shutdown and startup might not be available to an upper level manager, yet the manager could view the availability status of the server.

About the Oracle Management Agent

The Oracle Management Agent identifies targets, collects data about those targets, and detects problems in your environment (such as high CPU usage). A typical management framework deployment has one Management Agent on each host that is part of the enterprise.

The Management Agent is responsible for the managed targets that are running on that host. A target, or more specifically, a target instance, can be defined as any entity that can be monitored within an enterprise. This entity can be an application running on a server, the server itself, the network, or any of its constituent parts.

To store and process the information collected by the Management Agent, and to instruct the Management Agent to perform administrative tasks, the Management Agent relies on the part of Enterprise Manager that provides the core functionality of the framework, that is, the Oracle Management Service.

The Management Agent coordinates management activities on a host. In a typical configuration, one Management Agent will be running on the host. It performs management tasks for any targets on the host system. The Management Agent is responsible for:

- Executing management tasks
- Gathering and transferring metric data
- Detecting alert thresholds (warning and critical)

About the Oracle Management Repository

The Oracle Management Repository (Management Repository) is the storage location where all the information collected by the Management Agent gets stored. The Management Repository consists of schema definitions, database jobs, and stored

procedures running inside an Oracle database. The information in the Management Repository includes:

- Configuration information about the managed targets
- Historical metric data and alert information
- Client and web server response time information
- Managed target availability information
- Product and patch inventory information

The information stored in the Management Repository is useful for tasks such as end-to-end reporting and problem diagnosis, as well as service level agreement and availability reporting. Information stored in the Management Repository can be shared between any number of administrators accessing Central Consoles that point to the central Management Repository.

The Management Repository is the comprehensive source for all management information for targets that are managed through the OMS. The Management Repository is designed as an open schema. This allows users of Oracle's management infrastructure to customize how the information in the repository is used when the default capabilities are not sufficient to satisfy their business requirements.

If you are using Enterprise Manager's partner Extensibility Development Kit (EDK), then you cannot modify the Management Repository directly. You can access information available from the repository's public views in a number of areas, including report formation, association derivation, compliance rule evaluation, and the display of management information in a customized UI built with the plug-in. For more information, see the relevant sections of this guide.

About Metadata Plug-ins

A metadata plug-in (plug-in) extends the ability of Enterprise Manager to manage and monitor a specific type of target. The plug-in instructs the Management Agent on how to collect metric data for the target, and instructs Oracle Management Service (OMS) on what to do with the collected data.

A plug-in consists of a set of metadata files that serve specific functions at different tiers within the Enterprise Manager framework. For example, the target type metadata file is an integral part of defining a new target type. The EDK requires the target type metadata file, in addition to a default collection file, to create a new plug-in within the plug-in archive.

When you introduce a new target type to Cloud Control, the management features included in Enterprise Manager are automatically supported for the new targets by default. This includes core management features such as:

- Metric Collection and Alerts
- Availability and Blackouts
- Groups and Systems
- Default System Reports
- Other Common Console Features

You can also choose to enable additional management features for your target types by using application programming interfaces (APIs) supported by the EDK. These features include:

- Automated Discovery
- Target-to-Target Associations
- Configuration Management
- Custom Target Credentials
- Automation (or Jobs)
- Custom Reports
- Compliance Rules and Standards
- Custom User Interface for Management

Each of these management features requires additional metadata files, and possibly the creation of scripts, that must be packaged with the plug-in.

2

Getting Started With Enterprise Manager Plug-ins

This chapter provides an introduction to Enterprise Manager metadata plug-ins and the Extensibility Development Kit (EDK).

This chapter contains the following sections:

What is a Plug-in?

A plug-in is a group of files (such as target definition files, collection scripts to collect metrics from targets, and any custom UI components to customize user interfaces) that has been added to a plug-in archive using the Enterprise Manager Extensibility Development Kit (EDK). The plug-in files must be added to an archive before they become a plug-in officially. A plug-in archive file (*.opar) associates the files together as an official plug-in.

Each plug-in defines a new type or types of target that can be monitored by Enterprise Manager. A target, or more specifically, a target instance, can be defined as any entity that can be monitored within an enterprise. This 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 enabling you to add new target instances to the management framework from the Enterprise Manager console. Then you can take advantage of Enterprise Manager's monitoring and administrative features.

A plug-in consists of several types of files that serve specific functions at different tiers within the Enterprise Manager framework. For example, the target type metadata file is an integral part of defining a new target type. The EDK requires the target type metadata file, in addition to a default collection file, to create a new plug-in within the plug-in archive.

To create a new plug-in, you must have the following (minimum requirement):

- Plug-in metadata
- Target-type metadata

- Default collection metadata

When you use plug-ins to define custom target types for monitoring by Enterprise Manager, you can centralize all of your management information in the console.

When you introduce a new target type to Cloud Control, the management features included in Enterprise Manager are automatically supported for the new targets by default. This includes core management features such as:

- Metric Collection and Alerts
- Availability and Blackouts
- Groups and Systems
- Default System Reports
- Other Common Console Features

You can also choose to enable additional management features for your target types by using application programming interfaces (APIs) supported by the EDK. These features include:

- Automated Discovery
- Target-to-Target Associations
- Configuration Management
- Custom Target Credentials
- Automation (or Jobs)
- Custom Reports
- Compliance Rules and Standards
- Custom User Interface for Management

Each of these management features requires additional metadata files, and possibly the creation of scripts, that must be packaged with the plug-in.

What's New in Enterprise Manager Plug-ins?

This section describes the new features of Enterprise Manager plug-ins:

- Plug-in Deprecation and Obsolescence

  This release provides an EDK to support the development of custom plug-ins for managing target types not supported out of the box.

About the Extensibility Development Kit (EDK)

A key component of the Enterprise Manager Cloud Control architecture is the Extensibility framework. To enable Oracle partners to extend the Enterprise Manager platform, an Extensibility Development Kit (EDK) is provided with the product.

The EDK is a collection of tools, utilities, and documentation, including:

- Enterprise Manager Extensibility documentation: Provides general guidelines for programming Enterprise Manager plug-ins
- Reference Implementation: Provides a reference code implementation, code snippets, and so on for various Enterprise Manager features

- Build time tools to verify EDK conformance: A tool that you can use to validate and report any violations, with respect to Enterprise Manager Extensibility guidelines
- Packaging Tool: A tool to package the plug-in components tool (`empdk`)
- Verification Tool: A tool to validate plug-in code components and to report violations (if any).

The EDK includes a command line utility called `empdk`. Use this utility to package or validate a plug-in archive. For information about the `empdk` commands and their options, see the *Oracle Enterprise Manager Extensibility Programmer's Reference*.

After you download the EDK, unpackage it on your local system, and change your current directory to the location where you unpacked the EDK. The EDK contains reference documentation and guides to help you with plug-in development as well as the API reference you might need to integrate while developing plug-ins.

For information about downloading the EDK, see Installing the Extensibility Development Kit (EDK).

Contents of the EDK

The EDK archive contains the following directories:

- \bin

  Contains the `empdk` utility, which you use to:

  - Validate the structure of your plug-in
  - Package your plug-in

- \doc

  Contains the Oracle Enterprise Manager Extensibility Programmer's Guide and Programmer's Reference, as well as the EDK API Reference documentation, including documentation on Management Views. Review overview.html for links to the documentation provided.

  You can also access the EDK API Reference documentation directly through its index page (sdk_api_ref.html).

- \lib

  Contains internal libraries used by the EDK.

- \oui

  Contains internal libraries used by the EDK.

- \samples

  Contains a complete reference implementation of a plug-in, packaged as demo_hostsample.zip. The sample metadata files included should be used as examples of the files referenced throughout the EDK documentation.

  View the README packaged with the archive for instructions on building, deploying, and using the sample plug-in.

Other utilities referenced in this documentation, including EM CLI and EMCTL, are installed with Enterprise Manager and are typically deployed to the Oracle Management Service (OMS) host.

Installing the Extensibility Development Kit (EDK)

> **Note:**
>
> Before installing the EDK, you must have the following:
>
> - Latest version of the EDK ZIP archive from the Self Update console. (To access the Self Update console, from the Cloud Control console, select **Setup**, then **Extensibility**, and then **Self Update**.)
> - Java version 1.7.0_51 or later
> - Local system running Solaris, Linux, HP-UX, AIX, or Windows with New Technology File System (NTFS)

To install the EDK:

1. Download the EDK ZIP archive to your local system using one of the following options:
   - Using the Enterprise Manager Cloud Control console
     a. Log in to Enterprise Manager Cloud Control.
     b. From the **Setup** menu, select **Extensibility**, then select **Development Kit**.

        The Extensibility Development Kit (EDK) page appears.
     c. Under Installing the EDK, select **Download the Extensibility Development Kit to your workstation**.
     d. Save 13.1.0.0.0_edk_partner.zip to your local system.
   - Using the Enterprise Manager Command Line Utility (EM CLI), open a command prompt and run the following command:

     ```
     emcli get_ext_dev_kit
     ```

     This command downloads the EDK ZIP archive to the same directory from where you ran the command and does not require any parameters.

     > **Note:**
     >
     > For information about setting up EM CLI, see the *Oracle Enterprise Manager Cloud Control Extensibility Reference Guide*.
2. Set your JAVA_HOME environment variable and ensure that it is part of your PATH. For example:

   ```
   setenv JAVA_HOME /usr/local/packages/j2sdk1.7.0_51

   setenv PATH $JAVA_HOME/bin:$PATH
   ```
3. Unpackage the downloaded EDK ZIP archive to a directory on your local system. For example:

```
Unzip 13.1.0.0.0_edk_partner.zip
```

This command creates the following directories under the directory (*release_edk_partner*) where you unpackaged the EDK ZIP archive:

```
release_edk_partner
                   |
                    bin
                    doc
                    lib
                    oui
                    samples
                    README
```

For more information about the directory contents, see Contents of the EDK.

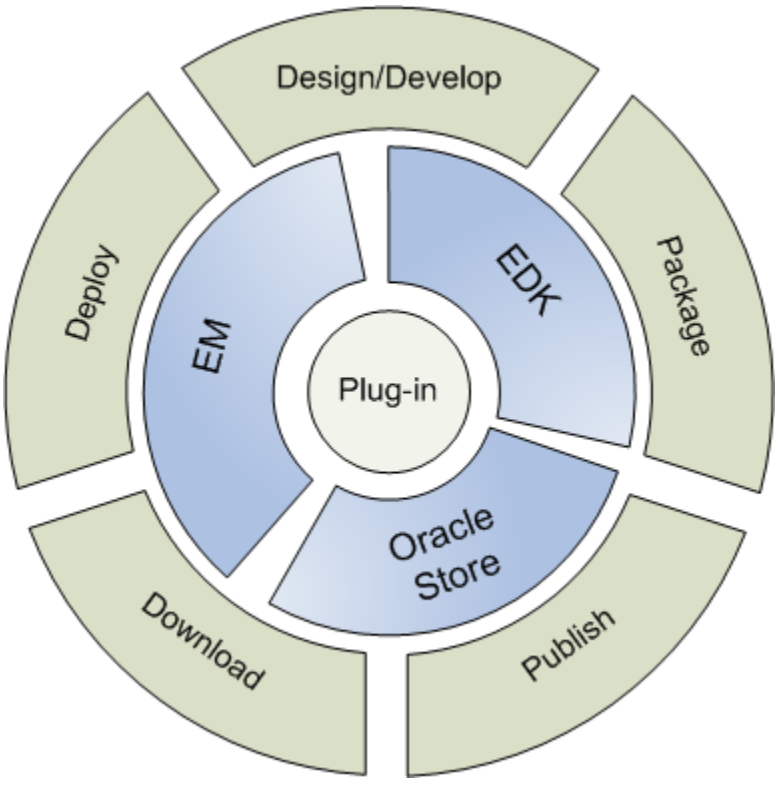
Plug-in Development Lifecycle

Developing a plug-in involves:

1. Designing the set of entities to be included in the plug-in
2. Defining the metadata that describes them to Enterprise Manager
3. Packaging and delivering the plug-in so that users can deploy and use it in Enterprise Manager installations

Figure 2-1 depicts the lifecycle of a plug-in.

