Oracle® Process Manager and Notification Server

Administrator's Guide

10*g* Release 3 (10.1.3)

B15976-01

January 2006

ORACLE®

Oracle Process Manager and Notification Server Administrator's Guide, 10*g* Release 3 (10.1.3)

B15976-01

Primary Author: Kurt Heiss

Contents

4 opmnctl Commands

10 Configuring Custom Process

A OPMN Troubleshooting

Index

Preface

This guide describes how to administer Oracle Process Manager and Notification Server (OPMN) for management of Oracle Application Server components.

This preface contains these topics:

Audience

The *Oracle Process Manager and Notification Server Administrator's Guide* is intended for administrators of Oracle Application Server.

Documentation Accessibility

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

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

Accessibility of Code Examples in Documentation

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

Accessibility of Links to External Web Sites in Documentation

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

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, seven days a week. For TTY support, call 800.446.2398.

Related Documentation

For more information, see these Oracle resources:

- Oracle Application Server Documentation Library
- Oracle Application Server Platform-Specific Documentation on Oracle Application Server Disk 1

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

What's New in OPMN?

This chapter describes the new features of Oracle Process Manager and Notification Server (OPMN) available in Oracle Application Server 10*g* Release 3 (10.1.3).

This chapter includes the following topics:

1.1 Grid Computing and OPMN

Grid computing is a software architecture designed to effectively pool together large groups of modular servers to create a virtual computing resource across which work can be transparently distributed. Grid computing enables computing capacity to be used effectively, at low cost, and with high availability.

With the new configurations and functionality of OPMN in 10.1.3, you can effectively utilize the possibilities inherent in the grid computing model. You can manage all of the computers in the grid using available OPMN commands.

1.2 opmn.xml Configuration

In Oracle Application Server 10.1.2 (10.1.2), configuration for the Oracle Notification Server (ONS) daemon was located in the `ons.conf` files. This file existed separately from the `opmn.xml` file, which is used for configuration of OPMN.

In 10.1.3, configuration for ONS is an element of, and can be configured from the `opmn.xml` file.

The information that was stored in `ons.conf` is now configured within the topology section under the `notification-server` element in the `opmn.xml`.

The following is an example of the `ons.conf` element in the 10.1.3 `opmn.xml` file:

```
<notification-server>
   <topology>
      <nodes list="node-list"/>
      <discover list="discover-list"/>
      <gateway list="gateway-list"/>
   </topology>
```

In 10.1.2 OPMN extracted the `instance id`, `instance name`, `cluster id`, and `cluster name` values from the `dcm.conf` file. In 10.1.3, these values are now specified directly as attributes to the `process-manager` element.

The following is an example of the `dcm.conf` element in the 10.1.3 `opmn.xml` file:

```
<process-manager
id="instance id"
name="instance name"
cluster-id="cluster id"
cluster-name="cluster name">
```

1.3 OPMN Logging Mechanism

OPMN and OPMN-managed processes generate log files during processing to enable you to troubleshoot difficulties you might have in execution of the process.

In 10.1.2, standard and debug messages were located in either the `ipm.log` or `ons.log` files.

In 10.1.3, standard and debug messages are located in the following:

- `opmn.log`: contains standard international log messages, all standard OPMN log messages and messages for ONS and Oracle Process Manager (PM).
- `opmn.dbg`: contains OPMN debug log messages (English only) for ONS and PM.
- `opmn.out`: OPMN console log messages (`stdout` and `stderr`)

In 10.1.3, logging is configured by component codes rather than level codes. The logging messages contain the literal value based on logging levels rather than an integer value; for example: `none, fatal, error, warn, notify, debug1, debug2, debug3, and debug4`.

You can dynamically query either the standard or debug log parameters using `opmnctl` commands. For example:

```
> opmnctl query target=log
> opmnctl query target=debug
```

You can also dynamically set the standard or debug log parameters using `opmnctl` commands. For example:

```
> opmnctl set target=log comp=<component codes>
> opmnctl set target=debug comp=<component codes>
```

The new setting is reset to the value in the `opmn.xml` file after OPMN restarts.

1.4 Dynamic Discovery

In 10.1.2, each OPMN instance had to be configured with the host and port values of the other ONS servers that it communicated with. This list was maintained in the `ons.conf` file that is maintained by DCM with a list of all of the ONS servers in a cluster. Whenever this file changed, restarting OPMN was necessary to reflect this change. In a grid environment where the number of servers that OPMN will communicate with may grow into the hundreds and where servers may come and go on a frequent basis, this type of static configuration was not desirable.

In 10.1.3, OPMN uses dynamic discovery of other ONS servers. Instead of configuring a list of all other servers to connect to, a discovery mechanism consisting of a multicast address or list of discovery servers is used by OPMN. ONS uses the discovery mechanism to announce new servers and join them into the ONS topology dynamically. This reduces the amount of configuration necessary for each Oracle Application Server instance, eliminates the need to restart OPMN when the topology changes, and removes configuration changes when the topology changes.

With dynamic discovery, the ONS network topology includes all of the Oracle Application Server instances that have been configured with the same discovery information.

1.5 Dynamic Resource Management

In 10.1.2, some features of process management for grid computing included the following:

- Management of all Oracle Application Server components (through the use of component modules).
- Ability to specify start order dependencies between Oracle Application Server components.
- Remote commands enabling execution of process management commands across hosts.
- Collection for Dynamic Monitoring Service (DMS) metrics for managed Oracle Application Server components including system metrics (for example, memory usage, CPU usage) and component specific metrics (for example, average response time, database connections in use).

In 10.1.3, the process management for grid computing is further enhanced with Dynamic Resource Management (DRM). DRM functionality provides a way for you to customize the management of your processes through configuration changes only. DRM enables you to have process management commands issued based on system conditions according to a set of user-configured directives.

DRM is designed to operate on each Oracle Application Server instance. In a cluster environment, DRM functionality is available on each separate local instance.

DRM enables you to specify a set of conditions that will trigger process management commands to be automatically issued. This is accomplished using Resource Management Directives (RMDs). RMDs describe a condition and action that should be taken when the condition occurs.

DRM operates on the DMS metrics that are available to OPMN. Metrics-based load balancing between Oracle Application Server instances in which OC4J passes information to `mod_oc4j` is available in 10.1.3.

Some examples of RMDs are:

- Start an additional OC4J process every day at 5 p.m. to accommodate peak usage hours.
- Restart an OC4J process whenever it's heap usage grows beyond 500 MBs.
- Spawn an additional OC4J process when average response time exceeds 500 milliseconds as long as there are less than 4 processes running.

For more information about RMDs, refer to Section 3.3, "Dynamic Resource Management".

1.6 Control at the Application Level

In 10.1.2, OPMN did not permit management of processes at the `process-set` or `process-type` level. For example, in 10.1.2, the smallest unit that OPMN can manage for Oracle Containers for J2EE (OC4J) is a single Java Virtual Machine (JVM). In this example, the JVM may actually be running multiple applications that will all be affected by either starting, stopping, or restarting.

In 10.1.3, J2EE applications are supported using the OPMN process request mechanism. There is automatic application state and metric updates. You can start, stop, restart, and check the status of your J2EE application using `opmnctl` commands.

OPMN manages applications with the help of runtime changes from OC4J and Oracle HTTP Server. This allows the shutdown or restart of an individual application which allows much finer grained control for performing operations such as application upgrades or resetting of unresponsive applications.

1.7 Service Failover

In 10.1.2, configured Oracle Application Server components were meant to be kept running by OPMN. For example, in 10.1.2, if an `opmnctl startall` command is entered, OPMN will start all configured Oracle Application Server components.

In 10.1.3, to allow for more flexible decisions at runtime, a dormant state for configured components has been implemented. When a configured Oracle Application Server component is in the dormant state, it is not started initially but is ready to be activated if needed.

The new dormant state implemented in 10.1.3 also introduces the concept of service failover. Service failover is mechanism in which a single or limited number of configured Oracle Application Server components are kept running on one of the servers within the computing grid. A configured Oracle Application Server component on one of the computers within the grid is started if the original running component fails. This can be thought of as having a dormant application configured on several or all grid computers and having OPMN constantly maintain a state across the grid where one instance is currently running the application. This new service failover functionality also enables you to configure preferential selection of each computer within the grid. For example, you may to wish to insure that an OC4J instance for your enterprise is run on a computers with a specific hardware setup.

1.8 Progressive Request Report

In 10.1.2, all parts of a user OPMN request had to finish before the results were reported back to the user.

In 10.1.3, user OPMN requests are reported in sequence and are available for review as each part of the request completes. The new progressive request report functionality,

which utilizes the new `report=true` attribute, causes OPMN to report back on each part of a request as it completes. The reports are available for process start, stop, and restart as well debug requests. For scoped requests each participating OPMN will send back reports to the originator for the request as each part completes.

1.9 Sequential Requests

In 10.1.2, an OPMN request is run for all affected processes at the same time, unless a dependency dictates a specific ordering. In 10.1.3, the implementation of sequential requests is introduced. Sequential requests are a mechanism that enables OPMN to perform a user request one process at a time.

By default OPMN issues jobs for all processes in parallel unless a dependency dictates a specific ordering.

In 10.1.3, if you issue a request in which `sequential=true` is specified as part of the command, then OPMN will only run the request on a single process at a time, waiting for the request to complete on the first before running the request on the second. When the request has finished on one process, it works on the next.

Dependencies are still honoured, and take part in the request sequentially as well.

1.10 opmnctl commands

In 10.1.3, the `opmnctl` commands offer greater ability to monitor your processes as well as an easier method for modification of the `opmn.xml` file.

The following list describes the new 10.1.3 opmnctl commands:

- `opmnctl set`: enables you to set OPMN log file parameters.
- `opmnctl query`: enables you to query OPMN log file parameters.
- `opmnctl config`: enables you to modify the opmn.xml file.
- `opmnctl status -port`: enables you to display the request connect string used to connect to the OPMN daemon.
- `opmnctl status -app`: displays the lifecycle of applications to be managed using opmnctl commands.

1.11 IPv6

In 10.1.2, ONS used version 4 of the Internet Protocol (IPv4), a 4 byte unsigned integer value (for 32 bit addresses) and the two byte unsigned integer (the remote port value) to identify any node in a cluster.

In 10.1.3, ONS supports IPv4 and the soon to be implemented version 6 of the Internet Protocol (IPv6) network interface.

IPv6 is intended to address the concern that there are too few IP addresses available for the future demand of device connectivity (especially cell phones and mobile devices). For more information refer to Section 3.10, "IPv6 Support"

2

OPMN: Overview

This chapter provides an overview of OPMN for Oracle Application Server. It features the following topics:

2.1 What is OPMN?

OPMN is installed and configured with every Oracle Application Server installation type and is essential for running Oracle Application Server.

OPMN features the following functionality:

- Provides a command-line interface for process control and monitoring for single or multiple Oracle Application Server components and instances.
- Provides an integrated way to manage Oracle Application Server components.
- Enables management of Oracle Application Server subcomponents and sub-subcomponents.
- Channels all events from different Oracle Application Server component instances to all Oracle Application Server components that can utilize them.
- Solves interdependency issues between Oracle Application Server components by enabling you to start and stop components in order.
- Enables customizing of enterprise functionality by using event scripts.
- Enables gathering of host and Oracle Application Server process statistics and tasks.
- Provides automatic restart of Oracle Application Server processes when they become unresponsive, terminate unexpectedly, or become unreachable as determined by ping and notification operations.
- Provides automatic death detection of Oracle Application Server processes.
- Does not depend on any other Oracle Application Server component being up and running before it can be started and used.

The OPMN server should be started as soon as possible after turning on the computer. OPMN must be running whenever OPMN-managed components are turned on or off.

Note: On the Microsoft Windows operating system, OPMN is installed as a Windows service (`Oracle<OracleHomename>ProcessManager`). It starts up automatically when you start or restart your computer. Refer to Section 4.3.2.3, "opmnctl startall" for more information.

Oracle Application Server components managed by OPMN should never be started or stopped manually. Do not use command line scripts or utilities from previous versions of Oracle Application Server for starting and stopping Oracle Application Server components. OPMN must be the last service turned off whenever you restart or turn off your computer.

Use the Application Server Control Console and the `opmnctl` command line utility to start or stop Oracle Application Server components.

2.1.1 Grid Computing and OPMN

Grid computing is a software architecture designed to effectively pool together large groups of modular servers to create a virtual computing resource across which work can be transparently distributed. Grid computing enables computing capacity to be used effectively, at low cost, and with high availability.

With the new configurations and functionality of OPMN in 10.1.3, you can effectively utilize the possibilities inherent in the grid computing model. You can manage all of the computers in the grid using available OPMN commands.

2.2 How OPMN Works

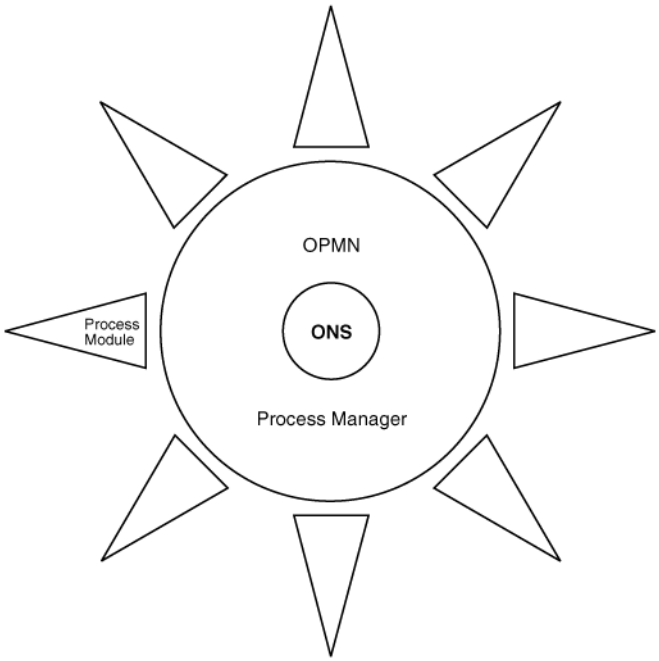
OPMN consists of a core grouping of three components that interpret and convey notification information sent between Oracle Application Server processes within the same or different OPMN servers.

The core of OPMN consists of the following three components:

- Section 2.2.1, "Oracle Notification Server"
- Section 2.2.2, "Oracle Process Manager"
- Section 2.2.3, "PM Modules"

Figure 2–1 shows the architecture of the core of OPMN.

Figure 2–1 OPMN Architecture



2.2.1 Oracle Notification Server

Oracle Notification Server (ONS) is the transport mechanism for failure, recovery, startup, and other related notifications between components in Oracle Application Server. It operates according to a publish-subscribe model: an Oracle Application Server component receives a notification of a certain type for each subscription to ONS. When such a notification is published, ONS sends it to the appropriate subscribers.

2.2.2 Oracle Process Manager

Oracle Process Manager (PM) is the centralized process management mechanism in Oracle Application Server and is used to manage Oracle Application Server processes. The PM is responsible for starting, restarting, stopping, and monitoring every process it manages. The PM handles all requests sent to OPMN associated with controlling a process or obtaining status about a process. The PM is also responsible for performing death-detection and automatic restart of the processes it manages. The Oracle Application Server processes that PM is configured to manage are specified in the `opmn.xml` file.

The PM waits for a user command to start a specific, or all Oracle Application Server processes. When a process is stopped, the PM receives a request as specified by the request parameters.

The OPMN server consists of 2 processes. The first OPMN server process has only one purpose: to start the second OPMN server process when necessary. The second OPMN server process handles all request traffic and does all the work. If the second OPMN server process goes down as part of an `opmnctl reload` command or an unexpected crash it will be restarted by the first OPMN server process.

On Microsoft Windows, the second OPMN server process will not be restarted if it is deliberately terminated. Instead, the first OPMN server process will exit as well. Recovering from this situation is accomplished by restarting the OPMN server from the command line or service manager.

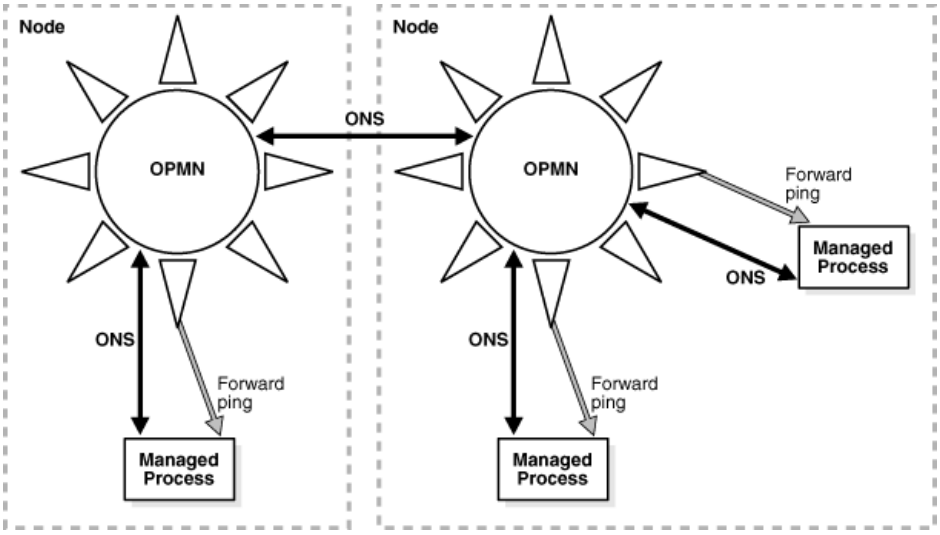
The Application Server Control Console also uses PM to manage processes.