Figure 2-1 Lifecycle of a Plug-in



The plug-in lifecycle represents the transition of a plug-in through the following stages:

- Design/Develop: Designs and develops the plug-in
- Package: Packages and releases the plug-in
- Publish: Publishes the plug-in at the Oracle Enterprise Manager Store
- Download: Enterprise Manager downloads plug-ins from Oracle Enterprise Manager Store
- Deploy: Deploys the plug-in to Oracle Management Service (OMS) and Management Agents
- Upgrade: Upgrades the plug-in to a later version (download and deploy)
- Undeploy: Removes the plug-in completely from OMS

Designing Your Plug-in Metadata

The first step in developing your plug-in is to design the contents of your plug-in. At the highest level, this includes deciding what set of target types will be included in the plug-in (a plug-in may include management metadata for one or more target types) and defining what set of management capabilities you want to support for each target type. (The complete available list is included in Plug-in Contents and Packaging).

Before creating plug-in files, you must define which parameters of the target type are required to monitor and manage your new component accurately. 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. Whenever a threshold is crossed, Enterprise Manager raises an incident, informing administrators of potential problems.

These metrics and collection settings are the required components of every plug-in. Additional capabilities such as job definitions, custom User Interface, and so on, are optional. Each feature to be included in the plug-in requires additional thought and design to determine what aspects of the target should be manageable for those features.

Some of the key features to consider when designing your plug-in include the following:

- Administrative features that are necessary to manage the target, such as administrative tasks or larger, and more complex jobs that may require automation support (for example, scheduling)
- Configuration data to be used for managing the target including associations between the target and other targets in network
- Reports and customized user interfaces to be provided for end-users to manage the targets

This is not a complete list and the information for integrating with each subsystem is summarized in this guide and provided in detail in the *Oracle Enterprise Manager Cloud Control Extensibility Reference Guide*.

Plug-in Contents and Packaging

The plug-in contents and packaging section includes the following topics:

Plug-in Metadata

Each plug-in includes the plug-in metadata, which is defined in XML files that represent all the information about a plug-in.

Plug-in metadata is used during plug-in deployment. It contains properties that identify the plug-in, such as name and version, and declares the set of target types that will be added to Enterprise Manager Cloud Control.

Metadata Services

Most of the Enterprise Manager features exposed to you through the Enterprise Manager EDK require the construction of metadata files (in XML). These metadata files define the items that are integrated for that target type for the particular subsystem. For example, a plug-in might contain a metadata XML file that defines the job types that are available for that target type. After deployment, the jobs will be available to be submitted or executed for any instance of that target type.

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.

In addition to the metrics to be collected for the target type, the target type metadata includes the target properties and credentials information. The target properties are the key set of properties that help to define the target. These properties are typically static though Enterprise Manager does support the ability to define dynamic instance properties that are evaluated by the Management Agent. For information about defining dynamic instance properties, refer to the *Oracle Enterprise Manager Cloud Control Extensibility Reference Guide*.

The plug-in includes a default collections file, which defines the frequency at which metrics and configuration data will be collected

The Enterprise Manager Extensibility Framework provides the support for packaging these metadata files in the EDK as well as for deploying these files to the appropriate subsystem within Enterprise Manager. This may include deploying information to the Management Server, or to the Management Agent, or both in some cases. For more information about packaging metadata files in the EDK, see Plug-in Packaging Structure.

In certain instances, the metadata includes references to other components packaged with the plug-in. For example, a job type definition might include a step that executes a Perl script against one of the targets to perform an administrative task. In this case, the job type definition is declared in the metadata and included in the Management Server

part of the plug-in while the script is included in the Management Agent part of the plug-in.

For a complete list of all metadata services available and the detailed documentation describing the use of each service, see the *Oracle Enterprise Manager Cloud Control Extensibility Reference Guide*.

Plug-in Packaging Structure

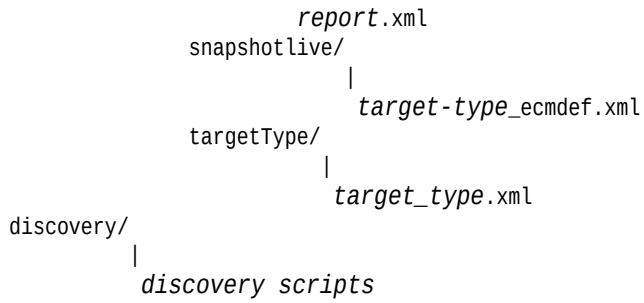
The final step in developing a plug-in is to package it into a plug-in archive. This step takes a staging directory (*plugin_stage*) as input where the files that comprise the plug-in are located. Example 2-1 summarizes the directory structure and the location for the files.

For information about the contents of the plug-in directory, see the *Oracle Enterprise Manager Cloud Control Extensibility Reference Guide*.

Example 2-1 Plug-in Directory Structure

```
plugin_stage/
            |
             plugin.xml
             agent/
                   |
                    plugin_registry.xml
                    default_collection/
                                       |
                                        target_type.xml
                    metadata/
                             |
                              target_type.xml
                    scripts/
                            |
                             scripts
             oms/
                 |
                  metadata/
                           |
                           default_collection/
                                             |
                                              target_type.xml
                           derivedAssoc/
                                        |
                                         derivedAssoc_rule.xml
                           discovery/
                                     |
                                      discovery.xml
                           gccompliance/
                                        |

compliance_rule.xml
                           jobTypes/
                                    |
                                     job_type.xml
                           mpcui/
                                 |
                                  mpcui.xml
                           reports/
                                   |
```



Releasing a Plug-in

After you have packaged the plug-in, you can distribute the plug-in archive (OPAR) file to the Enterprise Manager administrators that require support for the target types included in that plug-in. In this mode, Oracle does not certify the quality or safety of the plug-in for use in a production environment and appropriate evaluation in a test environment is strongly encouraged before deploying a plug-in that is not certified by Oracle.

To make the plug-in available for deployment, the Enterprise Manager administrator must import the plug-in archive into the Enterprise Manager installation. After this is complete, the plug-in will be available to be deployed in the same way as any other plug-in downloaded from Oracle. For information the plug-in import command, see the *Oracle Enterprise Manager Cloud Control Extensibility Reference Guide*.

Deploying a Plug-in

After a plug-in has been imported into Enterprise Manager, the Enterprise Manager administrator must explicitly deploy the plug-in to the Management server and to the appropriate Management Agents.

Plug-in deployment can be accomplished through the Enterprise Manager command line facility (EM CLI) or by selecting **Extensibility** from the **Setup** menu in the Enterprise Manager console. For information about the use of these options, see the *Oracle Enterprise Manager Cloud Control Extensibility Reference Guide*.

Automated Deployment During Discovery

If a plug-in includes support for automated discovery, Enterprise Manager deploys the plug-in to the appropriate Management Agent automatically as targets are discovered and marked as manageable targets (promoted) in Enterprise Manager. To enable automated discovery, you must include the appropriate discovery metadata in the plug-in.

For plug-ins that do not include support for the automated discovery of targets, the Enterprise Manager administrator must add targets to the Enterprise Manager console manually, specifying the Management Agents where the targets are located. In this case, the Enterprise Manager administrator must deploy the plug-in to those Management Agents before manually adding the target.

Plug-in Upgrade

As new versions of a plug-in are released, the Enterprise Manager administrator can import the new plug-in by using the self-update feature of Enterprise Manager to download the new version from the Oracle Store, or by importing the new plug-in archive if it is a private distribution.

At this point, the new plug-in version is available in Enterprise Manager but not deployed to any of the Management Servers or Agents. The Enterprise Manager administrator must explicitly deploy the new plug-in version using the same process for deployment described in Deploying a Plug-in.

Undeploying Plug-ins

If a plug-in is not in use anymore, you can remove it from Enterprise Manager. Undeploying a plug-in removes all elements of the plug-in from the Management Server and from the Management Agents where the plug-in was deployed previously.

Deprecating a Plug-in

Due to various business and technical reasons, sometimes released plug-ins must declare end-of-life and support for these plug-ins discontinues going forward.

When you deprecate a plug-in, you are announcing the end of life of the plug-in in advance and support will discontinue from the next major Enterprise Manager platform release. When support discontinues in the next major release, the plug-in becomes obsolete.

For example, if you deprecate a plug-in in Release 13.1.0.0, then the deprecated plug-in will have the same level of support in Release 13.1.0.1. It will not become obsolete until the next major release, such as Release 14.1. For information about deprecating plug-ins, see the "Deprecating a Plug-in" section in the *Enterprise Manager Cloud Control Extensibility Programmer's Reference*.

> **Note:**
>
> You cannot make a plug-in obsolete directly. You must mark it as deprecated in an earlier patchset release.

3

About Managed Targets

This chapter provides information about managed targets and contains the following sections:

Introduction to Managed Targets

Each 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 entity that can be monitored within an enterprise. Managed targets are the entities that Enterprise Manager can monitor and manage. Examples of targets include hosts, databases, application servers, applications, and listeners. As your environment changes, you can add and remove targets from Enterprise Manager as required.

Many of the commonly-used managed targets have been defined as part of the base Enterprise Manager product. They are preconfigured for management automatically when a management-ready product is installed. Oracle applications, Oracle databases and applications servers, and many of the operating systems that run Oracle products are all classified as management-ready targets.

Even though a target is predefined - for instance, monitoring levels, thresholds, and notification rules - you can still customize Enterprise Manager as required to meet business requirements. That is, you can perform value-added instrumentation to access more of the rich management functionality of Enterprise Manager than is provided with the standard target configuration.

Managed targets include:

- Applications that require management
- Separately configurable or controllable subsystems of an application
- Management components of the underlying application hardware topology

About the Target Model

The target model consists of many different types of entities, all of which are modeled as targets to derive the benefit of the security provided by the security infrastructure for targets. For example, systems, services, and groups are all modeled as targets. The line of distinction between the various entities has become blurred over time and has been subsumed into the overall notion of target. So a target could mean different things to different plug-in developers.

To bring distinction and clarity into what kind of entity is actually being modeled, Enterprise Manager employs the concept of high level classes called manageable entity (ME) classes into which each of the entities falls. Each of the classes has a well-

defined definition and capabilities. By looking into definitions, you should be able to tell which class the entity being modeled falls under.

For example, today redundancy groups are commonly mistaken for groups, whereas redundancy groups are systems. By following the definition of group and system, it should be clear under which class it falls.

Manageable Entity (ME)

In the Enterprise Manager context, a manageable entity is an entity that Enterprise Manager is capable of managing. This implies that the entity is exposed in some form to end users in the Cloud Control application, and has well-defined attributes and semantics.

There are several classes of manageable entities in Enterprise Manager. Each manageable entity class has the following characteristics:

- Definition: Specifies the rules and inherent attributes of the entity class
- Representation: Deals with how entities of that class are represented in the Enterprise Manager data model and exposed to end users
- Enterprise Manager Capabilities: Features and capabilities that Enterprise Manager provides to entities of that class

All manageable entities have the following common capabilities in Enterprise Manager. The capabilities listed under each ME are in addition to the common capabilities.

- They are protected by Enterprise Manager security model.
- They can all participate in relationships (associations) with other MEs. There are some restrictions that are noted alongside the class.
- They have unique identification
- Properties (name-value pairs) can be attached to the ME

A manageable entity in Enterprise Manager falls into one of the following classes only. Additional classes can be added in future releases. If an entity does not fall into any of the classes, then it is not a manageable entity or it is a new class of ME that requires support.

- Managed Targets
- Services
- Systems
- Groups
- Target components

A manageable entity can be in multiple states, as outlined below.

- Managed state or Not-Yet Managed (NYM) state

  Initially when a target is discovered, it is loaded into the Management Repository as an NYM entity. In this state, the entity can have associations but is not managed by a Management Agent yet. The user can go to the discovery results page and assign them to a Management Agent along with required credentials. The entity then goes to managed state. It is possible also to manually initiate target discovery from the Enterprise Manager UI and retrieve or provide all

necessary properties of a target and save it as a managed target directly. Automatic discovery is one use case where NYM targets come to existence.

- Existence only state

  A discovered entity that Oracle does not support will have an existence only state. It is similar to an NYM entity except that it cannot be managed by Oracle.

  When the plug-in developer registers this target type, they must add the `is_existence` type property to the target type metadata file. For more information about type properties and the target type metadata file, see the Creating Target Metadata Files from *Cloud Control Extensibility Programmer's Reference* on the **Extensibility** page of the Oracle Enterprise Manager Online Documentation set.

> **Note:**
>
> Plug-in developers using the EDK can define new target types that are managed within Enterprise Manager, but this is limited to Managed Targets and Target Components.The ability to define new install home, service, system, or group types is not supported as part of the EDK.
>
> The definition and capabilities of each of the classes are explained in the following sections.

Managed Target

A managed target is a manageable entity in Enterprise Manager that satisfies all the following conditions inherently (and is worth modeling). These are native to the entity and not derived from being represented in Enterprise Manager.

- Availability (Up/Down Status).
- Configuration attributes that can be collected
- Performance attributes that can be measured such as response time

Hosts, databases, listeners, and so on, are examples of managed targets.

The target type can be registered using target metadata XML described in Target Identity.

Target Identity

In this release of Enterprise Manager, you can rename a target to a new name without loss of history. You should not store an Enterprise Manager target name with any external data associated with the target that might be used later to locate the target in Enterprise Manager.

Lifecycle Status

The lifecycle status property is set by the end user to one of the following values:

- Development
- Test
- Release

- Production

For example, this property can be used in the priority processing of events. You do not have to do anything to make use of this property.

Groups

A group is a collection of manageable entities that allows end users to manage many MEs as a single logical unit. There are no required associations between members of the group – thus, members of a group may or may not have inherent relationships among themselves and with the group containing them. The group will have a *contains* association with its members.

Users decide membership in the group (direct addition or by criteria), so do not make any assumption on the composition of the group at design time.

Groups can be assembled by end users in an ad-hoc manner or by specifying specific business criteria (for example, by Line of Business, test versus production, target version, and so on).

There are two different types of groups from privilege perspective:

- Normal group

  Only view privilege on the group is propagated to the members.
- Privilege Propagating group

  Any privilege on the group is propagated to the members.

Systems

A system is a collection of inherently related manageable entities, which together provide one or more business functions or services.

The members of a system have well-defined relationships. These relationships are specified by associations.

The main difference between a system and a group is that a system is a collection of inherently related entities while groups are created mainly to manage many entities as one. Systems cannot contain groups; however, groups can contain systems.

System State

A system is fully formed if all of its underlying targets and the required associations have been discovered. If any of the required target and associations has not been discovered, then the system is partly formed. If the system cannot be created due to underlying logic problems, then the system is broken.

System operations are possible only on fully formed systems. System operations can be done on partially formed systems. It is your decision to support operations on partly-formed systems.

Availability will be computed for fully formed systems only. Charts and topology can be viewed for partly formed or broken systems. Root Cause Analysis (RCA) and other diagnostic utilities can work reliably on fully formed systems only.

Composite Targets

A composite target is a target that is composed of a number of related targets that are managed as a group. The related targets are often referred to as children or members of the composite target. As part of your plug-in, you will define the different target types that describe the composite target itself as well as its children target types

The composite target is a natural group of targets, the semantics of which are described in the plug-in itself. Enterprise Manager does not assign any additional semantics to the composite target other than the ability to represent the relationships between the composite and its children visually, either in the target navigator or in the topology view of the composite target. Any other semantics are assumed to be enforced with the plug-in code, either as an aggregation of data into composite metrics, associations between members, or operations (tasks or jobs) that span the composite children.

One typical composite example is that of a redundancy group. In this situation, a series of related targets form a group that is managed as a single composite entity. Each member of the group can also be managed separately as well as part of the group (composite) itself. The specific semantics of how the group operates, such as how failover occurs, how monitoring information is aggregated, and so on, are part of the logic defined within the plug-in. Enterprise Manager does not attempt to infer additional semantics from the composite definition, but it can display the set of members of the composite together and provide services of managing the associations between the members and other targets, associations between the members and the composite target itself and for retrieving membership or association details about the composite target or its members.

System Targets

In addition to composite targets, this release of the EDK adds support for the definition of a system target type as part of a metadata plug-in. While a composite target allows you to define a set of related targets that should be managed as a group, a system target type includes support for additional semantics provided by Enterprise Manager.

System targets are the basis for defining monitored services, which are the components of applications that run on the IT infrastructure. The IT infrastructure is modeled as a series of systems on which the services run. As such, Enterprise Manager supports the ability to view and monitor a system, and perform operations such as Root Cause Analysis of service failures for services that run on the system.

Services

A service models the access point of a business function offered by a target or system. A service can be associated with zero or one system. A service can use beacons or system components to compute availability and performance data.

For example, the e-mail service in the Beehive Application System uses a beacon transaction, which uses the SMTP protocol to check e-mail availability. Alternatively, service availability for a end user system can be defined by defining the key components of the system on which the service depends and then specifying the condition *ALL key components up* or *ANY key component up*.

A remote web service is an example of a service not associated with a system that Enterprise Manager manages. Enterprise Manager can still monitor the service using a beacon transaction to checks its availability and performance even though the underlying system is an infrastructure that is not known to or otherwise managed by Enterprise Manager.

This allows Enterprise Manager users to include remote services in the topology of their applications and to include the monitoring of those services as part of the application monitoring solution.

Management Capabilities Supported

The following table represents the type of capabilities that are supported for each of the various entity types that Enterprise Manager supports. *Members only* means that the operation is on the members of the ME and not on the ME itself. For example, Jobs can be run against members of the group but not on the group itself even though the job is submitted against the group.

Capability	Target	NYM	Group	Systems (Discovered)	Systems (user defined)	Service
Availability (up/down status)	Y	N	N	Y (optional)	N	Y
Performance Metrics (collected and rollup)	Y	N	Rollup only	Y	Rollup only	Y
Configuration Collections (save/ compare operations)	Y	N	N	Y	Members only	Y
Compliance Rules and Standards	Y	N	Y	Y	Y	Y
Can participate in associations	Y	Y	Restricted to contains with members	Y	User-defined only	Y
Assoc Derivation	Y	N	N	Y	N	Y
Jobs	Y	N	Members only	Y	Members only	Y
Events	Y	N	Members only	Y	Members only	Y
Patching, Provisioning	Y	N	Members only	Y	Members only	Y
Blackouts	Y	N	Members only	Y	Members only	Y
Templates	Y	N	Members only	Y	Members only	Y
Automatic Discovery (entity can be discovered or constructed automatically by Enterprise Manager)	Y	Y	N	Y	N	Y
Privileges	Y	Y	Y	Y	Y	Y
Privilege Propagation (propagate privileges to members)	N	N	Y (optional)	Y (optional)	Y (View only)	N

Capability	Target	NYM	Group	Systems (Discovered)	Systems (user defined)	Service
Metric Extensions	Y	N	N	Y	N	Y

4

About Enterprise Configuration Management

This chapter provides an overview of Enterprise Configuration Management and the Enterprise Configuration Management framework.

This chapter contains the following sections:

Introduction to Enterprise Configuration Management

Enterprise Configuration Management enables you to collect configuration information from a target, which is characterized as large and rarely changing collections of information with non-trivial structure. Such collections are collected rarely compared to regular (performance) metrics.

Configuration data should only be affected by administrators explicitly performing some change to a system, such as installing a patch or reconfiguring the target in some way. Configuration data must *not* change except due to some change initiated by a running system without explicit action from an administrator. Configuration data is collected at most once a day and should normally produce the exact same data as the previous collection.

Examples of configuration data include the maximum number of processes configured on a system and the maximum amount of available disk space. Examples of nonconfiguration data include the current number of processes on a system or the current amount of used and free disk space.

Enterprise Configuration Management provides a number of features, including:

- Infrequent (by default, daily) collection of a relatively large set of related configuration data
- On-demand refresh and scheduled refresh (through a job) of the configuration information
- Comparison of configurations to discover how they differ across targets. Users can customize comparisons through comparison templates.
- Saves configurations in the Management Repository as saved snapshots for later viewing, comparison, and any other operations related to configurations.
- Exports configurations into files and imports such files back into Enterprise Manager as saved snapshots. This can be used to archive configuration snapshots in third-party systems and to transfer them from one repository to another (for example, to transfer configuration snapshots to Oracle Support).

- Historical change tracking. When new configurations are inserted into the Management Repository, a comparison with an older configuration for the same target reveals what has changed in the configuration information. These changes are stored as part of the history of configuration information. End-users can view and search this history, as well as sign up to be notified when certain changes occur.
- Powerful search across all the configuration information in the enterprise or across a subset of targets (for example, within a group).
- Triggering association (relationship) collections to related targets.
- Provides the basis for compliance rules and standards, which are implemented on Enterprise Configuration Management data.

About Configurations

Enterprise Configuration Management collects configurations as a collection of configuration snapshots. A configuration snapshot is a large collection of information that changes infrequently relative to performance metrics. Each configuration snapshot is associated with an Enterprise Manager target. For example, an Oracle Home configuration snapshot is associated with an Oracle Home target.

Each snapshot type is associated with a given target type. For example, configuration snapshots of type `oracle_home_config` are associated with targets of type `oracle_home`.

> **Note:**
>
> Target types can have more than one associated snapshot type. For example, `oracle_home` can have other snapshot types associated with it, collecting other configuration information that is not collected by `oracle_home_config` already.