The PM uses the ONS to:

- detect that a process has completed initialization and is ready to receive requests
- determine what ports are in use
- obtain component specific runtime information

Figure 2–2 shows ONS communication across two nodes. ONS transports notifications between the 2 nodes and sends out notifications to subscribers.

Figure 2–2 Process Management on Two Nodes



ONS uses dynamic discovery to announce new servers and join them into the ONS topology dynamically. With dynamic discovery the ONS network topology includes all of the application server instances that have been configured with the same discovery information.

OPMN automatically determines which set of ONS servers to connect to at runtime based on the topology of the ONS network. The set of ONS server insures delivery of messages throughout the ONS network, offers protection from individual link failures, and minimizes the number of connections required between servers. The number and choice of which servers each ONS server is connected to is adjusted as servers join and leave the grid. Refer to Section 3.2, "Dynamic Discovery in opmn.xml" for more information.

2.2.3 PM Modules

The Oracle Process Manager Modules (PM Modules) implement Oracle Application Server component-specific process management functionality. The PM Modules pass notification information returned by other Oracle Application Server component PM Modules within the same or different OPMN servers.

The PM Modules:

- handle any communications originating from the running component.
- construct Oracle Application Server component specific control information (how to start, stop, restart the component).
- test responsiveness in an Oracle Application Server component specific manner to determine if a component is responding to requests.

2.3 What Oracle Application Server Components Does OPMN Manage?

OPMN manages all Oracle Application Server components except the Application Server Control Console.

OPMN enables you to explicitly manage Oracle HTTP Server and Oracle Containers for J2EE (OC4J).

You can also configure OPMN to manage other processes (including Oracle and other third-party products) using the Custom PM Module. See Chapter 10, "Configuring Custom Process" for more information.

Because of the extensible design of OPMN, add-on components are managed by OPMN without having to update OPMN itself.

OPMN also enables you to manage all of the Oracle Application Server server instances in your grid environment.

2.3.1 Oracle Enterprise Manager 10*g* Application Server Control Console

In addition to OPMN, you can also manage your enterprise using the Application Server Control Console. The Application Server Control Console leverages the functionality of OPMN to manage your Oracle Application Server enterprise. Using a Web browser, Application Server Control Console provides a graphical interface that enables management of all Oracle Application Server components in your network and enterprise.

See Also: *Oracle Application Server Administrator's Guide*

3

The opmn.xml File

This chapter provides an overview of the `opmn.xml` file for Oracle Application Server. It features the following topics:

3.1 opmn.xml

The *ORACLE_HOME*`/opmn/conf/opmn.xml` file is the main configuration file for OPMN. The `opmn.xml` file contains information for PM and Oracle Application Server component specific configuration. The `opmn.xml` file shows you which Oracle Application Server components OPMN is managing on your system.

The `opmn.xml` file does not contain component-specific element names. Component specific management code is located in the PM modules which get loaded by OPMN at startup according to what has been specified in the `modules` section of the `opmn.xml` file.

Each level has a specific set of configurations. In addition, there are several configuration elements that are accepted at more than one level to provide the flexibility of applying a configuration across an entire Oracle Application Server component or just part of a component.

```
<ias-component>
   <process-type>
      <process-set>
```

`<ias-component>`: This entry represents the Oracle Application Server component. It enables management of the component for processes such as starting and stopping.

`<process-type>`: This subcomponent of the `<ias-component>` entry declares the type of process to run by association with a specific PM module.

<process-set>: This sub-subcomponent of the <ias-component> entry enables you to declare different sets of optional runtime arguments and environments for the Oracle Application Server component.

The opmn.xml file contains Oracle Application Server component entries arranged in the hierarchical structure shown in Example 3–1.

Example 3–1 Element Entries in opmn.xml File

```
<ias-component id="OC4J">
   <process-type id="home">
      <process-set id="default_group">....
```

You can edit the opmn.xml file using Application Server Control Console. Click the **Process Management** link at the bottom of the Oracle Application Server instance home page. Do not stop the OPMN server after you edit the opmn.xml file. Application Server Control Console automatically reloads the updated opmn.xml file after you edit the file.

ONS can be configured from the opmn.xml file. Example 3–2 is an example of the ons.conf element in the opmn.xml file:

Example 3–2 ons.conf Element in opmn.xml File

```
<notification-server>
   <topology>
      <nodes list="node-list"/>
      <discover list="discover-list"/>
      <gateway list="gateway-list"/>
   </topology>
```

3.2 Dynamic Discovery in opmn.xml

OPMN uses dynamic discovery to contact other ONS servers in a cluster environment. The opmn.xml file contains a multicast address or list of discovery servers that is used by OPMN. ONS uses the discovery mechanism to announce new servers to a cluster and join them into the ONS topology dynamically.

The ONS network topology includes all of the Oracle Application Server instances that have been configured with the same discovery information.

The information is configured under the notification-server element in the opmn.xml file. Example 3–3 shows the notification-server element in the opmn.xml file:

Example 3–3 Notification Server element

```
<notification-server interface="type">
   <ipaddr remote="ip" request="ip"/>
   <port local="port" remote="port" request="port"/>
   <ssl enabled="boolean" wallet-file="path" wallet-password="password"
    openssl-certfile="path" openssl-keyfile="path" openssl-password="password"
    openssl-lib="path"/>
   <tune io-timeout="timeout" io-idle="interval" timeout="timeout"/>
   <topology>
      <nodes list="nodes"/>
      <discover list="nodes"/>
      <gateway list="nodes"/>
   </topology>
```

```
</notification-server>
```

The following sections describe the three configuration types for dynamic discovery:

- Section 3.2.1, "Multi-Cast Configuration"
- Section 3.2.2, "Discovery Server Configuration"
- Section 3.2.3, "Gateway Configuration"

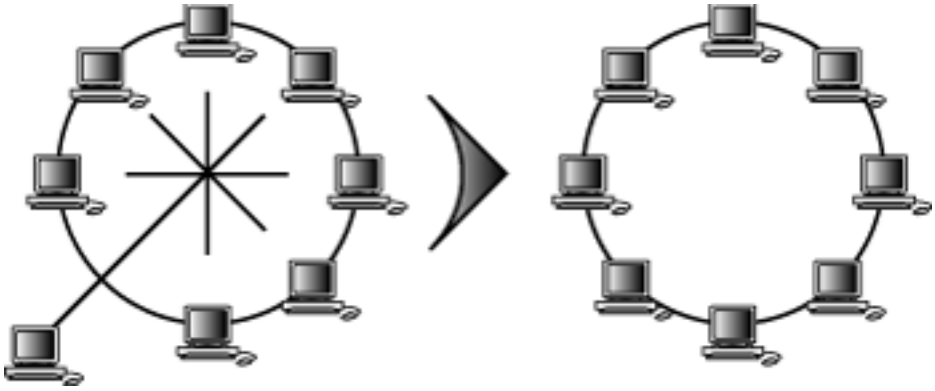
3.2.1 Multi-Cast Configuration

For multi-cast configuration of dynamic discovery, you configure a multi-cast address for all ONS servers in the `opmn.xml` file. ONS uses this address to discover all other Oracle Application Server instances in the cluster.

Figure 3–1, "Multi-cast Configuration" shows diagrammatically what occurs in the multi-cast configuration.

Any new Oracle Application Server instance in the cluster is announced using the configured multi-cast address. ONS then automatically manages the connection topology as instances are added or removed.

Figure 3–1 Multi-cast Configuration



More than one multi-cast address may be configured, but typically these would be used to isolate subnets from one another, and then gateways configured to connect them.

Refer to Chapter 6, "opmn.xml Common Configuration" for information on how to implement multi-cast configuration.

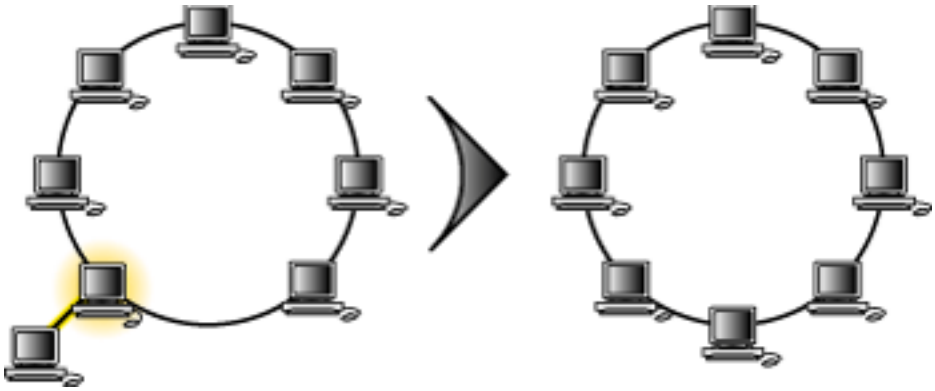
3.2.2 Discovery Server Configuration

For the discovery server configuration of dynamic discovery one Oracle Application Server instance acts as a discovery server for all instances. ONS uses the discovery server to discover all of the other Oracle Application Server instances in the cluster.

Figure 3–2, "Discovery Server Configuration" shows diagrammatically what occurs in the discover server configuration.

A new Oracle Application Server instance in the cluster contacts the configured discover server to announce itself. ONS then automatically manages the connection topology as instances are added or removed. More than one discovery server may be configured.

Figure 3–2 Discovery Server Configuration



For more information on discovery server configuration refer to Chapter 6, "opmn.xml Common Configuration".

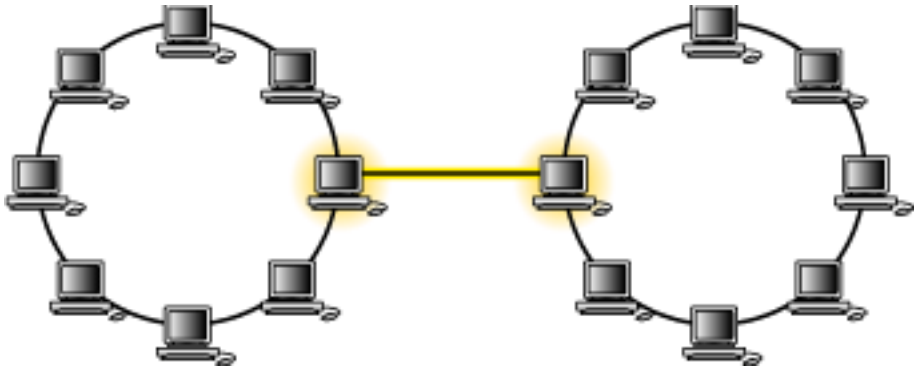
3.2.3 Gateway Configuration

For the gateway configuration of dynamic discovery, a gateway is used to interconnect multiple discovered topology rings.

Figure 3–3, "Gateway Configuration" shows diagrammatically what occurs in the gateway server configuration.

Gateway is used to interconnect multiple discovered topology rings.

Figure 3–3 Gateway Configuration



The gateway configuration is used when a network topology has nodes in different subnets or physical locations.

For more information on gateway configuration refer to Chapter 6, "opmn.xml Common Configuration"

3.3 Dynamic Resource Management

Dynamic Resource Management (DRM) is an OPMN capability designed to describe a set of desired behaviors and take actions to achieve the desired results. DRM functionality provides a way for you to customize the management of your processes through configuration changes only. The DRM enables you to have process

management commands issued based on system conditions according to a set of user-configured directives.

DRM is designed to operate on an Oracle Application Server instance. In a cluster environment, DRM functionality is available on each separate local instance.

The system conditions that can be taken into account include Dynamic Monitoring Service (DMS) metrics that are recognized by OPMN. During runtime DMS collects performance information, DMS metrics, that you can use to analyze system performance or monitor system status. This includes preset metrics that are always present as well as metrics that you define and implement. For more information about DMS refer to the *Oracle Application Server Performance Guide*.

DRM enables you to specify a set of conditions that will trigger process management commands to be automatically issued. This is accomplished using Resource Management Directives (RMDs). RMDs are a set of conditions that specify:

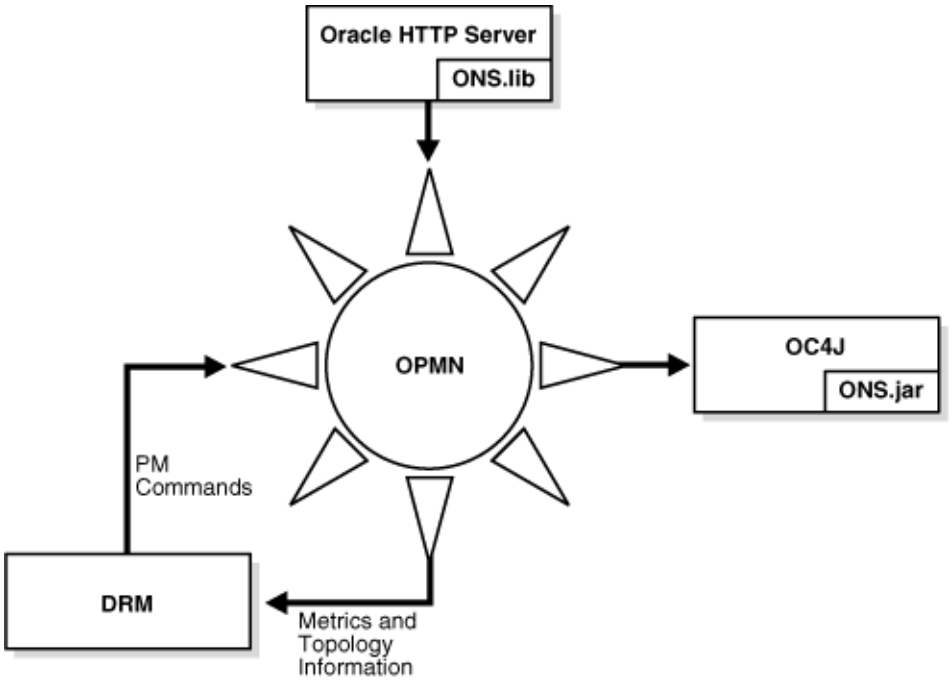
- A set of information that should be examined prior to performing an action
- The values of the set information that trigger an action
- The action that should be performed after the set of information is examined and analyzed.

RMDs enable you to configure OPMN to monitor the state of your system on a periodic basis and take action if conditions exceed the scope specified in the RMD.

For example, in the DRM graphic shown in Figure 3–4, the RMDs might be:

- At 17:00 hours every day, start two OC4J processes in anticipation of a peak load.
- If the Java Virtual Machine (JVM) heap size grow beyond 500 MB, start a second OC4J process.
- If the average response time rises above 500 milliseconds, then start an OC4J instance.

Figure 3–4 DRM Management



For more information about RMDs, refer to Section 3.3.1, "Resource Management Directives"

3.3.1 Resource Management Directives

A RMD describes a set of information that should be examined, a description of the values of this information that should trigger an action, and an action that should be performed. RMDs are comprised of seven features:

- **Metrics**: what information should be used to decide what action to take?

  For example:

  - memory usage
  - CPU utilization
  - average response time
  - request queue length
  - availability of database connections

- **Frequency**: how often should the metric values specified be examined?

  This setting enables you to control the evaluation of the RMD to be fast acting enough to react to changing system conditions but not so frequent that it will cause needless oscillation in system performance or unnecessary polling of metric values.

- **Conditional**: what rules should be applied to the metrics being examined?

  For example:

  - a comparison of a single metric with a fixed value. For example, checking if CPU utilization is above 75%.

  - a combination of values. For example, if CPU utilization is above 75%, memory usage is above 50 MB, and average response time is above 0.1 seconds.

- **Duration**: how many times or over what time period a condition must be met in order to trigger an action?

  This is used to avoid unnecessary actions when conditions are marginal. For example, the administrator may only want to take an action if CPU utilization is above 75% for more than 2 minutes to avoid allocating more resources due to a small spike in activity

- **Action**: what action should be performed when the state described by the first four items (metrics, frequency, conditional, and duration) are met?

  The action could include starting, stopping, or restarting a component. The action could also include requesting additional resources such as starting additional processes, or requesting the starting of an application on another node.

- **Exception**: what should be done if execution of the desired action fails?