A target configuration consists of a number of collected configuration snapshots. Typically, snapshot collections occur automatically when a Management Agent starts up as a scheduled event. Because of the size of these collections and the infrequent change rate, every 24 hours is a reasonable schedule cycle. Enterprise Configuration Management has an on-demand refresh feature that you can trigger whenever you want a new configuration, regardless of the schedule.

When a configuration is inserted into the Management Repository, it replaces the previous configuration for the given target. Enterprise Configuration Management compares the two configurations for any differences and records the differences as the configuration history for the target. While the exact time of change is not known exactly, the timestamp falls within the schedule cycle (24 hours) or the on-demand refresh. You can browse the historical information related to your configurations as well as the current information.

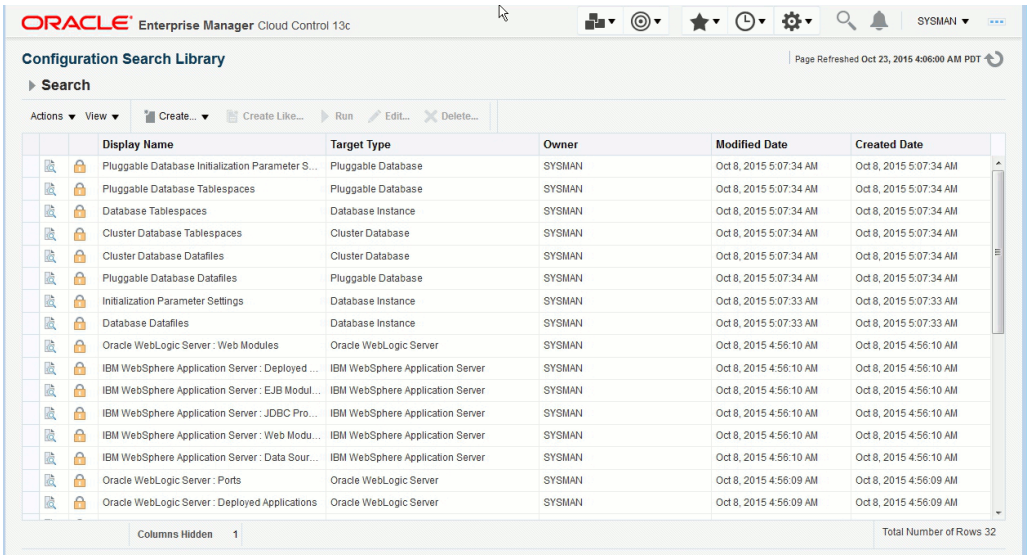
Viewing and Searching Configurations

Through the Enterprise Manager UI, you can view collected configurations and perform various operations on the configurations. Because all configurations are

recorded in the same Management Repository, you can perform configuration searches across all targets or a subset of targets. For example, find all the hosts across the enterprise with four CPUs and a minimum of 1GB of RAM.

To access the search capability, from the **All Targets** page, right-click the required target, select **Configuration**, then select **Search**. The Configuration Search Library page for the selected target appears, similar to Figure 4-1. For information about configuration searches, see the Cloud Control online help.

Figure 4-1 Configuration Search Library Page



Comparing Configurations

You can compare configurations for differences. For example, you can compare two hosts that have the same configuration to identify any problems on one of the hosts.

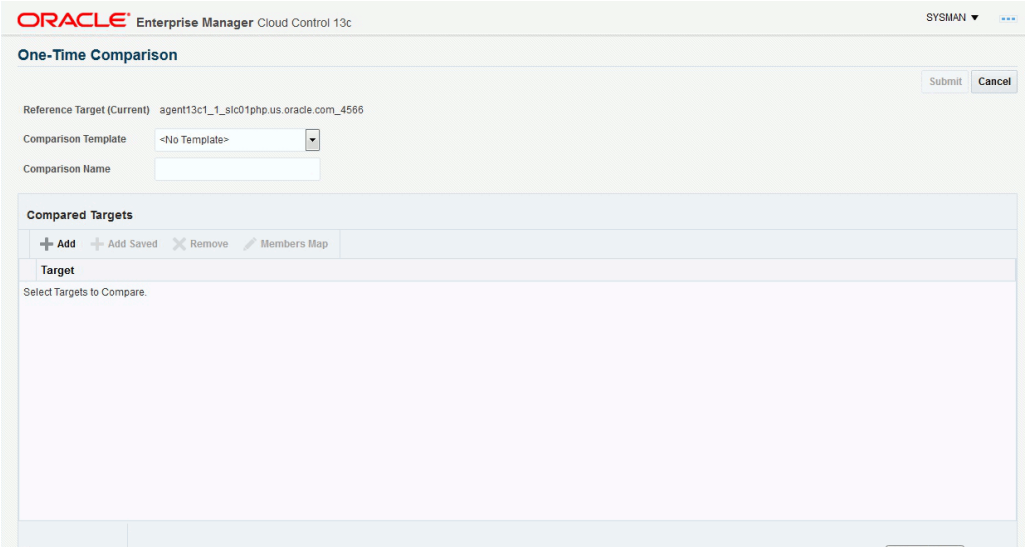
You can also compare one target's configuration, considered as the gold standard, against a number of other targets for configuration drift. Because operations such as this involve large volumes of data, you have the option of scheduling these comparisons during off-peak hours.

You can use comparison templates to customize comparisons also. A comparison template enables you to establish certain constants to take into account when comparing configurations of the given target type. For example, ignore items that are guaranteed to be different to reduce the information or to eliminate false positives. You can save these customized comparisons and share the templates among users and groups.

To compare configurations, from the **All Targets** page, right-click the required target, select **Configuration**, then select **Compare**. The Comparison Wizard appears, similar to Figure 4-2.

For information about setting up a comparison, see the Cloud Control online help.

Figure 4-2 Compare Configurations Page



Saving Configurations

Even though a configuration overwrites the previous one during collection, you have the option to save a configuration in the Management Repository for archiving.

> **Note:**
>
> The term *saved snapshot* refers to the saved configuration of a whole target and can include a number of configuration snapshots that are saved for that target and saved associations.

Also you can export a configuration to a file, which later can be imported back in to the same Management Repository or a different Management repository as a saved snapshot.

To save the latest configuration, from the **Targets** page, right-click the required target, select **Configuration**, then select **Save**. Provide a description of the configuration, then click **Submit Job**.

For information about working with saved configurations, see the Cloud Control online help.

About Associations and Topology

The Enterprise Configuration Management framework provides common mechanisms and conventions for representing relationships between targets and other IT-managed entities. These relationships, or associations, can refer to targets and also to finer-grained entities known as target components, such as an application's web service or a J2EE container's data source.

Associations can be kept up to date based on collected configuration data. As a plug-in developer, you can specify logic to derive associations from the configuration data. For more information about associations, see Using Derived Associations.

The Enterprise Manager topology viewer provides a graphical representation of how managed entities relate to other entities in the enterprise. Using the viewer, users can view the current associations.

5

Using Derived Associations

Effective management of IT infrastructure requires knowledge of the relationships between IT entities. Best practices such as those described by ITIL (Information Technology Infrastructure Library) rely on capturing and using such relationships.Enterprise Manager Cloud Control extends the kinds of relationships being supported and adds a declarative mechanism by which these relationships can be maintained. It also determines the membership of entities in a system based on relationships. Based on accurate relationships, various Enterprise Manager applications and components can support customer uses such as:

- Dependency analysis.

  For example, understanding the impact (to applications and infrastructure) of shutting down a host.
- Topology viewer.
- Change management.

  For example, tracking the source of cloned databases such as from test to production instances.
- End-to-end performance analysis, in which interdependencies between application components must be known in order to analyze and isolate issues.
- Change tracking of relationships, such as changes in the way VM resources are allocated.

This chapter covers the following:

Introduction

In Enterprise Manager, the concept of a relationship is internally referred to as an association. An association (association instance) represents a relationship between two managed entities and specifies three values, source, destination, and association type. For example, in “`database1 exposed_by listener1`", `database1` is the source, `listener1` is the destination, and “`exposed_by`" is the association type.

As a plug-in developer, you are responsible for defining those association types that apply to your managed entity types and for verifying that the correct associations (association instances) are present.

Understanding Enterprise Manager Association Concepts

This section describes association derivation rules, which provide a concise declarative means of defining association types. Association derivation (so called because the existence of associations is derived from collected data) provides a mechanism by which developers can cause association instances to be created and removed based on data collected from a target.

The association derivation mechanism allows you to keep the association consistent with the collected configuration data and to determine associations centrally based on all known data (instead of being done by agent logic, which has access to less data).

Using Association Derivation

To use association derivation:

1. Specify the logic to run after the collection of target configuration.

   The logic derives a set of association instances in the form of triples that specify the source managed entity globally unique identifier (GUID), association type, and destination managed entity GUID. For instance, the association derivation logic for targets of type oracle_listener could return triples that represent associations between the listener and each database for which it listens.

2. Create and run a SELECT statement that contains the logic used to derive the triples.

   Each returned row contains association type, source, and destination columns and represents an association that should exist.

3. Register the derivation logic against an Enterprise Configuration Management snapshot type.

   After every snapshot collection, the registered logic is invoked. Input to the logic is the GUID of the target for which the data was collected.

When the association derivation logic for snapshot S of target T is executed, the derived associations replace the previously derived associations for snapshot S of target T. For example, if associations A1 and A2 were collected yesterday and only A1 is collected today, then A2 is effectively deleted.

Using Association Derivation Rules Management

Association framework enhancements include the treatment of associations as configuration data. Enterprise Configuration Management features such as change tracking and saved snapshots now apply to associations as well as to traditional configuration data. Associations can now specify source and destination target components, as well as target GUIDs.

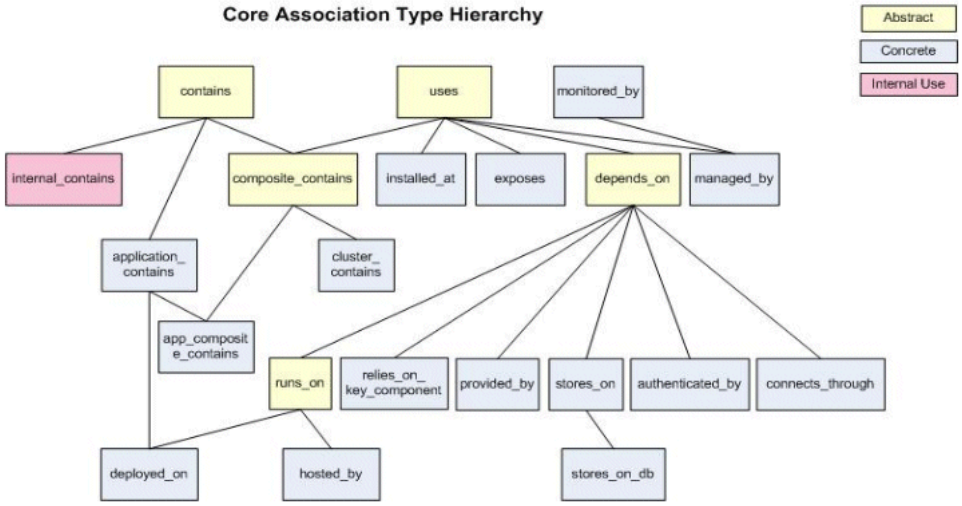
About Out-of-Box Association Types

Enterprise Manager provides a common set of association types that meets the needs of most plug-in developers. As a plug-in developer, you are encouraged to become familiar with these association types and use them if applicable. Oracle recommends that you update the Table of Integrators and Documents with links to the documents

describing your association types and your usage of all association types (allowed_pairs).

Figure 5-1 shows the core association type hierarchy.

Figure 5-1 Core Association Type Hierarchy



Overview of Plug-in Developer Responsibilities

As a plug-in developer, you are responsible for the following steps with regard to derived associations:

1. Identify all associations that need to be represented for any managed entities (MEs) that you own.

   This generally includes any containment or dependency associations between an ME you own and any other MEs. For each kind of association identified, you may need to coordinate with the owner of the related ME type to determine who should be responsible for assuring that association instances of that type are kept up to date. Some associations (in particular, `hosted_by` and `managed_by`) are automatically maintained by Enterprise Manager, so association derivation rules should not be used for these.

2. Understand the set of out-of-box association types that are shipped with Enterprise Manager and ensure the use of the most appropriate type.

   For more information, see About Out-of-Box Association Types.

3. Provide a query that returns the correct associations and performs acceptably.

   If required configuration data is not collected, you must also add such collections to assure acceptable performance. Your rules must identify the configuration tables on which the rule query depends so that the evaluation is triggered when required.

4. Ensure that association derivation rules are used to (declaratively) describe the associations that are to exist based on configuration data that resides in the Management Repository.

Rules are triggered by configuration collections (where target property changes are also treated as a configuration collection).

Maintaining Performance

Because the evaluation of derivation rules might be frequent, any poor performance of the rule queries can be problematic. Rule authors must ensure that any needed indexes are present and that they test query performance based on the specific queries that are generated for each trigger.

In particular, testing of the rule query must be done for each trigger because each trigger causes the execution of a different query. Note how rule query return values are bound to a given target GUID depending on your triggers.

You must have indexes that will make use of these bindings. Furthermore, queries must be written in such a way that they would not prevent the push of bindings from outside into your queries.

About Overlapping Associations

It is possible for more than one rule to derive the same association, although Oracle recommends that you avoid creating such overlapping rules. This section describes what happens when an association is derived by multiple rules and includes suggestions on when to avoid this and how.

Understanding Overlap Between Associations Derived by Rules

When more than one rule derives the same association, that association continues to exist until each rule no longer derives it. Sometimes, this is what you want. For example, suppose each of two application target types has knowledge of both the Oracle WebLogic Server on which it runs and the database it accesses. Based on that knowledge, each has a way to derive an association between the Oracle WebLogic Server and the database. If either rule derives the association, that association is real and should exist. Only when both rules no longer derive the association can you be sure that the association no longer exists.

The "exists when any rule derives it" semantics might not be what you intend. Consider two rules that could be defined for the `installed_on` association between the database and Oracle home. Both access the same data, but one is triggered by a property change to the Oracle home and the other by a change to the database. As soon as either rule determines the relationship is gone, then the association should be deleted. In such a case, use a single rule with two triggers.

Suppose you did not take care to write only one rule in such cases. You might think that this mistake is not serious because the association will be deleted soon. But this is not so, and the bogus association might exist indefinitely. If in the previous example, the association was derived using two rules, then the database is upgraded and its OracleHome property gets changed. The association with the old Oracle home should be removed, but this will not happen until the other rule is fired. However, nothing about the Oracle Home target has changed, so its rule is not triggered and the association remains. Indeed, it is often the case that only one target is changed and the other remains unchanged for a long period of time. As a general rule, associations based on data from a specific set of tables should be derived using a single rule with multiple triggers.

Unless there are different reasons for asserting an association exists, only use one rule. In such cases, the associations returned by derivation rules should be disjoint. Another way to state this is that for those associations, the set of all rows returned by all rule queries must specify no duplications. An association is identified by source, destination, and association type. This means that the combination of these three values should be unique.

Frequently Asked Questions

This section addresses three of the most frequently asked questions:

1. Which Tables Can I Reference in a Rule Query?
2. Are There Guidelines for When to Use Target Properties?
3. What is the Relationship Between Discovered and Derived Associations?

Which Tables Can I Reference in a Rule Query?

In most cases, your query and triggers will reference configuration (Enterprise Configuration Management) tables using the `CM$` views. If you refer to other tables and if that data might change independently of Enterprise Configuration Management table changes, then the associations might not be updated when required. If you have a use case in which a non Enterprise Configuration Management table is referenced where changes to that table must trigger rule evaluation, contact your Oracle representative.

Another consideration is the component in which the table is located. If the table your rule references is not part of the Enterprise Manager EDK, your plug-in must account for the dependency on that table's plug-in. For example, you must ensure that any object you reference already exists in the Management Repository using a plug-in dependency mechanism.

Are There Guidelines for When to Use Target Properties?

Target properties are being treated as configuration data and there is an Enterprise Configuration Management snapshot table that is populated for each target type. Some care should be taken in using data from this table:

- Many target properties are set at discovery time and never modified.
- Querying name/value pair data can be awkward and take longer than queries on other tables where the data is more structured.
  - If the data is available from both the target properties table and an Enterprise Configuration Management snapshot table, you should use the latter.
  - If you need to add collection of configuration data, you should do so in an Enterprise Configuration Management table, not as a new row in the target properties table.

In general, the use of target properties should be avoided and data should be collected and modelled using standard ECM mechanisms.

However, a rule might need to refer to target properties if, for example, the target has no Enterprise Configuration Management collections that can be added. If an association to such a target is to be created, there must be some way to identify it (for example, the rule must refer to its target properties).

If you must use target properties, then reference `MGMT_TARGET_PROPERTIES` in your rule query. You can also reference `MGMT$TARGET_PROPERTIES` in the rule query if the view already performs the join you need to do.

What is the Relationship Between Discovered and Derived Associations?

This is another example of overlapping associations (for more information, see About Overlapping Associations). For example, you might have discovery logic that discovers an association between targets T1 and T2, plus a rule that derives the same association. Oracle recommends that you do not write two sets of logic to create the same association. In this case it is suggested that:

- If a derivation rule is needed because the association might change, then write the derivation rule.
- If the association that is discovered will not change until the source or destination is removed, then discovering the association is fine and might be more efficient.

If you do write two sets of logic to create the same association (discovery logic and derivation rule), then the discovered association will remain and the derivation logic will also assert the existence of that association. If the rule evaluation later determines that the association should no longer exist, the rule's assertion will be removed, but the association will continue to exist unless you manually delete the discovered association.

6
Using the Jobs Framework

This chapter includes the following topics:

Introduction to the Jobs Framework

Enterprise Manager includes a job framework for managing the automation of administrative tasks performed against targets or groups of targets. The automation framework is tightly integrated with other Enterprise Manager subsystems such as targets, credentials, events, and so on, so the customer can monitor as well as manage their targets from the single console.

You can support automation requirements for target types defined in your plug-in by using the interfaces provided by the job framework. You can define job types as part of your plug-in, providing the automated support of critical administrative capabilities.

Enterprise Manager administrators can then schedule, execute, and monitor those jobs, in order to manage the targets supported by the plug-in. These jobs may be used to enforce management best practices, respond to alerts as corrective actions, and to otherwise automate the management of the enterprise in general.Enterprise Manager includes a job console that allows administrators to submit and monitor the execution of jobs.

Understanding Jobs

A job is the unit of work to be run by the job system. An administrator creates a job and specifies a schedule for when the job should run, such as `patch system B at midnight Sunday`. The Management Server schedules and runs jobs.

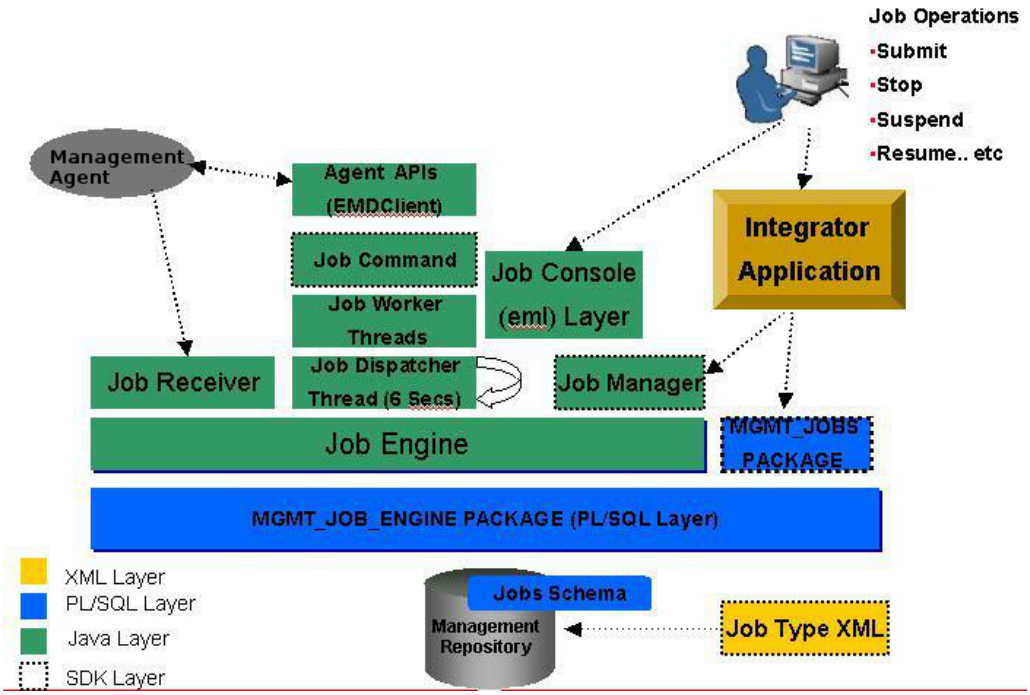
A job is based on a job type definition that defines the steps included in the job as well as the parameters required as input to run the job and the credentials necessary to access the targets accessed by the job. When a job is submitted, the values for parameters and credentials are supplied by the submitter.

A job execution is a collection of inter-related job steps. Steps can be grouped into step-sets. Steps within a step-set can run serially (one after another) or in parallel (simultaneously), but not both. Steps (and step-sets) can also be run depending on the success or failure of the other steps. For more information about these concepts, see the *Enterprise Manager Programmer's Reference*.

The steps in a job can process commands, scripts, and so on, on the Management Agent. Enterprise Manager provides several common commands that can be included in job type definitions that you create. These include such commands as remote operation (allows script execution), and file transfer commands including `put` and `get`.

The commands associated with each step are typically run on a remote node through the Management Agent. The coordination and overall status of a job is maintained by the Management Server and stored in the Management Repository.

Figure 6-1 Jobs Overview



A job may have one or more target lists. However, some jobs do not have targets and for these jobs the target list is empty or null. The target list is a set of targets that are required for one execution of the job. It is up to the job type to interpret the target list. For example, the OSCommand job type runs the specified command with the specified parameters in parallel against all the targets in the list. A job that clones schema in a database might interpret its target list a little differently. It might, for instance, consider the first target to be the *source database* from which to clone. It might consider the second argument to be a *target database* where it should populate the cloned schema. Note that a job can be submitted with multiple target lists, each one resulting in a separate execution.