  The exception could include alternative actions to perform or a message to be sent to a monitor such as Oracle Enterprise Manager 10*g*.

- **Interval**: how often should each a `process-set`, configured for the `ias-component`, be evaluated?

The DRM evaluates and performs the actions described by the RMDs.

Figure 3–5 shows DRM management on two hosts with two applications on each host, App1 and App2. Each host is sharing the application load. Each host has the capacity to run a maximum of five processes. Figure 3–5 shows that the demand for App1 is more than the demand for App2 so more processes have been started for App1 to handle the load.

Figure 3–5 DRM on 2 Hosts: App1

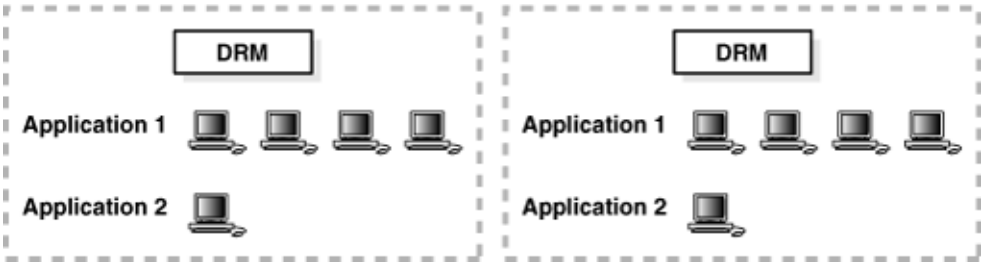


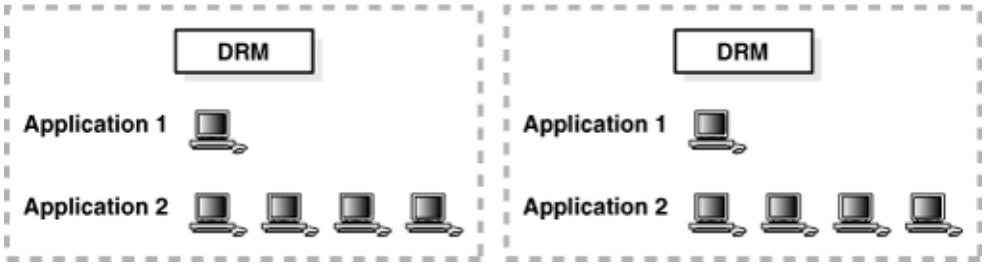
Figure 3–6 shows that as the demand for App2 increases and the demand for App1 decreases, processes servicing App1 are automatically shutdown and more processes are automatically started to service App2. Resource demand determination is made independently on each computer without coordination. Both computers detect the change in demand independently and take similar action to accommodate the change, with the overall effect of the total system adjusting to demand.

The total demand across both computers for each application varies between two and eight processes. The traditional way to provision hardware is to have enough system resources available to handle the peak demand for each application. Such a strategy would indicate having adequate system hardware to run sixteen processes.

Even with three computers capable of each running five processes, the demand on system resources for processing of the applications would not be sufficient. By

adjusting the available resources dynamically between App1 and App2 using DRM, peak load times for each application can be accommodated using only two computers.

Figure 3–6 DRM on 2 Hosts: App2



Example 3–4 shows the RMD syntax.

Example 3–4 RMD syntax

```
<rmd name="rmd name">
   <conditional>
      <!CDATA[conditional]>
   </conditional>
   <action value="action 1"/>
   <action value="action 2"/>
   ...
   <action value="action N"/>
   <exception value="exception 1"/>
   <exception value="exception 2"/>
   ...
   <exception value="exception N"/>
</rmd>
```

In Example 3–5, the RMD checks if the average request response time for its processes exceeds a maximum threshold for more than a minute, and if so it attempts to start another process if the configuration allows it.

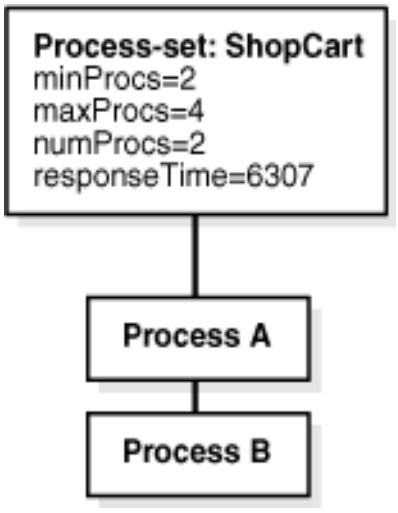
Example 3–5 RMD Response Time Syntax

```
<rmd name="rampUp">
   <conditional>
      <![CDATA[([process-set].responseTime > 5000) {duration(60)} &
       ([process-set].numProcs <[process-set].maxProcs)]]>
   </conditional>
   <action value="exec $ORACLE_HOME/scripts/logevent {ias-component} {process}
    slow response"/>
   <action value="start {process-set}"/>
   <exception value="exec $ORACLE_HOME/scripts/mailadmin {ias-component}
   {process-type} {process-set} could not start"/>
</rmd>
```

Figure 3–7, Figure 3–8, Figure 3–9, and Figure 3–10 diagram the actions of an RMD when configured parameters are surpassed.

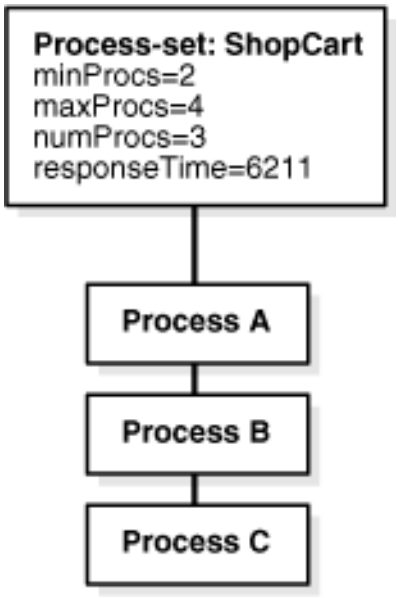
In Figure 3–7, the average request response time has exceeded the upper threshold, and the number of OC4J processes for the `process-set` is less than the configured maximum.

Figure 3–7 RMD-Upper Threshold Exceeded



In Figure 3–8, the RMD executes the actions, which entail starting another OC4J process for the ShopCart `process-set`.

Figure 3–8 Starting of Additional Process



The directive in Figure 3–8 has a complementary RMD (Example 3–6) to remove excess OC4J processes when the average request response time drops below a minimum threshold for at least five minutes.

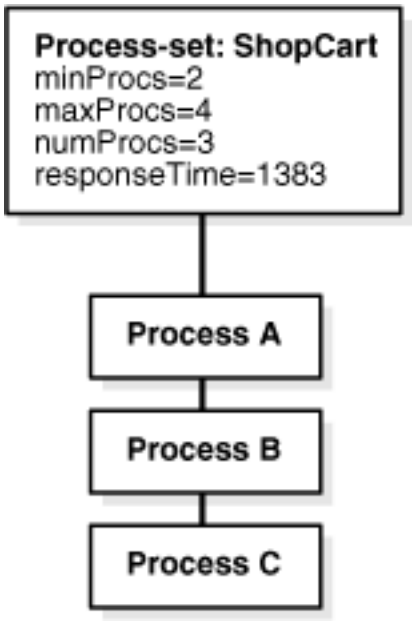
Example 3–6 Complimentary RMD

```
<rmd name="rampDown">
   <conditional>
      <![(CDATA[([process-set].responseTime < 2000) {duration(500)} &
       ([process-set].numProcs > [process-set].minProcs)]]>
   </conditional>
```

```
    <action value="exec $ORACLE_HOME/scripts/logevent {ias-component} {process}
     fast response"/>
    <action value="stop {process-set}"/>
    <exception value="exec $ORACLE_HOME/scripts/mailadmin {ias-component}
     {process-type} {process-set} could not stop"/>
</rmd>
```

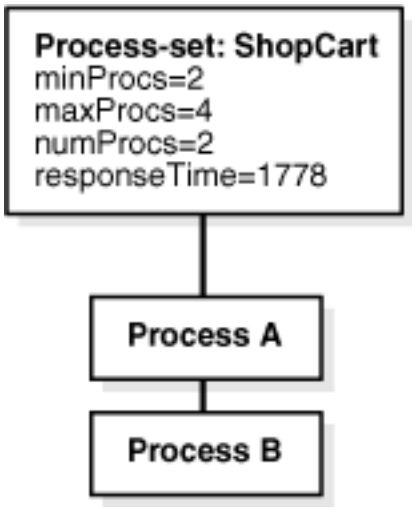
In Figure 3–9, the average request response time has been less than the lower threshold for at least five minutes, and the number of OC4J processes for the ShopCart `process-set` is more than the configured minimum.

Figure 3–9 Number of Processes exceeds Configured Minimum



In Figure 3–10, the RMD executes the actions, which stop an OC4J process for the ShopCart `process-set`.

Figure 3–10 RMD Stops a Process to Complete Request



3.3.2 RMD Configuration

RMDs can be configured in two ways. They can either be:

- associated with a specific hierarchy level in the components that are managed by OPMN (for example, a specific `ias-component` or `process-set`)
- global entities

RMDs associated with a specific place in the hierarchy can only specify metrics and actions that are associated with the RMD relative to their location in the hierarchy. Each RMD has an evaluation level, which specifies the lowest OPMN component level referenced by the directive, and RMDs which are defined at a higher level (`ias-component`, for example) will be inherited by any lower level components (`process-set`, for example) they reference. For example, an RMD defined at the `ias-component` level that references `process-set` will be inherited by all `process-sets` for the `ias-component` under which it was defined.

Global RMDs are configured in an independent section of the `opmn.xml` file. They can use metrics from anywhere in the DMS tree and can take action on any components managed by OPMN. For more information about DMS refer to the *Oracle Application Server Performance Guide*.

3.3.2.1 RMD Conditionals

The RMD conditional describes a state of the system that will trigger an action. The conditional consists of a logical combination of comparisons between values. Like all other aspects of OPMN, conditionals are case sensitive. The values that may be used can be:

- DMS metrics available from the OPMN DMS tree

  For more information about DMS refer to the *Oracle Application Server Performance Guide*.
- Constant values (for example, 500000)
- Temporal values (for example, 5 PM)

DMS metrics are described based on their location in the OPMN DMS tree. This description can be:

- a fully qualified path
- a hierarchical relative reference
- a global absolute reference

A fully qualified path to the metric desired such begins with a "`/` " and includes the full path from the OPMN instance, `/pm/host_statistics/freePhysicalMem`. The OPMN instance prefix is automatically added to these paths.

Metrics with fully qualified paths can be referenced by both hierarchical and global RMD conditionals.

If the RMD is hierarchical the metric can take a relative path form such as `[process-set].numProcs` which specifies the `numProcs` metric for any `process-set` in the hierarchy to which this RMD belongs.

The following component specifications are allowed for hierarchical RMD conditionals:

- `[ias-component]`: refers to the `ias-component` element within the hierarchy of the `opmn.xml` file.

- [process-type]: refers to the process-type within the hierarchy of the opmn.xml file.
- [process-set]: refers to the process-set within the hierarchy of the opmn.xml file.
- [process]: refers to the process within the run time hierarchy.
- [application]: refers to the application within the run time hierarchy. Unlike all other hierarchical specifications, a specific application can be named.

If the RMD is global the metric can be an absolute description of a starting point followed by a relative path. For example:

```
[ias-component=WebCache][process-set=WebCache].numProcs
```

which specifies the metric numProcs for the process-set Webcache that belongs to the ias-component WebCache.

The following component specifications are allowed for global RMD conditionals:

- [ias-component=<comp>]: refers to the ias-component <comp> in the opmn.xml file.
- [process-type=<ptype>]: refers to the process-type <ptype> in the opmn.xml file.
- [process-set=<pset>]: refers to the process-set <pset> in the opmn.xml file.
- [process]: refers to the process.
- [application]: refers to the application.
- [application=<app>]: refers to the application <app> running in the process.

As with OPMN process requests, components not specified in the directive are assumed to be wildcards; thus a global RMD referencing [process-set=home] would be evaluated for every process-set with the id home configured in the opmn.xml file.

No conditional is allowed to reference more than one OPMN component. For example, the same conditional cannot reference [ias-component=Webcache] and [ias-component=HTTP_Server]. Similarly, references to lower configuration components must belong to any referenced higher component within the same conditional.

With the exception of process, the sections surrounded by square brackets must match a DMS tree type as shown in a dump of the OPMN DMS tree.

For both hierarchical and global RMDs multiple bracketed sections can be specified but each subsequent section must further narrow the tree. The order must progress from the top of the tree toward the leaf. A process cannot appear before an ias-component.

Constant values must be simple, numeric values, including decimal values. When comparing a constant value with a DMS metric, the constant must have the same units as the DMS metric it is being compared to. The DRM will convert the numeric value on the right hand of an operator to the same type (decimal or integer) on the left, but no other unit conversions will be performed by the DRM.

Temporal values consist of a 24 hour format (hh:mm) time combined optionally with a day of the week indicator. The allowed day of the week abbreviations are mon, tue,

wed, thu, fri, sat, sun. Day of the week indicators can be a single value, a comma separated list, or a dashed inclusive list. For example:

- 17:00 Daily at 5 PM
- [mon-fri].21:45 weekdays at 9:45 PM
- [mon,wed,fri].5:00 Monday, Wednesday, Friday at 5 AM

The current time is represented by the key word {time}. A time comparison is evaluated false if the day of the week specified on the left of the operator does not include a day of the week specified on the right; by default the day of the week is only evaluated as equals and only if this is true is the time value itself evaluated for the specified operator. You can force the comparison to be made against a single day of the week by appending the @ character after the day, with Sunday (sun) having the value 0 and Saturday (sat) 6.

- {time} < [wed].12:00: evaluates true only on Wednesday until noon
- {time} < [wed@].12:00: evaluates true from Sunday at 00:00 until Wednesday at noon

Comparisons between values include the following: less than (<), less than or equal to (<=), greater than (>), greater than or equal to (>=), equals (=), or not equals (!=).

String values may also be specified between quotes (using the single quote character ("), but the only allowable operators are equals (=) and not equals (!=).

Comparisons may be logically combined using the operators and (&), or (|), not (!), and grouping (( )). The logical not unary operator (!) can be placed before any logical group of comparisons to logically negate the result of the outcome.

The key word {duration(value)} may also be used to indicate that a conditional should only trigger if it has evaluated as true over the specified time interval expressed in seconds. It is important to note that the conditional is only evaluated on a periodic basis. Specifying the duration value will force the conditional to only trigger if all evaluations over the time period meet the conditional. This is an approximation of ensuring that the condition holds true over the entire time period. The accuracy of this approximation depends on the ratio of the duration value to the evaluation period. When a conditional is evaluated as true, all durations for that evaluation are reset. When a conditional evaluates false, any duration that was not encountered during the evaluation is reset.

The following RMD conditional examples detect specific states:

- If the heap size of a JVM has exceeded 500 Megabytes (global RMD):

  ```
  [process-set=home][process].heapSize > 500000
  ```

- If the time is 5 PM on a weekday

  ```
  {time} = [MonFri].1700
  ```

- If the average request time is greater than 500 ms for at least 60 seconds and there are less then 4 processes running for the for the process-set at which this hierarchical RMD was configured (OC4J):

  ```
  ([process].avgReqTime > 500
  {duration(60)})&([process-set].numProcs <4)
  ```

- If there has been less than 50 megabytes of free system memory for the last 3 minutes:

  ```
  /pm/host_statistics.freePhysicalMemory < 50000
  {duration(180)}
  ```

3.3.2.2 RMD Actions

The Action section of an RMD is a list of action elements, each of which specifies a process management command that describes an action that should be performed if the conditional of the RMD is satisfied. The syntax of these commands is similar to that of `opmnctl` without the scope.

The possible actions for an RMD are:

- `start`: perform a start request with the given argument list
- `restart`: perform a restart request with the given argument list
- `stop`: perform a stop request with the given argument list
- `exec`: execute the given program or script with the listed arguments

The targets for the `start, restart`, and `stop` requests is assumed to be relative to the OPMN components referenced in the RMD conditional. Key words representing these components should be used to narrow the scope of the request.

The following set of key words are available:

- `{ias-component} ias-component=<comp>`
- `{process-type} process-type=<ptype>`
- `{process-set} process-set=<pset>`
- `{process} uniqueid=<uid>`
- `{application} application=<app>`

For `start, restart`, and `stop` actions, the key word `{process}` should not be used with the other keywords.

In addition the `exec` action can use the `{pid}` key word, which equates to `pid=<pid>` if the conditional refers to a `process-set` or process.

Note that an action cannot reference a key word for an OPMN component at a level below the lowest OPMN component referenced in the conditional (if the conditional only references an `ias-component`, then the action can only use the `{ias-component}` key word, for example). The exception to this rule is that an action may reference `{process}` if the conditional references a `process-set`, but this will force the RMD to be evaluated at the process level and not the `process-set` level.