Finally, every job must have a schedule. A schedule specifies when the job will run. The job system provides extensive scheduling capabilities including the ability to submit a job for immediate execution or to submit a job to run repeatedly according to any number of different scheduling options.

Defining Job Types

A job type is a specific job category, which carries out a well-defined unit of work. A job type is uniquely identified by a name. For example, *AppPatch* could be a job type that applies a patch to an Oracle applications installation. *OSCommand* could be a job type that runs a remote command, and so on.

A job type can be defined by an XML document that specifies the steps in the job, the work (command) that each step performs, and the relationships between the steps. Job types are included in a plug-in by using the `jobType` metadata service.

In addition to the job type definition, it is necessary to package any scripts referenced by the job type with the plug-in, as part of the Management Agent deployment.

For information about the definition and packaging of jobs, see the *Enterprise Manager Programmer's Reference*.

7

Using the Reporting Framework

This chapter includes the following sections:

Introduction to the Reporting Framework

The powerful reporting framework of Enterprise Manager makes information about your managed environment available to audiences across your enterprise. Reports are used strategically to present a view of enterprise monitoring information for business intelligence purposes, and they can also serve an administrative role by showing activity, resource utilization, and configuration of managed targets. IT managers can use reports to show availability of sets of managed systems. Executives can view reports on availability of applications (such as corporate e-mail) over a period of time. Enterprise Manager ships with over 100 reports addressing these various needs; additionally you can create your own customized reports.

Choosing a Reporting Technology

You can develop new reports using either BI Publisher or the Enterprise Manager Information Publisher (IP) reporting framework based on your product requirements. However, Information Publisher reports are deprecated from release 12*c* Release 1, therefore Oracle recommends using BI Publisher.

BI Publisher reports have the following advantages:

- Highly formatted, professional quality, output-only reports, with pagination and headers and footers.
- PDF, XLS, and HTML output formats.
- Enterprise Manager repository access model supported through Java Database Connectivity (JDBC) data source and per-process function for Virtual Private Database (VPD) (VPD is set as a logged on BIP user). Supports select statements and pipelined PL/SQL functions available to the MGMT_VIEW account.
- BI Publisher report access model using “foldering" and granting access to folders.
- Scheduling capabilities and delivery mechanisms (FTP and others).
- Ability to edit formats separately from the report definition.
- Ability to parameterize with BI Publisher parameter capabilities as a report is generated.

Enterprise Manager Information Publisher reports have the following advantages:

- Reports that looks like Enterprise Manager console pages (no tight control over layout).
- Tightly coupled with Enterprise Manager Oracle Management Services (OMS) and repository.
- Integrated Dashboards.

Developing BI Publisher Reports

BI Publisher reports have data model, layout, and translation components. The data model component is modeled using the BI Publisher UI. The report layout is generated using RTF templates. RTF templates are authored by BI Publisher report developers using the BI Publisher Desktop application, which is delivered as a Microsoft Word Plug-in.

A complete example of a BI Publisher report ships with the BI Publisher integration. This includes a data model and a report definition.

When a report has been developed and tested, all of these components can then be exported from BI Publisher as files and packaged as part of the plug-in.

For more information about the report development and packaging process, refer to the *Programmer's Reference*.

Developing Enterprise Manager Information Publisher Reports

Enterprise Manager Information Publisher reports are defined in XML and packaged with the plug-in. These report definitions specify the layout and content of the report. You should use the BI Publisher interfaces to develop new report definitions. You should not utilize the Enterprise Manager Information Publisher PL/SQL APIs.

These APIs are still supported for reports included in plug-ins created before Enterprise Manager 12*c* Release 1 (12.1.0.2), but are converted to 12*c* Release 1 (12.1.0.2). These APIs are not supported for the development of new reports.

8

About the Management User Interface

You can extend Enterprise Manager to support the management of new domains through the introduction of discovery, monitoring, and automation. While the Enterprise Manager framework provides a powerful set of features related to these management capabilities, most plug-in developers want to expose management capabilities in a way that is appropriate to their domain. The Metadata Plug-in Custom User Interface (MPCUI) features of Enterprise Manager provide you with this capability.

This chapter contains the following sections:

Introduction to the MPCUI Framework

The MPCUI framework supports controlled access to Enterprise Manager services through supported APIs. These APIs enable you to add user interfaces safely to Enterprise Manager as part of your plug-in, independent of the upgrade of Enterprise Manager to support other capabilities.

MPCUI provides a progressive set of options for building UIs depending on your requirements. MPCUI provides templates and reusable components that enables you to build custom UIs with a minimal amount of effort, both in terms of learning the framework for developing the UI and for certifying it. If you require more control over your UI, MPCUI provides the building blocks for constructing a customized management user experience, while providing the flexibility to construct that UI in a way most appropriate to the domain.

> **Note:**
>
> It is not necessary to include MPCUI as part of your plug-in. If you create a plug-in and do not provide custom UIs, your target still appears in Cloud Control and the default UI is shown.

Creating a Custom User Interface

MPCUI implementations are included with the plug-in as part of the Management Service section of the plug-in (for example, under the oms/metadata directory). MPCUI includes this option for implementing the user interface:

- HTML/JS based

This approach provides the ability to specify the user interface as a collection of HTML and JavaScript (JS) files. Consistent with the previous, Flex-based, release of MPCUI, multiple page support, dialog definition, charts, tables, and so on continue to be supported but now in HTML and JS.

User Interface Components

The user interface includes the following:

- Pages, Layout, and Navigation
- Packaged Regions
- Charts and Tables
- Other Components and Look and Feel

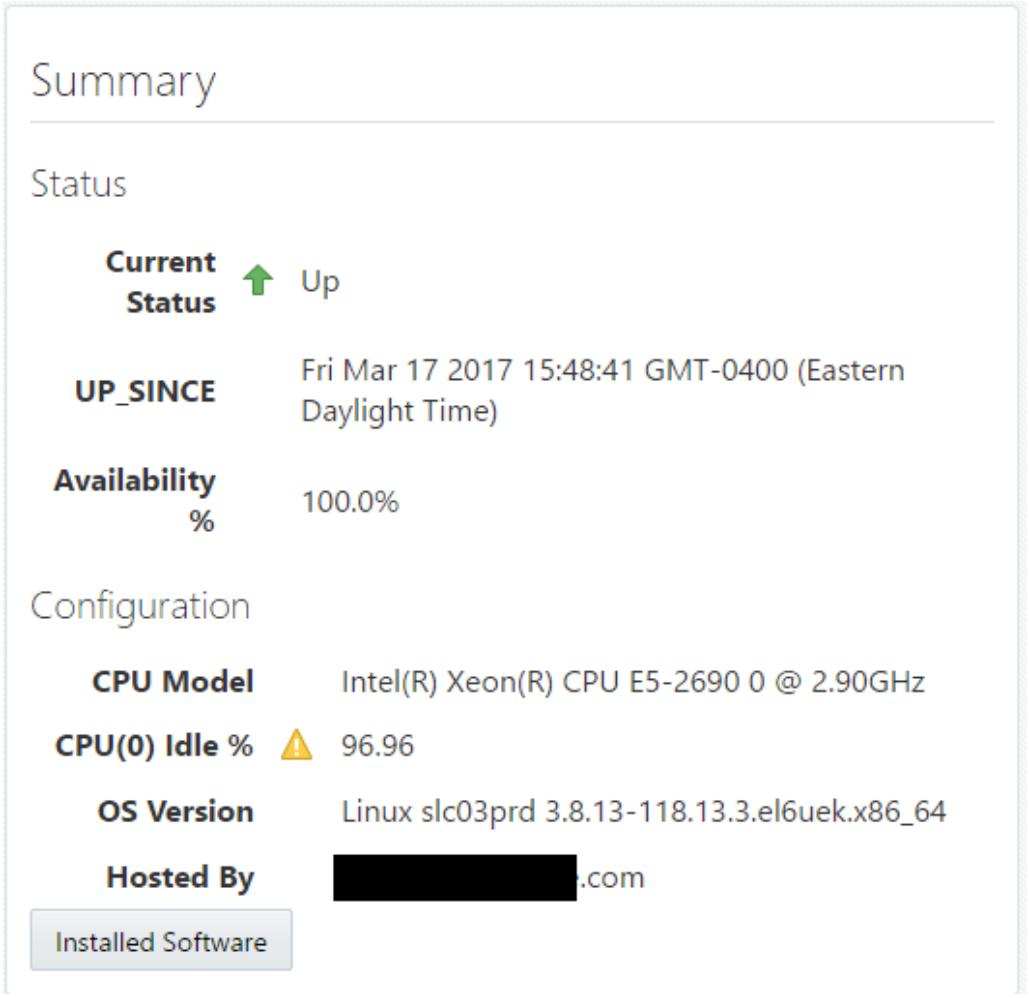
Pages, Layout, and Navigation

Using MPCUI, you can construct a user interface that is composed of multiple pages, dialogs, and trains (wizards). Define navigation between these pages as part of the MPCUI implementation as well as navigation to other Enterprise Manager pages or external URLs.

The MPCUI framework enables you to select pages that should be registered with the Enterprise Manager menu subsystem. This permits menu items to appear in the target menu allowing you to navigate to pages defined in the MPCUI implementation.

The MPCUI framework provides a simple means of laying out components on pages defined in the UI using a grid style layout. The framework includes a region component that provides a content container within the page and supports the expansion and collapse of that content. Figure 8-1 shows a region that contains summary information.

Figure 8-1 Summary Region

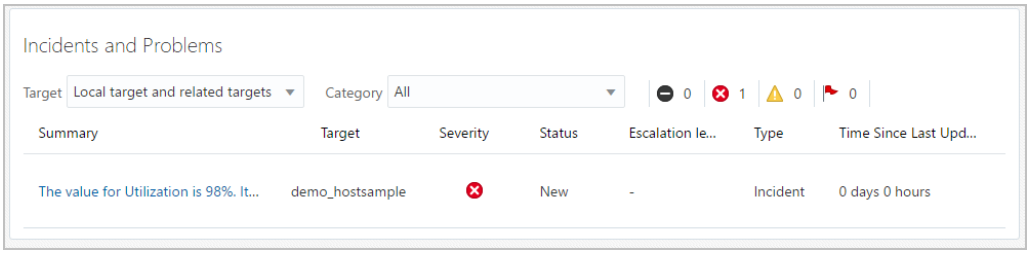


Packaged Regions

The MPCUI framework provides a number of reusable regions packaged with the plug-in development kit. This enables you to include content commonly shown in target home pages, such as event information or job summary information.

Figure 8-2 shows the issues region provided with the MPCUI framework:

Figure 8-2 Issues Region



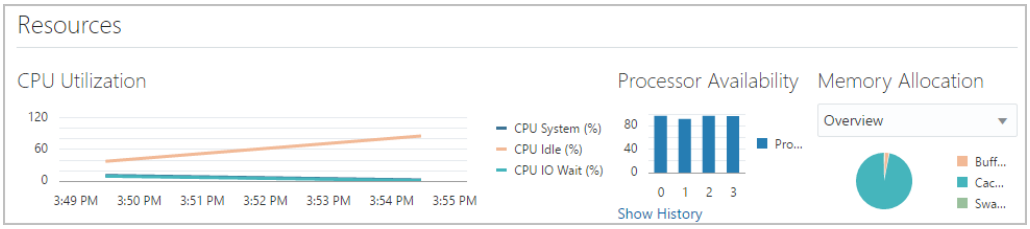
You can add the region to a page built in the user interface using a single entry in the page definition:

```
<mp-incident-region height="33%" width="100%"></mp-incident-region>
```

Charts and Tables

MPCUI provides support for including charts and tables in the user interface and simplifies the ability of selecting data to be shown in those components by specifying the metrics to be shown or the results of SQL statement execution. Supported chart types include pie, bar, line, and area charts.

Figure 8-3 Resources Region



Other Components and Look and Feel

In addition to the high-level components for page definition, regions, and charts, you have access to the complete set of components support by the JavaScript Extension Toolkit (JET) and basic HTML, including buttons, labels, text input, and so on.

In all cases, the MPCUI framework styles the components so that they appear with the same look and feel of all other Cloud Control user interfaces, ensuring that your customized UI appears as a fully integrated part of the Enterprise Manager Cloud Control product.

About MPCUI

MPCUI utilizes the JavaScript Extension Toolkit (JET) to enable the writing of HTML pages (aided by JS) that run within an Enterprise Manager page (chrome). It provides you with an experience integrated with the rest of the Enterprise Manager UI, while maintaining separation between your UI and code and the rest of Enterprise Manager.

While the framework is based on HTML/JS/JET and web services, the main goal of the framework is to simplify the process for building management UIs. To do this, the framework removes the underlying technologies as much as possible. It provides packaged components and services that can be combined to produce a management UI appropriate to your domain without requiring you to become an HTML/JS/JET, web services, or UI framework expert.

Overview

The MPCUI framework exposes a series of web services that you use to retrieve monitoring data (including target associations, properties, metrics, and configuration data) and automation services to perform synchronous tasks and asynchronous jobs.

The UI that you construct is packaged with the plug-in and deployed to Enterprise Manager along with other integration objects such as metric definitions, job definitions, and discovery and monitoring scripts.

The MPCUI objects in the plug-in are stored in the Management Repository. At run time, they are retrieved by Enterprise Manager framework code which renders a wrapper page including the Enterprise Manager chrome. The MPCUI runs within a frame in the page. This provides a sand-boxed environment that protects the rest of Enterprise Manager from any problems relating to the custom UI of a single plug-in.

The management UI has access to Enterprise Manager services, such as repository data, job subsystem, Agent services, and so on through the MPCUI services layer. The MPCUI services layer is composed of web services provided with the Enterprise Manager framework.

About the UI Framework

MPCUI includes a number of UI components that you can use to build a management UI including pages, trains, regions, charts, tables, and packaged components. Many of these components also leverage Enterprise Manager services to provide simplified access to and display of management information.

For example, the chart and table components enable you to specify metrics or packaged SQL queries within the component tags, providing a simple means of including monitoring data in the UI.

The MPCUI framework includes a number of different component layers of which you can take advantage. The framework includes the support for ensuring that the style of the management UI is consistent with Enterprise Manager style guidelines without requiring any additional effort.

To simplify the structure of an MPCUI-based application and make it fit more naturally into the Enterprise Manager framework, MPCUI includes the notion of a page. The management UI can include one or more pages to navigate to and from using constructs provided by the MPCUI framework. You can also integrate the page or pages into the Enterprise Manager menu system.

Construct the management UI specific to your domain using the appropriate UI elements included in the MPCUI framework and access Enterprise Manager services through the MPCUI services client library.

MPCUI Services

When constructing the user interface using MPCUI, you have access to a number of services supplied with the framework, enabling you to access Enterprise Manager data and perform operations. These services include:

- **Target Properties**

  Supports the retrieval of target instance properties, including version, category, and other dynamic instance properties

- **Target Associations**

  Supports the retrieval of targets associated with the current target, related hosts, members, and so on

- **Metric Data**

Supports the retrieval of metric data including current (real-time) values from the Management Agent or historical data stored in the Management Repository

- **SQL Results**

  Supports the processing of SQL statements packaged with the plug-in and the retrieval of the results of those statements

- **Jobs and Remote Operation**

  Supports the scheduling of jobs or the immediate processing of remote operations (Management Agent scripts) to perform administrative capabilities

MPCUI and the Extensibility Developer Kit

The Extensibility Development Kit (EDK) includes the elements necessary to develop MPCUI including the libraries and scripts to build the MPCUI.

9
Understanding Discovery

Automatic Discovery is part of Configuration Management in Enterprise Manager. It forms the foundation for configuration management and Enterprise Manager by identifying IT components and allowing you to manage these components using Enterprise Manager.

To view a visual demonstration about how to automatically discover hosts on your network, deploy Management Agents on these hosts and then discover targets on these hosts, access the following URL and click **Begin Video**.

`https://apex.oracle.com/pls/apex/f?p=44785:24:0::NO:24:P24_CONTENT_ID,P24_PREV_PAGE:5450,1`

This chapter contains the following sections:

Introduction to Automatic Discovery

Adding a target to Enterprise Manager through discovery involves two steps:

1. Discovering potential targets on hosts that are managed by Enterprise Manager.
2. Promoting targets to be managed by Enterprise Manager by assigning the target to a Management Agent.

Users can enable automatic discovery of potential targets on a host managed by Enterprise Manager.

To enable auto discovery, Management Agent-side discovery scripts are packaged with the plug-in. When auto discovery is configured on a host, the discovery part of the plug-in (discovery scripts) is deployed on the host.

After auto discovery is configured to run on the managed host, discovery takes place periodically and sends potential targets to the Management repository.

Users can view the targets from the discovery results UI and then take various actions such as promoting the target to be managed by Enterprise Manager.

Key Benefits of Adding Automatic Discovery

The following are the key benefits of adding automatic discovery:

- Easier to add targets to Enterprise Manager. Otherwise, the user must manually add each target to Enterprise Manager individually. This is a cumbersome and time-consuming task.
- Easier to manage multiple objects on a single host. Otherwise, the user must manually add each object on the host as a target and then repeat the process on all other hosts.

Automatic Discovery Overview

This section provides an overview of the components of automatic discovery:

- Plug-in

  The plug-in contains the components required to discover and promote targets of a particular target type. The plug-in includes discovery metadata to register with the discovery framework, discovery scripts, and monitoring scripts.

  > **Note:**
  >
  > Only the discovery part of the plug-in (discovery scripts and any content required to run discovery) is deployed when scheduling discovery.
  >
  > The complete plug-in (monitoring and discovery) is deployed only when the discovered target is promoted or when a target is manually added.

- Discovery module

  The discovery module defines a set of target types to discover on the hosts to which the plug-in is deployed.

  Specify discovery modules in the discovery metadata XML file. For information about the discovery metadata XML file, see the *Enterprise Manager Programmer's Reference*.

- Discovery metadata

  Discovery metadata enables you to register automatic discovery with the discovery framework. The Extensibility Development Kit (EDK) provides a discovery XML schema definition (XSD) for registering automatic discovery. For information about defining discovery metadata and a description of the elements, see the *Enterprise Manager Programmer's Reference*.

- Discovery Content

  Discovery content includes all the Perl scripts and JAR files that are required to discover targets of a particular type.

- Discovery parameters

  A discovery parameter passes input entered from the UI to the discovery script. These discovery parameters are available as environment variables in the discovery script running at the Management Agent. The discovery script uses this information when performing discovery.

  Define discovery parameters in the discovery metadata XML file. For information about the discovery metadata XML file, see the *Enterprise Manager Cloud Control Programmer's Reference*.

- Discovery schedule

  For plug-ins that are deployed already when Management Agents are installed or deployed to new hosts, and if discovery does not require any user inputs, then discovery is configured automatically and runs every 24 hours.

10

Understanding Compliance Standards

The Oracle Enterprise Manager Compliance Management solution provides the capability to define, customize, and manage Compliance Frameworks and Compliance Standards. It also provides the tools to evaluate targets and systems for compliance with business best practices in terms of configuration, security, storage, and so on.

This chapter contains the following sections:

For a detailed explanation of compliance, refer to the Managing Compliance chapter in the *Oracle Enterprise Manager Lifecycle Management Administrator's Guide*.

To view a visual demonstration about the Compliance Management framework, access the following URL and click **Begin Video**.

`https://apex.oracle.com/pls/apex/f?p=44785:24:0::NO:24:P24_CONTENT_ID,P24_PREV_PAGE:5773,1`

About the Compliance Management Solution

The Oracle Enterprise Manager Compliance Management solution:

- Determines if targets and systems have valid configuration settings automatically.
- Determines if targets and systems are exposed to configuration-related vulnerabilities automatically.
- Advises on how to change configuration to bring targets and systems into compliance with respect to best practices.
- Provides real-time monitoring of a target's files, processes, users, Windows registry entries, and more to let Enterprise Manager users know where a configuration change is taking place in their environment.
- Determines if real-time detected configuration changes are authorized by open change management requests. It creates violations when an action is determined to be unauthorized.
- Provides Oracle provided compliance standards to map to Compliance Standard rules. This mapping enables you to visualize how noncompliant settings and actions will affect any compliance framework that an organization follows.
- Provides a compliance-focused view of the IT configuration and change that is suitable for Line of Business owners, IT managers, and compliance managers to refer to regularly, enabling them to check on their organization's compliance coverage.

Overview of Compliance Management

The following sections provide an overview of the features of compliance management:

About Compliance Framework

A compliance framework is an industry-specified best practices guideline that deals with the underlying IT infrastructure, applications, business services and processes, and how they are organized, managed, and monitored. Compliance frameworks are hierarchical to allow for direct representation of these industry frameworks.

For information about defining a compliance framework and examples of compliance frameworks, see the *Oracle Enterprise Manager Cloud Control Extensibility Programmer's Reference*.

About Compliance Standards

A compliance framework maps to a set of compliance standards that perform a collection of checks following broadly accepted best practices to ensure that IT infrastructure, applications, business services and processes are organized, configured, managed, and monitored correctly. A compliance standard evaluation can provide information related to platform compatibility, known issues affecting other customers with similar configurations, security vulnerabilities, patch recommendations, and more. Customers can run an evaluation of compliance standards in order to learn about how they can bring their systems into compliance with recommended best practices and improve the stability and security of their systems.

A compliance standard is Enterprise Manager's representation of a compliance control that must be tested against a set of IT infrastructure to determine if the control is being followed. A compliance control is a description of the *test* that an IT organization would perform to ensure a policy, process, or procedure is being followed in a compliant manner. Compliance standards can be mapped to compliance frameworks so that violations can result in a compliance score impact on the compliance framework.

For information about defining compliance standards and examples of compliance standards, see the *Oracle Enterprise Manager Cloud Control Extensibility Programmer's Reference*.

About Compliance Standard Rules

Oracle Enterprise Manager Cloud Control 13*c* has five types of rules:

- Repository Rule

Performs a check against any metric collection data in the Enterprise Manager repository

- Real-time Monitoring Rule

  Monitors actions to files, processes, and more. Also captures user login and logout activities

- WebLogic Server (WLS) Signature Rule

  Checks a WebLogic target for support best practice configurations.