If the `start, restart`, and `stop` request returns a non successful status (200 is success) or an `exec` exits with any code other than 0, then the remaining actions are skipped and any exceptions configured for the RMD are executed.

A timeout value can be configured for each action. The default timeout for `start, restart`, and `stop` actions is the configured (or default) timeout for the OPMN request. The default timeout for `exec` actions is 30 seconds.

The results of the RMD requests are logged in the OPMN process manager log (the beginning and completion of the request with completion status at level 4, and full results at level 5). The `stdout` and `stderr` of `exec` programs or scripts is sent to *`$ORACLE_HOME`*`/opmn/logs/rmd.out`. Note that there is no rotation performed on the `rmd.out` file. Therefore, programs and scripts should maintain and use their own log files. Programs and scripts should not print output to either of the `stdout` and `stderr` file descriptors.

The following are RMD Action Examples:

- Start another JVM in the home OC4J instance within a hierarchical RMD defined within the home instance (note this `process-set` must be configured with `minprocs/maxprocs`):

  ```
  start {ias-component}{process-type}{process-set} numprocs=1
  ```

- Restart a JVM in the home instance within a hierarchical RMD defined within the home instance:

  ```
  restart {process}
  ```

- Stop the entire `ias-component` referred to by the RMD conditional:

  ```
  stop {ias-component}
  ```

- Execute the given program and pass in the referenced process UID and pid:

  ```
  exec $ORACLE_HOME/mybin/report.sh "RMD triggered" {process}
  {pid}
  ```

3.3.2.3 RMD Exceptions

The Exception section of an RMD is a list of process management commands that are to be performed in the case that any of the process management commands in the action section of the RMD fail to execute normally. The format of the exception section is identical to that of the action section.

3.3.2.4 RMD Configuration in the `opmn.xml` file

RMDs are configured in the `opmn.xml` file in a section entitled `rmd-defintions`. The conditional, action, and exception portions of each RMD are attributes in the `.xml` definition. Example 3–7 shows how an RMD would be defined as a hierarchical RMD in the `opmn.xml` file and Example 3–8 shows how an RMD would be defined as a global RMD.

Example 3–7 Hierarchal RMD Example

```
<process-type id="home" moduleid="OC4J" status="enabled">
  .
  .
  .
  <rmd-defintions>
    <rmd name="requesttime" description="Start another OC4J (if possible) when the
     response time of any exiting OC4J in home exceeds 500msecs for a minute"
     interval="30">
      <conditional>
      <![CDATA[([process].avgReqTime > 500 {duration(60)})&([process-set].numProcs
      < 4)]]>
      </conditional>
      <action value="start {ias-component}{process-type}{process-set}
numprocs=1"/>
      <exception value="exec $ORACLE_HOME/mybin/mailer.sh {ias-component}
       {process-type}{ process-set} failed to start on RMD request"/>
    </rmd>
  </rmd-defintions>
</process-type>
```

Example 3–8 Global RMD Example

```
<process-manager>
  .
```

```
  .
  .
  <rmd-defintions>
    <rmd name="requesttime" description="Start another OC4J (if possible) when the
     response time of any exiting OC4J in home exceeds 500msecs for a minute"
     interval="30">
      <conditional>
      <![CDATA[([process].avgReqTime > 500 {duration(60)})&([ias-component=OC4J]
       [process-set=home].numProcs < 4)]]>
      </conditional>
      <action value="start {ias-component}{process-type}
       {process-set}.numprocs=1"/>
      <exception value="exec $ORACLE_HOME/mybin/mailer.sh {ias-component}
       {process-type}{process-set}failed to start on RMD request"/>
    </rmd>
  </rmd-defintions>
</process-manager>
```

3.3.2.5 RMD Evaluation

RMDs will be evaluated periodically based upon their configured interval or the default value of 30 seconds. Note that there is a trade off between CPU cycles consumed evaluating RMDs and the sensitivity of the execution of RMDs. The evaluation period and the time value defined by the `{duration()}` key word should relate in a reasonable ratio.

RMDs are evaluated based upon where they are configured and what their conditionals reference. A global RMD that only references full paths to metrics or the internal temporal value is truly global and only a single evaluation is performed for each configured interval. A hierarchical RMD that is defined under an `ias-component`, but references a `process-set` will be evaluated once for each interval for each `process-set` configured for the `ias-component`. Any RMD that references a process in either its conditional, actions or exceptions will be evaluated once for each interval for each process that exists under the components referenced by the RMD. Note that RMDs that reference a process will only be evaluated if a process exists and is in the Alive state.

DMS metrics are retrieved at evaluation time. If a referenced DMS metric cannot be found, that part of the conditional will evaluate as false.

Invalid type comparisons for metrics are discovered at evaluation time. When the invalid type comparisons are encountered, the current evaluation is aborted, the RMD is disabled, and all other evaluations based upon this RMD are aborted. An error is logged in the OPMN process manager log.

3.4 Service Failover

Service failover is mechanism to specify a critical process that must be run somewhere in an Oracle Application Server cluster if service is disrupted on a processing server. This enables you to preferentially select which processes must be kept running.

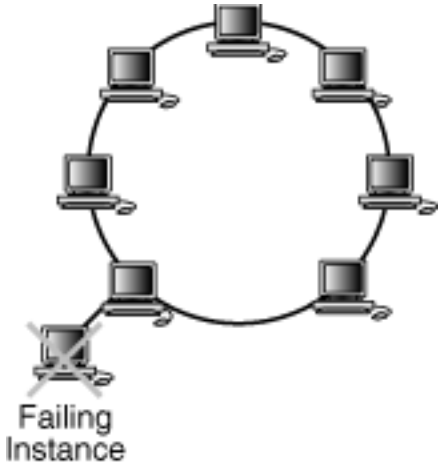
Any `process-type` `opmn.xml` file element may be configured as a service fail-over such that, once started, OPMN will ensure that the configured number of processes for the service are running on Oracle Application Server instances somewhere in the cluster.

You can configure which Oracle Application Server instances will participate in the service fail-over on an instance by instance basis. You can configure each instance for preferential selection of running the process on available instances.

Only one `process-set` may be defined for each `process-type` configured as a service fail-over. Only one process will be run for each service fail-over instance.

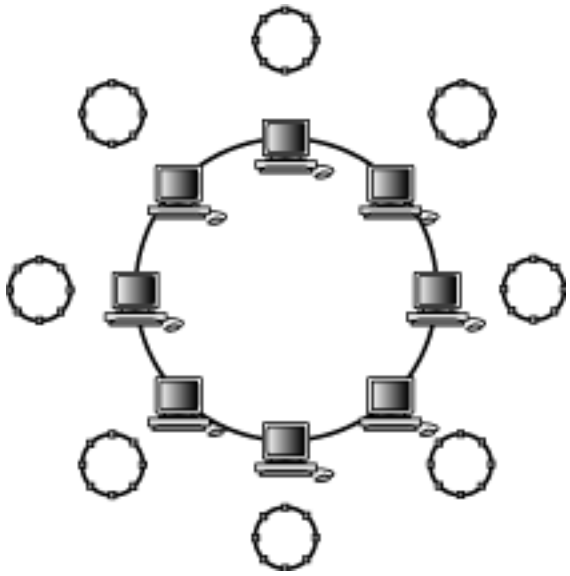
In the following Figure 3–11, a service fail-over process has been started in a cluster where all instances are configured to participate in the service fail-over.

Figure 3–11 Start of Service Failover



As shown in Figure 3–12, if the instance on which the service fail-over process is running goes down, such as for maintenance or an unprotected power outage or network failure, OPMN will select another participating Oracle Application Server instance on which to run the process. All of the instances shown in Figure 3–12 are participating in the service fail-over.

Figure 3–12 Operative Service Failover



3.5 Automatic Restart

OPMN gives the user control over automatic death detection and restart of components; you can configure the parameters by which OPMN determines a process has died and disable automatic restart for individual components.

OPMN monitors the operation of its managed processes by the following methods:

- Operating system level detection of Oracle Application Server process death
- Periodic ping requests to Oracle Application Server processes
- Periodic status notification from Oracle Application Server processes (reverse-ping)

The ping and notification functionality is only used where appropriate according to the functionality of the Oracle Application Server component.

OPMN automatically restarts Oracle Application Server components that terminate unexpectedly. OPMN will also restart processes that are unresponsive according to the result of notification and ping operations.

See Also:

- Chapter 6, "opmn.xml Common Configuration"

3.6 Event Scripts

You can configure OPMN to execute your own custom event scripts whenever a particular component starts, stops, or crashes. You can select from one or more of the following event types:

- **pre-start**: OPMN runs the pre-start script after any configured dependency checks have been performed and passed, and before the Oracle Application Server component starts. For example, the pre-start script can be used for site-specific initialization of external components.
- **pre-stop**: OPMN runs the pre-stop script before stopping a designated Oracle Application Server component. For example, the pre-stop script can be used for collecting Java Virtual Machine stack traces prior to stopping OC4J processes.
- **post-crash**: OPMN runs the post-crash script after the Oracle Application Server component has terminated unexpectedly. For example, a user could learn of component crashes by supplying a script or program to be executed at post-crash events which sends a notification to the administrator's pager.

See Also:

- Section A.2.4, "Troubleshooting with Event Scripts"
- <event-scripts> in Chapter 6, "opmn.xml Common Configuration"

3.7 Start Order Dependencies

Some Oracle Application Server components and services require that other components and services are up and running before starting. OPMN is configured at installation with default start order dependencies, which enables you to start all of the components in an instance in the proper order with a single command. Refer to the *Oracle Application Server Administrator's Guide* for more information on Oracle Application Server dependencies.

OPMN is configured with a set of dependencies but you can configure additional dependencies according to the environment

3.8 OPMN Log Files

The log files generated by OPMN provide important information that can help you identify and diagnose performance and configuration issues. The Application Server Control Console makes reviewing these log files easier by helping you locate and view Oracle Application Server component log files.

See Also:

- Section A.2.1, "OPMN log Files"
- *Oracle Application Server Administrator's Guide*

3.9 Security

The OPMN local listener port used by ONS clients and PM administrative processes do not use Secure Socket Layer (SSL) encryption for security, but rely on two other mechanisms to ensure authorized access to the OPMN server:

- OPMN binds the local listener port to the local host. Users on the local system can connect to this port and issue OPMN process control requests. Information requests are allowed on the OPMN request port, which is bound to the system IP. The request port does not have SSL encryption.
- When the OPMN server process first starts up and successfully binds to the local port, it creates a string of printable ASCII characters which it uses as a key for local connections. All connection attempts on the local port must include this key or the connection is closed by the OPMN server. The ASCII character string is written into the *ORACLE_HOME*`/opmn/conf/.formfactor` file. Processes that cannot access the `.formfactor` file are not permitted to interact with the OPMN server.

For security reasons, the OPMN server logs any attempts to connect to its local port with an invalid form factor key (a key that does not match the value written by this OPMN process into the `.formfactor` file).

In addition to attempted security violations, there are four common user errors that can cause this error to occur:

- The user attempts to run the OPMN client manually with the wrong user identification. Only the application server user can read the value from the `.formfactor` file, and so requests or processes run as the wrong user will not be able to provide the correct key to the OPMN server.
- The user is attempting to run an OPMN client from the wrong *ORACLE_HOME*. It is possible to have multiple *ORACLE_HOME* instances set up on the same system. If the other *ORACLE_HOME* instances have OPMN configured to use the same local port then the Oracle Application Server process request from the wrong *ORACLE_HOME* will read the wrong `.formfactor` file.
- The user has manually changed the local port configuration in the `opmn.xml` file and started a new OPMN server without first stopping the previous OPMN server. The new OPMN server will run, bind to the new port, and overwrite the `.formfactor` file. The previous OPMN server is now unreachable through the local port, and can only be shutdown through remote OPMN requests (if SSL and authentication are configured) or by manually stopping the previous OPMN server.

- The Oracle Application Server and the Oracle Database both use ONS. When these two products are installed onto the same host, an ONS port conflict arises since the default port values (local="6100" remote="6200") for ONS are the same for both the Oracle Database and Oracle Application Server.

  ONS with the Oracle Database is only used for special configurations and therefore is typically never started. However, the database listener will attempt to connect to the Database ONS server but will end up connecting to the ONS server that was installed with Oracle Application Server. ONS (as part of OPMN) is always started whenever Oracle Application Server is started.

  Because the Oracle Database is installed in a different *ORACLE_HOME* than that of Oracle Application Server, the Database ONS does not have access to the `.formfactor` file that was created when OPMN started up with the Oracle Application Server. As a result, the database listener attempts to connect to OPMN; the DB listener interprets it as a its standalone ONS) without a form factor string. Oracle Application Server OPMN logs an error similar to the following in the `ons.log` file:

  ```
  04/11/15 18:43:32 [4] Local connection 0,127.0.0.1,6100 invalid form factor
  ```

  This is expected OPMN behavior Oracle Application Server; preventing client access to the ONS server unless they possess the correct formfactor string.

  To avoid having the Oracle Database listener contact the Oracle Application Server OPMN server, change the default local and remote port values for the ONS server that was installed with the Oracle Database. Alternatively, you can apply the latest Oracle Database patchset available on OTN:

  ```
  http://www.oracle.com/technology/products/
  ```

3.9.1 Remote Security

OPMN supports remote requests to other OPMN servers in the same cluster, but for security reasons all process control requests (start, restart and stop) are only enabled if SSL is enabled in the `opmn.xml` file and a wallet file is configured. If neither SSL nor a wallet file are configured, OPMN will reject any remote process control request with HTTP code 403.

The remote port used for remote administration must be SSL-enabled. The remote port should only be used for communication between multiple OPMN servers. Oracle Application Server components and Application Server Control Console transmit through the local port which is inaccessible to remote administration. All access control and authentication is controlled by going through Application Server Control Console.

During Oracle Application Server installation, OPMN will be configured to use a wallet that contains a default certificate. For secure operation, the certificate used must be replaced with a secure, unique certificate. Refer to the *Oracle Application Server Security Guide* for information on how to configure certificates within an Oracle Wallet.

The `orapki` script, which is documented in the *Oracle Application Server Security Guide*, can be used to generate a random, self-signed certificate that is sufficient to provide the appropriate level of security for OPMN.

Note that use of OPMN in a cluster requires configuration with certificates that are trusted by all other members of the cluster. This can be accomplished by either using the same wallet in all OPMN instances in the cluster or using certificates created by a common certificate authority.

See Also: *Oracle Application Server Security Guide*

3.10 IPv6 Support

In 10.1.3, ONS is able to concurrently support the IPv4 and IPv6 network interfaces.

IPv4 is version 4 of the Internet Protocol (IP). It was the first version of the Internet Protocol to be widely deployed, and forms the basis for most of the current Internet (as of 2004).

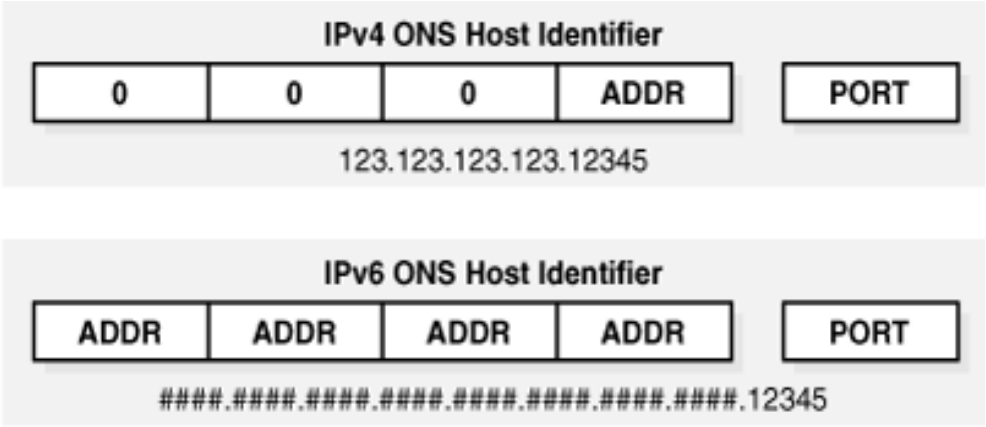
IPv4 uses 32-bit addresses, limiting it to 4,294,967,296 unique addresses, many of which are reserved for special purposes such as local networks or multicast addresses, reducing the number of addresses that can be allocated as public Internet addresses.

IPv6 is intended to address the concern that there are too few IP addresses available for the future demand of device connectivity (especially cell phones and mobile devices). IPv6 supports 340 undecillion (3.4 × 1038) addresses.

As shown in Figure 3–13, for output, such as debug or log records, each IPv4 identifier will be displayed as four, eight bit fields for the address (each a three digit decimal format) and a single 16 bit field for the port (a five digit decimal format).

Each IPv6 identifier will be displayed as eight 16 bit fields for the address (each a four digit hexidecimal format) and a single 16 bit field for the port (the five digit decimal format).

Figure 3–13 IPv4 and IPv6 Host Identifier



4

opmnctl Commands

This chapter provides an overview of `opmnctl` commands for Oracle Application Server components managed by OPMN. It features the following topics:

- Section 4.1, "opmnctl"
- Section 4.2, "opmnctl Command Quick Reference"
- Section 4.3, "opmnctl Detailed Command Description"

4.1 opmnctl

`opmnctl` is the supported tool for starting and stopping all components in an Oracle Application Server instance, with the exception of the Oracle Enterprise Manager 10*g* Application Server Control Console (Application Server Control Console). `opmnctl` provides a centralized way to control and monitor Oracle Application Server components from the command line. You can use `opmnctl` to execute control and monitoring commands across multiple Oracle Application Server instances simultaneously.

`opmnctl` also enables you to perform operations on a specified Oracle Application Server instance in a cluster or all instances in a cluster using an optional parameter called `scope`. You can also use the `scope` option to control an individual Oracle Application Server process.

The `opmnctl` command is located in the following directory locations:

(Linux) *`ORACLE_HOME`*`/opmn/bin/opmnctl`

(Microsoft Windows) *`ORACLE_HOME`*`\opmn\bin\opmnctl`

Note: Oracle Application Server components managed by OPMN should never be started or stopped manually. Do not use command line scripts or utilities from previous versions of Oracle Application Server for starting and stopping Oracle Application Server components. Use the Application Server Control Console and the `opmnctl` command line utility to start or stop Oracle Application Server components.

Note: Oracle recommends starting OPMN as the user that has installed Oracle Application Server.

4.1.1 opmnctl Syntax

The following command shows an example of the syntax of the `opmnctl` command:

```
opmnctl [verbose] [<scope>] <command> [<options>]
```

Table 4–1 provides a description about `opmnctl` syntax.

Table 4–1 opmnctl Syntax

Syntax	Description
verbose	Prints detailed execution message, if available.
scope	Specifies where the request is routed. Refer to Section 4.3.1.1, "Scope" for a list of options.
command	Specifies an `opmnctl` command. Refer to Example 4–1 for a list of commands.
options	Specifies options for the command. Refer to Section 4.3.4.1.1, "Options for the Status Command of opmnctl" for a list of options.

4.2 opmnctl Command Quick Reference

Example 4–1 lists `opmnctl` commands for quick reference. You can obtain the same output information by executing the `opmnctl help` command.

Example 4–1 opmnctl Commands

```
prompt > opmnctl help

  scope      command      options
_________   __________   __________
            start                        - Start opmn
            startall                     - Start opmn and all managed processes
            stopall                      - Stop opmn and all managed processes
            shutdown                     - Shutdown opmn and all managed processes
[<scope>]   startproc    [<attr>=<val>..]  - Start opmn managed processes
[<scope>]   restartproc  [<attr>=<val>..]  - Restart opmn managed processes
[<scope>]   stopproc     [<attr>=<val>..]  - Stop opmn managed processes
[<scope>]   reload                         - Trigger opmn to reread opmn.xml
[<scope>]   status       [<options>]       - Get managed process status
[<scope>]   dmsdump      [<attr>=<val>&..] - Get DMS stats
[<scope>}   set          [<attr>=<val> ..] - Set opmn log parameters
[<scope>]   query        [<attr>=<val> ..] - Query opmn log parameters
            ping         [<max_retry>]     - Ping local opmn
            validate     [<filename>]      - Validate the given xml file
            config       [<options>]       - Modify the opmn xml file
            help                           - Print brief usage description
            usage        [<command>]       - Print detailed usage description
```

4.3 opmnctl Detailed Command Description

The following sections contains detailed descriptions of the `opmnctl` commands listed in Example 4–1. The `opmnctl` commands are displayed in the following sections:

- Section 4.3.1, "Command Definitions"
- Section 4.3.2, "Server Control Commands"
- Section 4.3.3, "Process Control Commands"

- Section 4.3.4, "Status Commands"
- Section 4.3.5, "Help Commands"

4.3.1 Command Definitions

`opmnctl` features command definitions that enable you to further define the action you would like to execute with OPMN.

This section describes the command definitions available with the `opmnctl` command. It includes the following sections:

- Section 4.3.1.1, "Scope"
- Section 4.3.1.2, "Attributes"
- Section 4.3.1.3, "Verbose"

4.3.1.1 Scope

Syntax: `@instance[:instname[:instname...]]`

`@cluster[:clusname[:clusname...]]`

The `scope` option specifies which Oracle Application Server instances the `opmnctl` command applies to. You can use the scope option for `opmnctl` commands for single or multiple Oracle Application Server instances and clusters.

- **`@instance:`** If you do not specify a name after `@instance` option, the `opmnctl` command is applied to the local Oracle Application Server instance; local refers to the Oracle Application Server instance or cluster containing the OPMN server handling the request. The default is the local Oracle Application Server instance. If the `@instance` option is followed by Oracle Application Server instance names, the request will be routed to Oracle Application Server instances. To apply the command to one or more Oracle Application Server instances, specify `@instance[:instname[:instname...]]`.
- **`@cluster:`** If you do not specify a name after `@cluster` option, the `opmnctl` command is applied to the local Oracle Application Server cluster. If `@cluster` is followed by a set of 1 or more cluster names, the request will be routed to the all Oracle Application Server instances contained in the specified Oracle Application Server clusters. To apply the command to all Oracle Application Server instances within one or more Oracle Application Server clusters, specify `@cluster[:clusname[:clusname...]]`.

For example, the following command starts OC4J on Oracle Application Server instance named "`myInst2.foo.com`":

```
prompt > opmnctl @instance:myInst2.foo.com startproc ias-component=HTTP_Server
```

See Also: Chapter 5, "Using OPMN"

4.3.1.2 Attributes

syntax: `<attribute>=<value>`

The `opmnctl` attributes enable you to apply process control operations to specific Oracle Application Server components.

For example, the following command starts all Oracle Application Server processes configured for OracleAS Wireless:

```
prompt > opmnctl startproc ias-component=wireless
```

Refer to Chapter 5, "Using OPMN" for additional `opmnctl` command examples.

Table 4–2 lists the attribute names and values that can be used with the `opmnctl` command:

Table 4–2 opmnctl Attribute Names and Values

Attribute Name	Attribute Values
`ias-component`	Value should be the same as the value for the `id` attribute for the `<ias-component>` element in the `opmn.xml` file.
`process-type`	Value should be the same as the value for the `id` attribute for the `<process-type>` element in the `opmn.xml` file.
`process-set`	Value should be the same as the value for the `id` attribute for the `<process-set>` element in the `opmn.xml` file.
`mode`	Value can either be `sync` or `async`. The default value is "`sync`, meaning that this request operates synchronously, and waits for the operation to complete before returning. "`async` indicates that the request returns immediately, while OPMN continues to perform the request until the operation finishes.
`timeout`	This can only be specified in sync mode. The value is in seconds. After this timeout expires, OPMN does not continue to perform the request for `startproc` operations. The request does continue for `restartproc` and `stopproc` operations.
`uniqueid`	This value is assigned by OPMN after starting up. You can use this value when you execute the `opmnctl restartproc` and `opmnctl stopproc` commands.

See Also: Chapter 5, "Using OPMN"

4.3.1.3 Verbose

Syntax: `opmnctl verbose command`

The `opmnctl verbose` option enables you to obtain detailed information about the command you are executing.

For example, the following command outputs the information shown in Example 4–2:

```
prompt> opmnctl verbose startproc ias-component=HTTP_Server
```

Example 4–2 opmnctl verbose output

```
HTTP/1.1 200 OK
Content-Length: 0
Content-Type: text/html
Response: Ping succeeded.

opmnctl: starting opmn managed processes...
HTTP/1.1 200 OK
Content-Length: 571
Content-Type: text/html
Response: 1 of 1 processes started.

<response>
<opmn id="jerichar-sun.us.oracle.com:6200" http-status="200" http-response="1 of 1
 processes started.">
   <ias-instance id="M140801.jerichar-sun.us.oracle.com">
```

```
      <ias-component id="HTTP_Server">
         <process-type id="HTTP_Server">
            <process-set id="HTTP_Server">
               <process id="1954086921" pid="9355" status="Alive" index="1"
                log="/home/demoas/M140801/opmn/logs/HTTP_Server~1"
               operation="request" result="success">
               </process>
            </process-set>
         </process-type>
      </ias-component>
   </ias-instance>
</opmn>
</response>
```

4.3.2 Server Control Commands

The `opmnctl start, startall, reload, stopall,` and `shutdown` commands enable you to control the OPMN server.

- Section 4.3.2.1, "Server Control Commands on Microsoft Windows"
- Section 4.3.2.2, "opmnctl start"
- Section 4.3.2.3, "opmnctl startall"
- Section 4.3.2.4, "opmnctl stopall"
- Section 4.3.2.5, "opmnctl shutdown"
- Section 4.3.2.6, "opmnctl reload"

Output is not generated for the successful execution of an `opmnctl` server control command. Refer to Appendix A, "OPMN Troubleshooting" if you receive any error messages during `opmnctl` command execution.

4.3.2.1 Server Control Commands on Microsoft Windows

On the Microsoft Windows operating system, OPMN is installed as a Windows service `(Oracle<OracleHomename>ProcessManager)` and it starts up automatically when you restart your computer. When you start or stop OPMN using Windows Services you start or stop *all* OPMN-managed components on the local instance.

Use the Application Server Control Console and the `opmnctl` command line utility to start or stop Oracle Application Server components.

4.3.2.2 opmnctl start

Syntax: `opmnctl start`

Use this command to start the OPMN server for a local Oracle Application Server instance without starting OPMN-managed processes.

Execute this command as soon as possible after starting your computer.

Note: OPMN starts up automatically on Microsoft Windows when you start or restart your computer. All OPMN-managed processes are also started.

See Also: Chapter 5, "Using OPMN"

4.3.2.3 opmnctl startall

Syntax: `opmnctl startall [timeout=<seconds>]`

Use this command to start OPMN as well as the OPMN managed processes for a local Oracle Application Server instance. The `startall` is equivalent to the `start` command and the `startproc` command without arguments. Oracle recommends using the `start` or `startproc` command.

This command operates synchronously and waits for the operation to complete before returning. To set a timeout for the request, specify the timeout value in seconds.

Components with `id-matching="true"` will not be started.

Enter the following command for additional detailed information:

```
prompt > opmnctl usage startall
stopall
start startall startproc
```

On Microsoft Windows, you can also perform an `opmnctl startall` by starting the `Oracle<OracleHomename>ProcessManager` service in the Windows services control panel. The `Oracle<OracleHomename>ProcessManager` starts automatically when you start or restart your computer.

4.3.2.4 opmnctl stopall

Syntax: `opmnctl stopall`

Use the `opmnctl stopall` command to shut down the OPMN server as well as the OPMN-managed processes for the local Oracle Application Server instance. This request operates synchronously; it waits for the operation to complete before returning.

Shutting down the OPMN server is not necessary during normal operation. Shutting down the OPMN server prevents remote commands to OPMN from executing on the Oracle Application Server instance until OPMN is restarted.

The `opmnctl stopall` command should only be executed prior to shutting down OPMN and your computer. This request first tries to stop all OPMN-managed processes gracefully. Processes which will not stop gracefully will be forcefully shutdown. After stopping all managed processes, the OPMN daemon will shutdown itself.

The `opmnctl stopall` command should only be used when it is necessary to stop the OPMN daemon. Once started, the OPMN daemon should remain up until it is necessary to restart the computer or some other unforeseen administrative event occurs.

To stop all OPMN-managed processes without stopping the OPMN daemon, consider using the `opmnctl stopproc` command without any arguments.

To restart the OPMN daemon without restarting any OPMN-managed processes, consider using the `opmnctl reload` command. The `opmnctl reload` command is the appropriate command to use when the only goal is to restart the opmn daemon with a new configuration.

Use the `opmnctl stopproc` command if you want to stop all OPMN managed processes.

Use the `opmnctl reload` if you want OPMN to reread its configuration.

Enter one of the following commands to obtain additional information:

```
prompt > opmnctl usage stopall
```

or

```
prompt > opmnctl usage shutdown
```

4.3.2.5 opmnctl shutdown

Syntax: `opmnctl shutdown`

Use the `opmnctl shutdown` command to shut down the OPMN server as well as the OPMN-managed processes for the local Oracle Application Server instance.

The `opmnctl shutdown` command quickly shutdowns the OPMN daemon and OPMN-managed processes for the local Oracle Application Server instance.

The `opmnctl shutdown` command is similar to the `opmnctl stopall` command but waits less time before initiating a forceful termination of OPMN-managed processes. After all of the OPMN-managed processes are stopped, the OPMN daemon will shutdown itself.

The `opmnctl shutdown` command should only be performed when it is necessary to stop the OPMN daemon. Once started, the OPMN daemon should remain up until it is necessary to restart the computer or some other unforeseen administrative event occurs.

To stop all OPMN-managed processes without stopping the OPMN daemon, consider using the `opmnctl stopproc` command without any arguments.

To restart the OPMN daemon without restarting any OPMN-managed processes, consider using the `opmnctl reload` command. The `opmnctl reload` command is the appropriate command to use when the objective is to restart the OPMN daemon with a new configuration.

On Microsoft Windows, you can also perform an `opmnctl shutdown` by stopping the `Oracle<OracleHomename>ProcessManager` service in the Windows services control panel.

Use the `opmnctl stopproc` command if you want to stop all OPMN managed processes.

Use the `opmnctl reload` if you want OPMN to reread its configuration.

Enter one of the following commands to obtain additional information:

```
prompt > opmnctl usage stopall
```

or

```
prompt > opmnctl usage shutdown
```

4.3.2.6 opmnctl reload

Syntax: `opmnctl [scope] reload`

Use this command to trigger the OPMN to reread its configuration files in the requested scope. This command restarts the OPMN server without restarting any Oracle Application Server processes managed by OPMN. The OPMN server for the Oracle Application Server instance must be up and running.

Note: On Microsoft Windows, you can highlight the `Oracle<OracleHomename>ProcessManager` in the services control panel and select **Restart**. The restart of the service is not equivalent to an `opmnctl reload`, however. This action is equivalent to an `opmnctl shutdown` followed by an `opmnctl startall`. It is a much slower operation than `opmnctl reload` because it restarts OPMN and all the processes managed by OPMN.

Enter the following command for additional detailed information:

```
prompt > opmnctl usage reload
```

See Also: Section 4.3.1, "Command Definitions"

4.3.3 Process Control Commands

The `opmnctl` process control commands enable you to start, stop, or restart single or multiple Oracle Application Server components. You can control an Oracle Application Server component at the `<ias-component>`, `<process-set>`, or `<process-type>` level.

This section describes the process control commands available with `opmnctl`. It includes the following process control commands:

- Section 4.3.3.1, "opmnctl startproc, opmnctl restartproc and opmnctl stopproc"
- Section 4.3.3.2, "Progressive Request Reports"
- Section 4.3.3.3, "Sequential Requests"
- Section 4.3.3.4, "opmnctl config"
- Section 4.3.3.5, "Starting a Specific J2EE Application"

Output is not generated for the successful execution of an `opmnctl` process control command. Refer to Appendix A, "OPMN Troubleshooting" if you receive any error messages during `opmnctl` command execution.

4.3.3.1 opmnctl startproc, opmnctl restartproc and opmnctl stopproc

Syntax: `opmnctl [<scope>] startproc [<attr>=<value>...]`

`opmnctl [<scope>] restartproc [<attr>=<value>...]`

`opmnctl [<scope>] stopproc [<attr>=<value>...]`

Use these commands to start, restart, or stop OPMN-managed processes in the requested scope. The OPMN server for the Oracle Application Server instance must be up and running.

The following attributes and values can be used with the `startproc, stopproc,` and `restartproc` commands:

- `ias-component, process-type,` and `process-set:` The values for these attributes should be the same as the `id` value specified in the `opmn.xml` file. If no attribute is supplied, the command is applied to all OPMN-managed processes other than those that are configured in the `opmn.xml` file with `id-matching="true"`. To execute commands on components configured with `id-matching="true"`, it is necessary to specify the `ias-component` argument.

- `mode`: The `mode` attribute value can be either `sync` or `async`; the default value is `sync`. The `sync` value for mode causes the `opmnctl` command to operate synchronously and wait for the command to be executed completely before a return prompt is displayed. The timeout element can only be specified when the value of mode is `sync`. The value is specified in number of seconds. After the specified timeout expires, the operation is aborted for `startproc` but not for `restartproc` or `stopproc`. The `opmnctl` command prompt returns, the OPMN server continues to perform the `opmnctl restartproc` or `stopproc` command request until the operation is finished.