- Agent-side Rule

  Detects configuration problems on the Management Agent. This enables the implementation of the Security Technical Implementation Guide (STIG) security specifications.

- Manual Rule

  There are checks that must be performed but cannot be automated. For example, a common security check is "to ensure secure access to the data center". These types of checks can be accounted for in a compliance framework.

Compliance standard rules specify the actual check that is going to happen. These rules are mapped to one or more compliance standards.

For information about defining compliance standard rules and examples of compliance standard rules, see the *Oracle Enterprise Manager Cloud Control Extensibility Programmer's Reference*.

Some Considerations for Creating Compliance Standards

A compliance standard refers to one or more compliance standard rules. When creating a compliance standard, the standard should be granular enough so that it can map appropriately to one or more related compliance frameworks. For example, consider this compliance framework structure that exists in the Oracle Generic Compliance Framework:

- Change and Configuration Management (compliance framework subgroup)
  - Database Change (compliance framework subgroup)
    - Configuration Best Practices for Oracle Database (compliance standard)
    - Configuration Best Practices for Oracle RAC Database (compliance standard)
    - Configuration Best Practices for Oracle Pluggable Database (compliance standard)

  Many compliance standards will exist that should be mapped to this part of the Compliance Framework structure, each with their own rules to address this specific requirement. One may check that configuration settings are set properly. Another may be used to check in real-time if anyone changes a configuration setting.

  In this example, the "Database Change compliance framework subgroup" can relate to many different types of targets. Oracle Database, Oracle RAC Database, and Oracle Pluggable Database all have their own types of configurations that all need to be secured. Any Standards created to monitor these target-specific configurations would map to the same "Database Changes subgroup".

If compliance standards are structured in a granular way so that they can map to existing and future compliance frameworks, then violations in a rule can be rolled up to impact the score of the compliance framework properly.

About Compliance Evaluation

Compliance standards are evaluated on targets. Evaluation results of the compliance framework, compliance standards and target levels are available to the end user from the Enterprise Manager UI.

Compliance evaluation is a process of validating requirements and regulations imposed by a compliance standard against a target. To measure this, the compliance standard rules perform single health or real-time monitor checks that are grouped into compliance standards, which together are one *test* of compliance. Then these compliance standards are grouped into respective compliance frameworks so that the results of the *test* can be associated with the relevant areas of the customer's framework.

Compliance evaluation generates a score for a target, that is how much the target is compliant with the standard. A 100% Compliance Score means that the target follows all requirements and regulations imposed by the compliance standard.

Because Target Compliance must be monitored regularly, you must associate a compliance standard with targets. Evaluation is performed automatically for any associated targets when their state refreshes.

11
Understanding Software Library

This chapter contains the following sections:

Introduction to Software Library Framework

Oracle Software Library (Software Library) is one of the core features offered by Enterprise Manager Cloud Control. Technically, it is a repository that stores certified software entities such as Software Patches, Virtual Appliance Images, Reference Gold Images, Application Software and their associated directive scripts. Software Library enables you to select any of the Oracle-supplied entities and customize them, or create a custom entity of your own. Once defined, you can reference these reusable entities from a Deployment Procedure to automate the operations like: patching, provisioning, and so on.

Key Features of Software Library Framework

Software Library framework supports:

- Defining and registering metadata that is used by plug-in integrators to extend a Software Library to include extensions, and Out-of-box entities. For more information, see Defining Metadata to Extend Software Library.
- Creating, managing, and accessing Software Library entities using various interfaces like: Software Library console, EM CLI, Action Script API, and so on which leverage the custom extensions registered. For more information, see Creating and Managing Software Library Entities.
- Using the Software Library entities in various flows like: staging through Software Library console, job step execution, in a deployment procedure, and so on. For more information, see Using Software Library Entities.

Software Library Extensibility Concepts

This section gives a high level overview about the various attributes used in the Software Library extensibility framework:

- Types and Subtypes

All entities in Software Library belong to a type or a subtype. Normally, a type and subtype together define certain common features of the entities in terms of common and searchable metadata/configuration properties, their default values, file association requirements, and so on. Typically, Software Library framework defines and maintains the following types of artifacts:

- **Directive**: Entities of this type represent scripts or executables.
- **Component**: Entities of this type typically represent an installable software bundle. A subtype of the component type called **Generic Component** is also available by default with Enterprise Manager Cloud Control.

All the other entity types that are appear in Software Library console, for example: Virtualization, Bare Metal Provisioning, and so on are basically extensions to the Software Library. These custom types and subtypes are part of plug-ins that ship by default with Enterprise Manager, and appear in the Software Library console once the EM is configured. Starting with Enterprise Manager 12.1.0.3 patch set 2, the Software Library framework is being extended to Oracle partners so that they can use it efficiently to define and register their own custom types and subtypes.

> **Note:**
>
> If your have entities other than Directives, then you are recommended to create your own custom type and one or more subtypes.

- Folders

  A folder is a container of entities. A folder either contains other child folders or entities in it. Software Library allows you to organize the different user-defined or plugin-defined entities into logical folders for efficient management. Folders can be referred by their URN in Software Library.

- Entities

  Entities are the primary artifacts stored in Software Library. An entity always has a folder associated. Similar entities may be grouped under a logical folder, and are further categorized by the type or subtype they are assigned. In general, it is a good practice to organize related entities into folders of their own.

  An entity can optionally have the following:

  - Attributes: Attributes are defined by the type/subtype definition. Usually, they contain simple string data types and must be defined in the metadata. Note that they are applicable to all the revisions of an entity.
  - Properties: Properties are used for specifying environment specific values. These are applicable only to specific revisions of an entity.
  - Attachments: Attachments are files that are related to the entity and are to be stored with the entity. Attachments may be any document or file that describes the entity or its associated script/software/configuration to its consumers. For example, Readme file. Attachments do not typically participate in patching/ provisioning flows, and are not staged or copied to the targets.
  - Notes: Notes are comments that can be added to an entity. These are applicable to all revisions of an entity.
  - Associated Files: Associated files can either be uploaded to a Software Library or be stored and referenced from an external location. Once associated, the

files can be retrieved during provisioning, and staged to the desired destination target by Software Library. These are applicable only to particular revisions of an entity.

You can specify one of the following maturity status: Untested, Beta or Production for every entity revision. An entity revision can be in one of these states: Incomplete, Active or Ready. The state of an entity revision is computed based on the state of its associated files and related metadata.

> **See:**
>
> For more information about Software Library entities and its usage, refer to *Oracle Enterprise Manager Administration.*

- Entity Revisions

  An entity can have one or more revisions. When the entity is created, its revision in Software Library is set to 0.1, following which, with each update the entity is revised by 0.1. Every entity revision is identified by a unique internal identifier, referred to as the revision's Uniform Resource Name (URN). Using the URN, you can identify entities even outside Software Library framework, for example, in jobs, deployment procedures, and so on. The URN is used to refer to an entity revision while attempting to use or create or modify it using interfaces like EM CLI, Jobtype, and so on.

- External ID

  When a plug-in integrator defines entities in Software Library metadata XML, an identifier called the `External ID` needs to be specified for the entity. This identifier is used to track changes to an entity definition, while releasing a plug-in, in comparison with a previous release of the plug-in. When the plug-in is upgraded, the External ID of the latest revision is compared to the previous one, and only if the Ids do not match, the updated entity from the XML is re-applied. Therefore, revision is specific to Software Library updates of the entity, while the External ID helps to track the changes to the entity definition in metadata.

Defining Metadata to Extend Software Library

Plug-in integrators can define Software Library metadata like: types, subtypes, entities, folders, and register them using the Metadata Registration Service (MRS). The steps required to create and edit the entity through UI, default values, attributes can be described in the type/subtype metadata. The name, description, properties and files can be described in the entity metadata. During plug-in installation, the metadata is registered with Software Library, and the defined types, subtypes, folders, and entities get created accordingly.

Once registered, the custom types, subtypes, folders, and entities can be accessed using the Software Library interfaces like Cloud Control UI or EMCLI. The registered folders and entities are Oracle Owned, and can be viewed, to use them, you can use the Create Like feature of Software Library that enables you to customize the default entities to suit your requirements. In addition to this, you can create new entities through UI or EMCLI for the custom type or subtype registered. The UI flows for the entities of the custom type/subtype are defined by the UI specification in the metadata.

When the plug-in is uninstalled, the types, subtypes, folders, entities are removed. Entities of the custom type/subtype created using UI or EMCLI are also removed during the un-deployment of the plug-in.

Creating and Managing Software Library Entities

There are many interfaces to create and manager Software Library entities:

- Enterprise Manager Cloud Control
- Enterprise Manager Command Line Interface (EMCLI)

> **Note:**
>
> An entity once created and saved in the Software Library using one of the following interfaces: Software Library console, EM CLI, Action Script API, and so on, can be accessed at a later point using any of the other interfaces.

Enterprise Manager Cloud Control

To use Software Library Console, from **Enterprise** menu, select **Provisioning and Patching,** and then click **Software Library.** The Software Library console is single GUI enabled page that facilitates managing the complete lifecycle of the entities (create, manage, delete) efficiently. Using the Software Library features, you can perform a host of tasks such as: create folders, create entities, edit entities, delete entities, manage maturity status, add notes and attachments, manage privileges, and so on. Once the plug-in is defined and registered, from the Software Library console, you can choose the plug-in integrator defined types and subtypes when creating the entities. This in turn allows, creating entities that closely model the required artifact. You can also customize the default Oracle-owned entities using the **Create Like** option. For more information about using the Enterprise Manager Cloud Control to perform these operations, see *Oracle Enterprise Manager Administration.*

Enterprise Manager Command Line Interface (EMCLI)

You may choose to use a command line interface called EM CLI to automate the creation and management of entities. For more information about the Software Library verbs and their usage, see *Oracle Enterprise Manager Lifecycle Management Administrator's Guide*.

Using Software Library Entities

The user defined entities and the default entities from the Software Library can be used to complete various processing flows.

> **Note:**
>
> For information about how plug-in UI uses the Software Library search service, see *Oracle Enterprise Manager Cloud Control Extensibility Programmer's Reference*.

- **Using Entities in Jobs**

  Plug-in integrators who create and register custom job types can nest the Software Library job types `SwlibStageEntities` and `SwlibUploadFiles` within their job type definition to create custom flows.

  For more information on how to create custom job types, refer to the section in EDK for Adding job types

- **Using entities in Deployment Procedures**

  You can either create your own deployment procedure using the PAF framework, (User Defined Deployment Procedures), or customize the default Oracle-owned deployment procedure to suit your requirement. Lets assume that to do so, you need to include a Component step or Directive step to search for the component/ directive, and include them in the procedure. When these steps in procedure are executed, the staging of the chosen entity is performed.

  For more information about creating UDDP, and Customize Oracle Owned Procedures, see *Oracle Enterprise Manager Lifecycle Management Administrator's Guide*

- **Staging through Software Library Console UI**

  To test and verify the usage before including them in other flows, you can select the required entity and stage the same through Software Library Console UI.

- **Using Action Script API**

  Entity details can be searched and retrieved through Action Script web service API. Once retrieved, the details can be displayed in a custom manner like FLEX UI.

Index

A

B

C

D

E

F

G

I

J

L

M

N

O

P

R

S

T

U

V

W