  The `async` value for mode causes the return prompt to be displayed immediately, while the OPMN server continues to perform the `opmnctl` command request until the operation is finished.

- `uniqueid`: This value is assigned by OPMN after starting up. You can use this value when you execute the `restartproc` and `stopproc` commands. You can obtain this value by entering the following command and obtaining the unique number for the Oracle Application Server component in the `uid` column of the generated output:

  ```
  prompt > opmnctl status -l
  ```

Attribute names other than those listed may be specified for some types of Oracle Application Server processes managed by OPMN. Unique attribute name should be specific to each type of Oracle Application Server process.

Using the `opmnctl startproc, restartproc,` or `stopproc` commands with a specified scope and attributes enables control of specific processes in your enterprise. You can execute the `opmnctl startproc, restartproc,` or `stopproc` commands at the `<ias-component>`, `<process-type>` and the `<process-set>` level.

For example, the following command starts OracleAS Wireless at the `<process-set>` level:

```
prompt > opmnctl startproc ias-component=wireless process-type=alert_server
process-set=alert_instance_1
```

The following command restarts OC4J at the `<process-type>` level:

```
prompt > opmnctl restartproc ias-component=OC4J process-type=home
```

The following command stops Oracle HTTP Server at the `<ias-component>` level:

```
prompt > opmnctl stopproc ias-component=HTTP_Server
```

Enter one of the following commands to obtain additional information:

```
prompt > opmnctl usage startproc
```

or

```
prompt > opmnctl usage restartproc
```

or

```
prompt > opmnctl usage stopproc
```

See Also:

- Section 4.3.1, "Command Definitions"
- Chapter 5, "Using OPMN"

4.3.3.2 Progressive Request Reports

The `report=true` attribute when used with `startproc`, `restartproc`, or `stopproc` enables OPMN to report back on each part of a request as it completes. For example, if an `opmnctl startproc` request will attempt to start 4 processes, OPMN will report back to the user the result of each process start attempt as soon as it completes. This attribute works on scoped requests in the same way.

For example, the following shows the reports display when a request is issued:

1. `% opmnctl @instance:your_company:ias_stado17_1 startproc report=true opmnctl: starting opmn managed processes...`

2. `% opmnctl @instance:your_company:ias_stado17_1 startproc report=true opmnctl: starting opmn managed processes...`

   ias_stado17_1/HTTP_Server/HTTP_Server/HTTP_ Server/1,86061,17501: success

   ias_stado17_1/ppid/ppid/ppid/1,86063,17503: success

3. `% opmnctl @instance:your_company:ias_stado17_1 startproc report=true opmnctl: starting opmn managed processes...`

   `ias_stado17_1/HTTP_Server/HTTP_Server/HTTP_ Server/1,86061,17501: success`

   `ias_stado17_1/ppid/ppid/ppid/1,86063,17503: success`

   your_company/HTTP_Server/HTTP_Server/HTTP_ Server/1,2452,11833: success

4. `% opmnctl @instance:your_company:ias_stado17_1 startproc report=true opmnctl: starting opmn managed processes...`

   `ias_stado17_1/HTTP_Server/HTTP_Server/HTTP_ Server/1,86061,17501: success`

   `ias_stado17_1/ppid/ppid/ppid/1,86063,17503: success`

   `your_company/HTTP_Server/HTTP_Server/HTTP_ Server/1,2452,11833: success`

   ias_stado17_1/OC4J/home/default_group/1,86062,17502: success

   ias_stado17_1: 3 of 3 processes started.

5. `% opmnctl @instance:your_company:ias_stado17_1 startproc report=true opmnctl: starting opmn managed processes...`

   `ias_stado17_1/HTTP_Server/HTTP_Server/HTTP_ Server/1,86061,17501: success`

   `ias_stado17_1/ppid/ppid/ppid/1,86063,17503: success`

   `your_company/HTTP_Server/HTTP_Server/HTTP_ Server/1,2452,11833: success`

   `ias_stado17_1/OC4J/home/default_group/1,86062,17502: success`

   `ias_stado17_1: 3 of 3 processes started.`

   your_company/OC4J/home/default_group/1,2451,11851: failure

   failed to start a managed process after the maximum retry limit

```
Log:/private/oracle/OraHome_1/opmn/logs/OC4J~home~default_
group~1 your_company: 1 of 2 processes started.
```

4.3.3.3 Sequential Requests

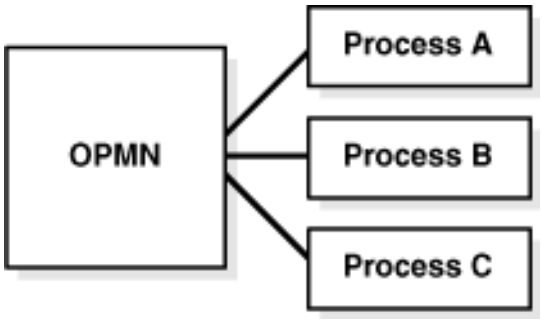
By default an OPMN request is run for all affected processes at the same time, unless a dependency dictates a specific ordering. If the attribute `sequential=true` is specified when used with the `startproc`, `restartproc`, or `stopproc` command, then OPMN will only run the request on a single process at a time, waiting for the request to complete on the first before running the request on the second. When the request has finished on one process, it works on the next.

Note that dependencies are still honoured, and take part in the request sequentially as well.

As shown in Figure 4–1, by default OPMN issues jobs for all processes in parallel such that they run at the same time (except when honoring dependencies). For example, with the following command:

```
> opmnctl startproc ias-component=OC4J
```

Figure 4–1 OC4J Processes in Parallel



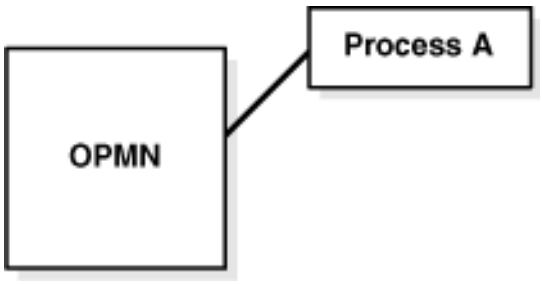
If the sequential attribute is set to true, OPMN will only perform the request upon one process at a time (shown in Figure 4–2).

For example the following command:

```
% opmnctl startproc ias-component=OC4J sequential=true
```

starts all of the OC4J processes sequentially.

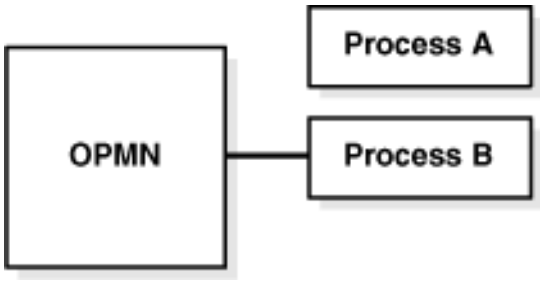
Figure 4–2 OC4J Process Sequential Request #1



OPMN is processing one OC4J process, before moving on the next shown in Figure 4–3.

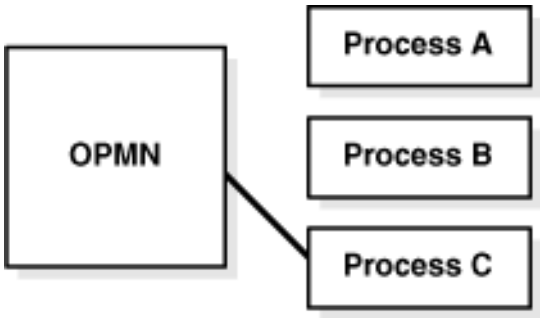
As shown in Figure 4–3, when the request completes for the first OC4J process, the request starts on the next OC4J process.

Figure 4–3 OC4J Process Sequential Request #2



As shown in Figure 4–4, all affected OC4J processes have completed the request.

Figure 4–4 OC4J Process Sequential Request #3



4.3.3.4 opmnctl config

Syntax: `opmnctl config <target> <operation> <options>`

The `opmnctl config` command enables you to execute configuration operations on either the `topology` or `port` element of the `opmn.xml` file using the command line. You do not need to access the `opmn.xml` file to perform a `topology` or `port` configuration.

`<target>` = `topology` or `port`

`<operation>` = `update` or `delete`

`<options>` = the use of this parameter is dependent on your selection for the `<target>` and `<operation>` arguments

The `topology` target is used to configure the `discover` element in the `opmn.xml` file. The `discover` element provides a list of discovery service addresses. The local OPMN server will use the discover element to find and connect with remote OPMN servers in the same cluster. You can configure multiple discover elements.

The arguments for the `topology` command are:

- update: for the update parameter, a discover attribute with a value is required. The format is discover=<value> where the value is the discovery service address.
- delete: for the delete parameter, a discover attribute is required but no value needs to be supplied.

The following examples show usage of the described arguments and parameters:

- Update the discover attribute in the opmn.xml file with the discovery service address of *234.5.6.7.8910:

```
> opmnctl config topology update discover="*234.5.6.7:8910"
```

- Delete the discover attribute in the opmn.xml file:

```
> opmnctl config topology delete discover
```

The port target is used to configure the port elements in the opmn.xml file. The arguments for the port command are:

- ias-component=<id>: this argument specifies the ias-component in which the port element resides.
- process-type=<id>: this argument specifies the process-type in which the port element resides.
- portid=<id>: this argument specifies the port ID that needs to updated or deleted.
- range=<port range>: optional argument for the update operation. This parameter is invalid for the delete operation.
- protocol=<port protocol>: optional argument for the update operation. This is an invalid argument for the delete operation.

The following examples show usage of the described arguments and parameters:

- Update the default Web site port element for the OC4J ias-component element, in the home process-type using HTTP protocol:

```
> opmnctl config port update ias-component=OC4J process-type=home
  portid=default-web-site protocol=http
```

- Delete the default Web site port element for the OC4J ias-component element, in the home process-type:

```
> opmnctl config port delete ias-component=OC4J process-type=home
portid=default-web-site
```

4.3.3.5 Starting a Specific J2EE Application

You can start or stop your J2EE based application using the application attribute. You can use this attribute with the startproc, restartproc, or stopproc, opmnctl commands.

For example:

- prompt > opmnctl startproc application=mailserver

  Start the application mailserver on every process in the local Oracle Application Server instance that has the application mailserver.

- `prompt > opmnctl @cluster startproc application=password-manager`

  Start the `password-manager` application on every process in the entire Oracle Application Server cluster that has the `password-manager` application

- `prompt > opmnctl @cluster startproc process-type=home application=web-module`

  Start the application `web-module` on every process in the entire Oracle Application Server cluster that belongs to the `process-type` named `home` and has the application `web-module`.

- `prompt > opmnctl @instance:inst1:inst2:inst3 startproc process-type=home application=web-module`

  Start the application `web-module` on every process in the three named Oracle Application Server instances that belong to the `process-type` named `home` and has the application `web-module`.

4.3.4 Status Commands

The `opmnctl status` commands enable you to determine the status of OPMN-managed processes.

This section describes the command options available with the `opmnctl` command. It includes the following sections:

- Section 4.3.4.1, "opmnctl status"
- Section 4.3.4.2, "opmnctl dmsdump"
- Section 4.3.4.3, "opmnctl ping"
- Section 4.3.4.4, "opmnctl set"
- Section 4.3.4.5, "opmnctl query"

See Also:

- Section 4.3.1.1, "Scope"
- Section 4.3.1.2, "Attributes"
- Section 4.3.4.1.1, "Options for the Status Command of opmnctl"
- Section 4.3.4.1.2, "opmnctl status -port"
- Section 4.3.4.1.3, "opmnctl status -app"

4.3.4.1 opmnctl status

Syntax: `opmnctl [<scope>] status [<options>]`

The `status` command enables you to obtain information on the Oracle Application Server processes managed by OPMN.

The output is a text table. Each row in the table represents one Oracle Application Server process.

You can customize the status command in the following ways:

- Supply a scope to obtain status of processes running on other Oracle Application Server instances
- Change the information displayed about each Oracle Application Server process

- Remove the table headers from the output
- Change the field separator
- Change the record separator
- Change the width of individual columns
- Change the justification of the data in an individual column

Enter the `opmnctl usage status` command to obtain full details on how to use the status command.

Example 4–3 shows the output after entering the `opmnctl status` command for the `AppSrv1` instance on host `comp1` for the domain `yourcompany.com`:

Example 4–3 opmctl Status Output

```
prompt > opmnctl status

Processes in Instance: AppSrv1.comp1.yourcompany.com
-------------------+---------------------+--------+---------
ias-component      | process-type        |    pid | status
-------------------+---------------------+--------+---------
OC4J               | OC4J_Demos          |    N/A | Down
OC4J               | home                |  29268 | Init
HTTP_Server        | HTTP_Server         |  29099 | Alive
```

You can use the `opmnctl status` command with `<scope>` to obtain additional detailed information. For example, the following command gives you the status of every process of every component of every OracleAS Instance in an entire cluster:

```
prompt > opmnctl @cluster status
```

See Also: Section 4.3.1.1, "Scope"

4.3.4.1.1 Options for the Status Command of opmnctl

The following are the options you can specify for the `<options>` parameter:

- **`-l`:** Use this option to obtain the `uniqueid (uid)` value and other specific process parameter information.

  ```
  prompt > opmnctl status -l
  ```

  For example, the following command outputs the information shown in Example 4–4:

Example 4–4 opmnctl status -l output

```
Processes in Instance: j2eeuser.yourcompany.com
-------------+------------+------+-------+---------+-------+---------+--------
ias-component|process-type|pid   |status |uid      |memused|uptime   | ports
-------------+------------+------+-------+---------+-------+---------+--------
-OC4J        | home       | 5611 | Alive |632225812|105008 |17:55:58 |jms:3701,
rmi: 3201,ajp:3000
```

The `uid` information enables you to stop or restart an individual Oracle Application Server process.

For example, the following command stops the `home process-type`:

```
prompt > opmnctl stopproc uniqueid=632225812
```

- **`-fsep <string>`**: Use this option to assign a field separator value for your `opmnctl status` output. The default value is `|`.
- **`-rsep <string>`**: Use this option to assign a record separator value for your `opmnctl status` output. The default value is **`\n`**.
- **`-noheaders`**: Use this option if you do not want a header displayed after you run the `opmnctl status` command.
- **`-fmt <fmtlist>`**: This is a single string containing one or more statistic identifiers connected together where each identifier has the following format: `<statname>[<width>{<justification>]`. The default value is: `%cmp18%prt18%pid5R%sta8`.

  Table 4–3 lists the format string syntax for the `<fmtlist>` option:

Table 4–3 Format String Syntax

Format String Syntax	Description
`<statname>`	This must be one of the following: ■ `clu`: Oracle Application Server cluster name ■ `ins`: Oracle Application Server instance name ■ `cmp`: Oracle Application Server component ID ■ `prt`: `process-type` ID ■ `prs`: `process-set` ID ■ `idx`: index of process in `process-set` ■ `pid`: operating system process ID ■ `uid`: OPMN `uniqueid` ■ `typ`: name for this kind of process ■ `sta`: process status ■ `stm`: start time (ms) ■ `utm`: up time (ms) ■ `cpu`: cpu time (ms) ■ `mem`: memory used (in KB) ■ `por`: port list
`<width>`	Specifies the size for the field. Output shorter than this value receives padding according to the specified <justification>. Output longer than this value is truncated, and terminated with '~'. Default: width of each datum.
`<justification>`	Specifies the justification for the field. This enables you to justify output when it is less than the width. It is `L`, `R`, or `C` (left, right, or center justification). Default: `L`

For example, the following command displays the output shown in Example 4–5:

```
prompt> opmnctl status -noheaders -fsep @ -fmt %cmp%prt%pid%sta
```

Example 4–5 opmnctl status -noheaders output

```
OC4J@home@N/A@Down
```

```
HTTP_Server@HTTP_Server@13926@Alive
```

See Also: Section 4.3.1, "Command Definitions"

Enter the following command for additional detailed information:

```
prompt > opmnctl usage status
```

4.3.4.1.2 opmnctl status -port

The `opmnctl status -port` command enables you to display the request connect string used to connect to the OPMN daemon. For example, the command:

```
prompt > opmnctl status -port
```

displays:

```
123.your_company.com:6200
```

This information can be used by a remote tool that is seeking to access OPMN. For example, in a J2EE Server and Process Management installation type, the request port information can be used in conjunction with the `admin_client.jar` utility to perform a deployment to an OC4J instance within the scope of the identified OPMN instance.

This `opmnctl status -port` command is a convenient shortcut that replaces the need to look inside of the `opmn.xml` file to determine the request access port.

4.3.4.1.3 opmnctl status -app

The `opmnctl status -app` command displays information for applications (`module-ids`) that are managed by OPMN.

For example, after entering the following command:

```
> opmnctl status -app
```

you will see the following output:

```
application type: OC4J
-----+-----------+---------+---------+----------+-------
pid  | name      | state   | rtid    | routable | parent
-----+-----------+---------+---------+----------+-------
2816 | system    | started | g_rt_id | true     |
2816 | default   | started | g_rt_id | true     |system
2816 | bc4j      | stopped | g_rt_id | false    |
2816 | ascontrol | started | g_rt_id | true     |
```

Only applications that are reported by live processes show up in the `opmnctl status -app` output.

The information that is reported using the `opmnctl status -app` command varies for each `module-id`. A separate table is created for each `module-id`. Data that is not available with a process is shown as empty or as `N/A` in the process table.

You can use the command options specified in Section 4.3.4.1, "opmnctl status" with the `opmnctl status -app` command.

4.3.4.2 opmnctl dmsdump

Syntax: `opmnctl [<scope>] dmsdump [<attr>=<value>[&<attr>=<val>...]]`

The `opmnctl dmsdump` command enables you to print the Oracle Dynamic Monitoring Service (DMS) statistics for OPMN. You can obtain a printout of process control operations for specific Oracle Application Server components. If no attributes are specified, performance data for all OPMN-managed processes for your Oracle Application Server components are printed out.

DMS enables you to monitor a specific performance metric, a set of performance metrics, or all performance metrics. Options allow you to specify a reporting interval to report the requested metrics.

Multiple `<attr>=<value>` pairs must be separated by an `&`. For example, the following `opmnctl` command:

```
prompt > opmnctl dmsdump "table=opmn_ons&format=xml"
```

will output the set of statistics that are gathered for ONS. The output includes the ports that ONS listens on and the number of notifications that ONS has processed. The output is in `.xml` format rather than text. If you want to review the output in text format do not include `&format=xml` on the command line.

For more information about DMS performance metric attributes and values refer to the *Oracle Application Server Performance Guide*.

4.3.4.3 opmnctl ping

Syntax: `opmnctl ping` [`<max_retry>`]

The `opmnctl ping` command enables you to contact the local OPMN server to verify operation. `<max_retry>` specifies the maximum number of retry times. If `<max_retry>` is specified, the local OPMN is pinged every one second, until the command execution succeeds or `<max_retry>` is reached.

For example, the following command,

```
prompt > opmnctl ping 10
```

designates pinging of OPMN 10 times until the ping command succeeds

4.3.4.4 opmnctl set

Syntax: `opmnctl [<scope>] set [<attr>=<value> ...]`

The `opmnctl set` command sets the logging configuration for OPMN.

The `scope` defines where the `opmnctl set` request will be routed. If none or only `@instance` is specified, the request will be routed to the local Oracle Application Server instance only. If `@instance` is followed by specified instance names, the request will be routed to the specified Oracle Application Server instances.

If `@cluster` is specified, the request will be routed to all instances in the cluster.

An attribute name must be specified along with an attribute value. The following attribute names are required by OPMN for this command:

- `target`: the value for `target` can be either `log` or `debug`, which refer to the `opmn.log` or the `opmn.dbg`, respectively. Refer to Section A.2.1, "OPMN log Files" for more information.

> **Note:** Enable usage of the `opmn.dbg` file only after conferring with Oracle Support. The `opmn.dbg` file is used by Oracle Support to debug and diagnose OPMN issues. Messages that are contained in the `opmn.dbg` file are typically not readily comprehensible to the user.

- `comp`: specifies the OPMN internal components and subcomponents

4.3.4.4.1 The `comp` Attribute

The value for the `comp` attribute can be either `ons` or `pm`. Additionally, the attribute value can be a specific set of sub-components for either the ons or `pm` attributes.

The following values for `comp` specify the OPMN internal components and subcomponents:

- `internal`:specifies the common internal information for OPMN
- `ons`:specifies the ONS component information for OPMN
- `pm`: specifies the PM component information for OPMN

Both the ons and `pm` components consist of subcomponents which may be specified using the `component[subcomponents]` syntax where component can be either ons or `pm`. If both ons and `pm` are specified together they must be separated by a semicolon in the `opmn.xml` file. If subcomponents are listed, the listed items must be separated by a comma.

Table 4–4 ONS Component Codes

ONS element	Definition
all	all subcomponents
local	local information
listener	listener information
discover	discover (server or multicast) information
servers	remote servers currently up and connected to the cluster
topology	current cluster wide server connection topology
server	remote server connection information
client	client connection information
connect	generic connection information
subscribe	client subscription information
message	notification receiving and processing information
deliver	notification delivery information
special	special notification processing
internal	internal resource information
secure	SSL operation information
workers	worker threads

Table 4–5 PM Component Codes

PM element	Definition
all	all subcomponents
requests	HTTP (user) requests
remote	remote HTTP requests
scheduler	scheduler thread and resource information
monitor	monitor thread information
workers	worker threads
process	managed processes
depend	dependency processing
rmd	RMD directives
fos	service failover information
internal	internal resources
schedjobs	periodic scheduled jobs
procjobs	for each process scheduled jobs
fos	service failover processing
dms	DMS processing
modules	`process module` information. Only modules which call the `modLog()` (or `modDebug` for the debug log) function will yield output. A specific module or list of modules may be specified with modules(`module-ids`). `module-ids` can be specified with a colon seperated list of `module-ids` to be displayed: modules (`module1-id:module2-id`). `module-ids` that do not match a configured and enabled module are not processed.

Each subcomponent for ons and `pm` may be prefaced with the negation character, !, which deselects the subcomponent. By using `all` with negated sub-components, specific subcomponents can be easily eliminated from the display.

Components and subcomponents are set or negated in the order in which they are encountered. Therefore:

```
ons[all,!topology]
```

will yield all ons subcomponents excluding topology, while:

```
ons[!topology,all]
```

will yield all ons subcomponents including topology.

4.3.4.5 opmnctl query

Syntax: `opmnctl [<scope>] query [<attr>=<value> ...]`

The `opmnctl query` command enables you to query the logging configuration for OPMN.

The attribute name of `target` must be specified along with an attribute value. The value for `target` can be either `log` or `debug`, which refer to the `opmn.log` file or the `opmn.dbg` file, respectively. Refer to Section A.2.1, "OPMN log Files" for more information.

4.3.5 Help Commands

The `opmnctl help` commands enable you to obtain additional information regarding OPMN.

This section describes the help command options available with the `opmnctl` command. It includes the following sections:

- Section 4.3.5.1, "opmnctl help"
- Section 4.3.5.2, "opmnctl usage"
- Section 4.3.5.3, "opmnctl validate"

4.3.5.1 opmnctl help

Syntax: `opmnctl help`

Use this command to print a short syntax description of `opmnctl` commands.

Example 4–6 shows the output from the `opmnctl help` command.

Example 4–6 opmnctl help Output

```
prompt > opmnctl help

usage: /ORACLE_HOME/bin/opmnctl [verbose] [<scope>] <command> [<options>]

verbose: print detailed execution message if available

Permitted <scope>/<command>/<options> combinations are:

  scope      command      options
_________   __________   _________
            start                        - Start opmn
            startall                     - Start opmn & all managed processes
            stopall                      - Stop opmn & all managed processes
            shutdown                     - Shutdown opmn & all managed processes
[<scope>]   startproc   [<attr>=<val>..]  - Start opmn managed processes
[<scope>]   restartproc [<attr>=<val>..]  - Restart opmn managed processes
[<scope>]   stopproc    [<attr>=<val>..]  - Stop opmn managed processes
[<scope>]   reload                        - Trigger opmn to reread opmn.xml
[<scope>]   status      [<options>]       - Get managed process status
[<scope>]   dmsdump     [<attr>=<val>&..] - Get DMS stats
[<scope>}   set         [<attr>=<val> ..] - Set opmn log parameters
[<scope>]   query       [<attr>=<val> ..] - Query opmn log parameters
            ping        [<max_retry>]     - Ping local opmn
            validate    [<filename>]      - Validate the given xml file
            config      [<options>]       - Modify the opmn xml file
            help                          - Print brief usage description
            usage       [<command>]       - Print detailed usage description
```

4.3.5.2 opmnctl usage

Syntax: `opmnctl usage [<command>]`

The `usage` command displays help for all `opmnctl` commands, or only for the specified command.

The command can be one or more of the following:

- `start`
- `startall`

- startproc
- stopall
- stopproc
- restartproc
- reload
- shutdown
- ping
- status
- dmsdump
- config
- help

For example, enter the following command to receive the output shown in Example 4–7:

```
prompt > opmnctl usage stopall
```

Example 4–7 opmnctl usage stopall output

```
opmnctl stopall
Stop opmn daemon and opmn managed processes for local ias instance.

This request first tries to stop all opmn managed processes gracefully.
Processes which will not stop gracefully will be forcefully shutdown.  After
stopping all managed processes, the opmn daemon will shutdown itself.

This request should only be performed when it is necessary to stop the opmn
daemon.  Once started, the opmn daemon should remain up until it is necessary to
restart the computer or some other rare administrative event occurs.

To stop all opmn managed processes without stopping the opmn daemon, consider
using the stopproc command without any arguments.

To restart the opmn daemon without restarting any managed processes, consider
using the the reload command.  The reload command is the appropriate command to
use when the only goal is to restart the opmn daemon with a new configuration.

This request operates synchronously and will wait for the operation to complete
before returning.
```

4.3.5.3 opmnctl validate

Syntax: `opmnctl validate` [<filename>]

The `opmnctl validate` command validates the XML syntax of the `opmn.xml` file. The default *ORACLE_HOME*`/opmn/conf/opmn.xml` is validated if the filename parameter is not specified. The <filename> can be specified by either the relative or absolute path.

Only one file can be validated at a time.

5

Using OPMN

This chapter provides command-line examples on how to use OPMN for Oracle Application Server. It features the following topics:

5.1 Starting OPMN

OPMN does not depend on any other Oracle Application Server component being up and running before it can be started and used. The OPMN server should be started as soon as possible after turning on the host.

Use the following command to start OPMN without starting other Oracle Application Server components:

```
prompt > opmnctl start
```

5.2 Starting and Stopping OPMN-Managed Processes for a Local Oracle Application Server Instance

Use the following command to **start** OPMN-managed processes for a local Oracle Application Server instance:

```
prompt > opmnctl startproc
```

Use the following command to **stop** OPMN-managed processes for a local Oracle Application Server instance:

```
prompt > opmnctl stopproc
```

Note: Without arguments the `opmnctl startproc` and `opmnctl stopproc` commands start and stop all OPMN-managed processes.

5.3 Starting and Stopping all OPMN Managed Processes for a Remote Oracle Application Server Instance

Use the following command to **start** OPMN managed processes for a remote Oracle Application Server instance:

```
prompt > opmnctl @instance:oracleas2.foo.com startproc
```

Use the following command to **stop** OPMN managed processes for a remote Oracle Application Server instance:

```
prompt > opmnctl @instance:oracleas2.foo.com stopproc
```

5.4 Starting and Stopping an Oracle Application Server Component in a Local Oracle Application Server Instance

Use the following command to **start** Oracle Internet Directory in a local Oracle Application Server instance:

```
prompt > opmnctl startproc ias-component=OID
```

Use the following command to **stop** Oracle Internet Directory in a local Oracle Application Server instance:

```
prompt > opmnctl stopproc ias-component=OID
```

5.5 Starting and Stopping an Oracle Application Server Process Type in a Local Oracle Application Server Instance

Use the following command to **start** the `performance_server process-type` in a local Oracle Application Server instance:

```
prompt > opmnctl startproc ias-component=OC4J process-type=performance_server
```

Use the following command to **stop** the `performance_server process-type` in a local Oracle Application Server instance:

```
prompt > opmnctl stopproc ias-component=OC4J process-type=performance_server
```

5.6 Starting and Stopping a Multi-Oracle Application Server Instance Environment

Use the following command to **start** a multi-Oracle Application Server instance environment from local instance `oracleas1`:

```
prompt > opmnctl @instance:oracleas1.foo.com:oracleas2.bar.com startproc
```

This command starts all processes of all components on both instances specified with the <scope> argument. Notice that the local instance oracleas1 is specified in the command.

Use the following command to **stop** a multi-Oracle Application Server instance environment from local instance oracleas1:

```
prompt > opmnctl @instance:oracleas1.foo.com:oracleas2.bar.com stopproc
```

This command stops all processes of all components on all two instances specified with the <scope> argument. Notice that the local instance oracleas1 is specified in the command.

Note: You must also indicate your local Oracle Application Server instance when using the @instance <scope> with other Oracle Application Server instances. Commands with a <scope> argument only operates on the instances described by the <scope> argument. The command will only be applied to the local instance if it is described in the <scope> argument.

5.7 Starting a Component on an Oracle Application Server Cluster

Use the following command to **start** the same Oracle Application Server component on multiple Oracle Application Server instances:

```
prompt > opmnctl @cluster startproc ias-component=HTTP_Server
```

Use the following command to **stop** the same Oracle Application Server component on multiple Oracle Application Server instances:

```
prompt > opmnctl @cluster stopproc ias-component=HTTP_Server
```

6

opmn.xml Common Configuration

This chapter provides common configuration examples, and descriptions of elements and attributes for the OPMN `opmn.xml` file.

It contains the following topics:

- Section 6.1, "Example of opmn.xml Elements and Attributes"
- Section 6.2, "opmn.xml Element and Attribute Descriptions"

6.1 Example of opmn.xml Elements and Attributes

Example 6–1 shows all possible elements and attributes that may appear in an `opmn.xml` file that are not specific to any Oracle Application Server component.

> **Note:** OPMN will convert slashes in the path value string to be those of the directory path separator character for the system on which OPMN is running (for Linux each \ character is converted to **/**; for Microsoft Windows each **/** is converted to \).
>
> OPMN uses the **^** character as an escape character to disable slash conversion. **^/** on a Microsoft Windows system will yield a **/** in the string. Specify two **^** characters if you need to specify the **^** character in the resultant string. For example, **^^** yields **^**.

Example 6–1 Common Configuration Elements and Attributes

```
<opmn>
<log path="path" comp="comp-codes" rotation-size="kBytes" rotation-hour="HOD"/>
<debug path="path" comp="comp-codes" rotation-size="kBytes" rotation-hour="HOD"/>
   <notification-server interface="type">
      <ipaddr remote="ip" request="ip"/>
      <port local="port" remote="port" request="port"/>
      <ssl enabled="boolean" wallet-file="path" wallet-password="password" openssl-certfile="path"
       openssl-keyfile="path" openssl-password="password" openssl-lib="path"/>
      <tune io-timeout="timeout" io-idle="interval" timeout="timeout"/>
      <topology >
         <nodes list="nodes"/>
         <discover list="nodes"/>
         <gateway list="nodes"/>
      </topology>
   </notification-server>
   <process-manager insecure-remote-requests="boolean">
      <process-modules >
         <module path="path" tag="tag-id" status="state" cron="interval">
```

```
            <module-data >
               <category id="id">
                  <data id="id" value="value" process-conversion="boolean"/>
               </category>
            </module-data>
         <module-id id="module-id"/>
   </module>
</process-modules>
<ias-instance id="ias-instance-id" name="ias-instance-name" ORACLE_HOME="path">
   <environment >
      <variable id="id" value="value" append="boolean" process-conversion="boolean"/>
   </environment>
<!-- module-data -->
   <rmd-definitions >
      <rmd name="name" description="description" interval="interval">
         <conditional>
            <![CDATA[condition]]>
         </conditional>
      <action value="action"/>
      <exception value="exception">
   </rmd>
</rmd-definitions>
<ias-component id="component-id" id-matching="boolean" status="state">
   <!-- environment -->
   <!-- module-data -->
   <dependencies >
      <database db-connect-info="connect" infrastructure-key="key" timeout="depend-timeout"
       cache-timeout="cache-timeout"/>
      <OID address="address" infrastructure="boolean" timeout="depend-timeout"
       cache-timeout="cache-timeout">
         <ssl enabled="boolean" wallet-file="path" wallet-password="password">
      </OID>
      <OSSO host="hostname" port="port" URI="uri" timeout="depend-timeout"
       cache-timeout="cache-timeout">
         <ssl enabled="boolean" wallet-file="path" wallet-password="password">
      </OSSO>
      <managed-process ias-instance="ias-instance-id" ias-component="ias-component-id"
       process-type="process-type-id" process-set="process-set-id" autostart="boolean"
       autostop="boolean" timeout="depend-timeout"cache-timeout="cache-timeout"/>
   </dependencies>
<!-- rmd-definitions -->
   <process-type id="process-type-id" module-id="module-id" status="state" working-dir="path"
    service-failover="num" service-weight="value">
      <!-- environment -->
      <!-- module-data -->
      <!-- dependencies -->
      <event-scripts >
         <pre-start path="path">
         <pre-stop path="path">
         <post-crash path="path">
      </event-scripts>
         <!-- rmd-definitions -->
      <start timeout="timeout" retry="num"/>
      <stop timeout="timeout"/>
      <restart timeout="timeout" retry="num"/>
      <ping timeout="timeout" retry="num" interval="interval"/>
      <port id="id" range="range"/>
      <process-set id="process-set-id" restart-on-death="boolean" numprocs="num" minprocs="min"
       maxprocs="max" status="state" working-dir="path" parallel-requests="boolean">
         <!-- environment -->
```

```
        <!-- module-data -->
        <!-- dependencies -->
        <!-- event-scripts -->
        <!-- rmd-definitions -->
        <!-- start: -->
        <!-- stop -->
        <!-- restart -->
        <!-- ping -->
        <!-- port -->
      </process-set>
      </process-type>
      </ias-component>
      </ias-instance>
      <!-- rmd-definitions (global) -->
   </process-manager>
</opmn>
```

6.2 opmn.xml Element and Attribute Descriptions

This section describes the elements and attributes in the `opmn.xml` file that are not specific to any Oracle Application Server component. This section also provides attribute descriptions of the elements.

<opmn>

Required: true
Default: none
Parents: none
Attributes: none

`<opmn>` is the top-level element in the `opmn.xml` file.

<log>

Required: false
Default: *see attributes*
Parents: `<opmn>`
Attributes: `path`, `comp`, `rotation-size`, `rotation-hour`

The configuration definitions for the OPMN log mechanism.

path="path"
Required: true
Default: *`$ORACLE_HOME`*`/opmn/logs/opmn.log`
Valid Values: The path name for the OPMN log file.

All directories specified in the path attribute must exist. OPMN must have read and write permissions for the directory containing the log file. The *`$ORACLE_HOME`* directory may be used.

comp="comp-codes"

Required: true
Default: `internal`, `ons`, `pm`
Valid Values: A list of `comp-codes`. A semi-colon must be used to separate the items on the `comp-codes` list.

The `comp` attribute specifies the component codes for logged events. These codes can be viewed and changed dynamically at OPMN run-time using the following commands:

```
> opmnctl query target=log
> opmnctl set target=log comp=<comp-codes>
```

The values revert back to the configured values in the `opmn.xml` file when OPMN is reloaded.

The following comp-codes are displayed in the log and debug elements of the `opmn.xml` file:

- `internal`: a log for the common internal information for OPMN
- `ons`: a log for the ONS component information for OPMN
- `pm`: a log for the PM component information for OPMN

Both the `ons` and `pm` components consist of subcomponents which may be specified using the `component[subcomponents]` syntax. The component can be either `ons` or `pm` (separated by a semicolon if both are specified). The list of valid subcomponents for the given component are each separated by a comma. For example, `comp="ons[local,listener];pm"`.

The following Table 6–1 lists the ONS component codes.

Table 6–1 ONS Component Codes

ONS Attribute	Definition
all	all subcomponents
local	local information
listener	listener information
discover	discover (server or multicast) information
servers	remote servers currently up and connected to the cluster
topology	current cluster wide server connection topology
server	remote server connection information
client	client connection information
connect	generic connection information
subscribe	client subscription information
message	notification receiving and processing information
deliver	notification delivery information
special	special notification processing
internal	internal resource information
secure	SSL operation information
workers	worker threads

The following Table 6–2 lists the PM component codes.

Table 6–2 PM Component Codes

PM Attribute	Definition
all	all subcomponents
requests	HTTP (user) requests
remote	remote HTTP requests
scheduler	scheduler thread and resource information
monitor	monitor thread information
workers	worker threads
process	managed processes
depend	dependency processing
rmd	RMD directives
fos	service failover information
internal	internal resources
schedjobs	periodic scheduled jobs
procjobs	for each process scheduled jobs
fos	service failover processing
dms	DMS processing
modules	`process module` information. Only modules which call the `modLog()` (or `modDebug` for the debug log) function will yield output. A specific module or list of modules may be specified with modules(`module-ids`). `module-ids` can be specified with a colon seperated list of `module-ids` to be displayed: modules (`module1-id:module2-id`). `module-ids` that do not match a configured and enabled module are not processed.

Each subcomponent (for `ons` or `pm`) may be prefaced with the negation character **!** which deselects the subcomponent. By using the term "**all**" with negated sub-components, specific subcomponents are eliminated from the display.

Components and subcomponents are set or negated in the order in which they are encountered. The `ons[all,!topology]` will yield all `ons` subcomponents excluding `topology`, while `ons[!topology,all]` will yield all `ons` subcomponents including `topology`.

rotation-size="kBytes"

Required: false
Default: none
Valid Values: An integer.

The `rotation-size` is the maximum size in kilobytes of the log file. When the log file reaches the configured size, the OPMN logging mechanism will close the log, rename it with a time stamp suffix, and then create a new log file. This attribute may be used with `rotation-hour`.

rotation-hour="HOD"

Required: false
Default: none
Valid Values: An integer value between 0 to 23.

At the given hour of the day, the OPMN logging mechanism will close the log, rename it with a time stamp suffix, and then create a new log file. This attribute may be used with `rotation-size`.

<debug>

Required: false
Default: *see attributes*
Parents: <opmn>
Attributes: `path`, `comp`, `rotation-size`, `rotation-hour`

The `debug` element contains the configuration definitions for the OPMN debug log mechanism.

> **Note:** Enable usage of the `opmn.dbg` file only after conferring with Oracle Support. The `opmn.dbg` file is used by Oracle Support to debug and diagnose OPMN issues. Messages that are contained in the `opmn.dbg` file are typically not readily comprehensible to the user.

path="path"

Required: true
Default: *$ORACLE_HOME*`/opmn/logs/opmn.dbg`
Valid Values: The path name for the OPMN debug log file.

All directories specified in the path name must already exist, and OPMN must have read and write permissions for the directory in which the debug log file resides. The *$ORACLE_HOME* directory may be used.

comp="comp-codes"

Required: true
Default: `internal`, `ons`, `pm`
Valid Values: A list of `comp-codes`. A semi-colon must be used to separate the items on the `comp-codes` list.

The `comp` attribute specifies the component codes for logged events. These codes can be viewed and changed dynamically at OPMN run-time using the following commands:

```
> opmnctl query target=log
> opmnctl set target=log comp=<comp-codes>
```

The values revert back to the configured values in the `opmn.xml` file when OPMN is reloaded.

The following comp-codes are displayed in the log and debug elements of the `opmn.xml` file:

- `internal`: a log for the common internal information for OPMN
- `ons`: a log for the ONS component information for OPMN
- `pm`: a log for the PM component information for OPMN

Both the `ons` and `pm` components consist of subcomponents which may be specified using the `component[subcomponents]` syntax. The component can be either `ons` or `pm` (separated by a semicolon if both are specified). The list of valid subcomponents for

the given component are each separated by a comma. For example, `comp="ons[local,listener];pm"`.

The following Table 6–3 lists the ONS component codes.

Table 6–3 ONS Component Codes

ONS Attribute	Definition
all	all subcomponents
local	local information
listener	listener information
discover	discover (server or multicast) information
servers	remote servers currently up and connected to the cluster
topology	current cluster wide server connection topology
server	remote server connection information
client	client connection information
connect	generic connection information
subscribe	client subscription information
message	notification receiving and processing information
deliver	notification delivery information
special	special notification processing
internal	internal resource information
secure	SSL operation information
workers	worker threads

The following Table 6–4 lists the PM component codes.

Table 6–4 PM Component Codes

PM Attribute	Definition
all	all subcomponents
requests	HTTP (user) requests
remote	remote HTTP requests
scheduler	scheduler thread and resource information
monitor	monitor thread information
workers	worker threads
process	managed processes
depend	dependency processing
rmd	RMD directives
fos	service failover information
internal	internal resources
schedjobs	periodic scheduled jobs
procjobs	for each process scheduled jobs
fos	service failover processing

Table 6–4 (Cont.) PM Component Codes

PM Attribute	Definition
dms	DMS processing
modules	`process module` information. Only modules which call the `modLog()` (or `modDebug` for the debug log) function will yield output. A specific module or list of modules may be specified with modules(`module-ids`). `module-ids` can be specified with a colon separated list of `module-ids` to be displayed: modules (`module1-id:module2-id`). `module-ids` that do not match a configured and enabled module are not processed.

Each subcomponent (for `ons` or `pm`) may be prefaced with the negation character **!** which deselects the subcomponent. By using the term "**all**" with negated sub-components, specific subcomponents are eliminated from the display.

Components and subcomponents are set or negated in the order in which they are encountered. The `ons[all,!topology]` will yield all `ons` subcomponents excluding `topology`, while `ons[!topology,all]` will yield all `ons` subcomponents including `topology`.

rotation-size="kBytes"

Required: false
Default: none
Valid Values: An integer.

The `rotation-size` is the maximum size in kilobytes of the log file. When the log file reaches the configured size, the OPMN logging mechanism will close the log, rename the log with a time stamp suffix, and then create a new log file.

The rotation-size attribute may be used with the `rotation-hour` attribute.

rotation-hour="HOD"

Required: false
Default: none
Valid Values: An integer value between 0 to 23.

The `rotation-hour` attribute for use with the log file at the specified time of day, the OPMN logging mechanism will close the log file, rename it with a time stamp suffix, and then create a new log file. This attribute may be used with the `rotation-size` attribute.

<notification-server>

Required: true
Default: none
Parents: <opmn>
Attributes: interface

The `notification-server` element configures or, contains the elements to configure the ONS portion of OPMN.

interface="type"

Required: false
Default: any
Valid Values: any, IPv6, or IPv4

By default, OPMN will support both the IPv6 and IPv4 network interfaces (see Section 3.10, "IPv6 Support"). OPMN binds to both interfaces for its listener ports and attempts connections using either interface. OPMN always attempts to use the IPv6 interface first, if such an address is available. This behavior is the same as the default behavior for the Sun Java Virtual Machine (JVM).

Both IPv4 and IPv6, require that you use the same network for connection to other OPMN servers. For example, IPv6 forces OPMN to only use the IPv6 network interface even if the IPv4 interface is available. This means that OPMN is only be able to connect to (and be connected from) other OPMN servers that can use the same network interface. If you configure IPv6 or IPv4 on a system that does not provide the same network interface support, OPMN will not start.

<ipaddr>

Required: true
Default: none
Parents: <notification-server>
Attributes: `remote, request`

The `iapddr` element specifies host information for ONS listener threads and host port bindings.

remote="ip; ip"

Required: false
Default: none
Valid Values: an IP address (in the ###.###.###.### format) or host name to which ONS will bind its remote port. A list of IP values may be configured to force ONS to bind to a specific set of IP addresses. Each IP value must be separated by a semi-colon on the list.

The `remote` attribute is the IP address or host name to which ONS will bind its remote port. The remote port is used for ONS to ONS communication. Notifications pass from ONS to ONS through the remote port, and OPMN uses ONS to route remote requests to other OPMN servers through the remote port.

request="ip; ip"

Required: false
Default: IP address for default system host name
Valid Values: an IP address (in ###.###.###.### format) or host name to which ONS will bind its request port. A list of IP values may be configured to force ONS to bind to a set of specific IP addresses. The values on the list must be separated by a semi-colon.

The `request` attribute is for the IP address or host name to which ONS will bind its remote port. This port can only be used for obtaining status information.

<port>

Required: true
Default: none
Parents: <notification-server>
Attributes: `local, remote, request`

The `port` element contains configuration information for ONS listener threads host and port bindings.

local="port"

Required: true
Default: none
Valid Values: A valid port number.

The `local` attribute is for the ONS local port value.

remote="port"

Required: false
Default: none
Valid Values: A port number.

The `remote` attribute is for the ONS remote port value.

request="port"

Required: false
Default: none
Valid Values: A port number.

The `request` attribute is for the ONS request `port` value.

<ssl>

Required: false
Default: none
Parents: <notification-server>
Attributes: `enabled, wallet-file, wallet-password`, `openssl-certfile`, `openssl-keyfile`, `openssl-password`, and `openssl-lib`

The `ssl` element is used for ONS to ONS security and authentication configuration. You may configure either the Oracle SSL layer (`wallet-file` and `wallet-password`) or the Open SSL layer (`openssl-certfile`, `openssl-keyfile`, `openssl-password`, and `openssl-lib`). You cannot use both. If you configure both Oracle and Open SSL layers within the same `opmn.xml` file, the server will not start.

The Oracle SSL layer uses the libraries shipped with either the Oracle Application Server or Oracle Database products. The Oracle SSL layer uses the standard Oracle wallet.

The Open SSL layer provides better performance, documentation, and support for encryption hardware. It also has uses less hardware space within OPMN. Open SSL is an open source tool kit to develop security protocols. You can find more information at:

`http://www.openssl.org`

enabled="boolean"

Required: true
Default: none
Valid Values: `true` or `false`

If the `enabled` value is set to true, it enables SSL connections for ONS.

wallet-file="path"

Required: false (for Oracle SSL only)

Default: none
Valid Values: The path name to the Oracle wallet.

The `wallet-file` attribute specifies the path name to the Oracle wallet to use for authentication on ONS connections. The *`$ORACLE_HOME`* directory may be used.

wallet-password="password"

Required: false (for Oracle SSL only)
Default: none
Valid Values: A string for the wallet password.

The `wallet-password` attribute is the password string for the specified Oracle wallet.

openssl-certfile="path

Required: true (for Open SSL only)
Default: none
Valid Values: The path name to the Open SSL cert file.

The `openssl-certfile` attribute specifies the Open SSL certificate file to use for authentication on ONS connections. The *`$ORACLE_HOME`* directory may be used.

openssl-keyfile="path

Required: true (for Open SSL only)
Default: none
Valid Values: The path name to the Open SSL key file.

The `openssl-keyfile` attribute specifies the Open SSL key file to use for authentication on ONS connections. The *`$ORACLE_HOME`* directory may be used.

openssl-password="password"

Required: true (for Open SSL only)
Default: none
Valid Values: A string for the `openssl-keyfile` password.

The `openssl-password` attribute is the password string for the specified `openssl-keyfile`.

openssl-lib="path"

Required: false (for Open SSL only)
Default: none
Valid Values: The path name to the Open SSL library.

The `openssl-lib` attribute specifies the path name to the Open SSL library where the shared library files (for example, `libssl` and `libcrypto`) reside. If the `openssl-lib` path name is not provided, then the environment from which OPMN is launched must include the path name in the proper system variable. The *`$ORACLE_HOME`* directory may be used.

<tune>

Required: false
Default: *See attributes*
Parents: notification-server

Attributes: `io-timeout`, `io-idle`, `timeout`

The `tune` element contains the information for the Tuneable Notification Server ONS parameters.

io-timeout="timeout"

Required: false
Default: 30
Valid Values: 0 or an integer value >= 4 and <= 3600

The `io-timeout` is the socket read timeout value in seconds used by each remote OPMN (or ONS) server directly connected to the local server. If the remote server does not receive any data across the connection with the local server in the configured `io-timeout` period, the remote server will time-out the connection and close the socket. The `io-timeout` value is the timeout period remote OPMN (or ONS) servers will use for the local server.

The `io-timeout` value is also used as a timeout for resource cleanup after an ONS connection has disconnected. If the connection is reestablished before the timeout period, the resources are transferred to the new connection. Otherwise the resources are released after the timeout period has expired.

Service failover participant state, duplicate ONS notification detection state, and notifications routed to the downed connection all use the `io-timeout` value. Notifications are queued to the down connection until the timeout occurs or the connection is reestablished.

The `io-timeout` parameter should be increased for servers running on very busy or overloaded systems.

A value of 0 disables `io-timeout` checking on remote servers for the local server. The `io-timeout` disables cleaning up of failover participant state, duplicate notification detection state, and notifications queued for the down connection. Notifications will continue to be queued until the connection is reestablished, which can consume an ever expanding amount of memory on the remote OPMN or ONS servers if the timeout check is disabled and the connection never comes back. The `io-timeout` parameter should only be set to 0 for debug purposes.

A configured non- zero value that is less than the minimum value is set to the minimum, and a value greater than the maximum is set to the maximum.

io-idle="interval"

Required: false
Default: `io-timeout` - (`io-timeout` / 3)
Valid Values: an integer value >= 2 and <= (`io-timeout` - 2)

The interval in seconds between sending a message to each remote server to which the local server is directly connected. If no normal network traffic is sent from the local server to any remote server in the configured interval, a message is sent to the remote server.

The `io-idle` parameter ensures that an idle but responsive OPMN server does not have its connection timed out when with a remote server. On busy systems, this value should be far enough below the `io-timeout` value such that there is enough time for the local server to queue and send the idle message to the remote server before the remote server detects the timeout.

If the `io-timeout` is 0, the `io-idle` attribute is ignored. A configured value that is less than the minimum value is set to the minimum, and a value greater than the maximum is set to the maximum.

timeout="timeout"

Required: false
Default: 20
Valid Values: an integer value >= 1 and <= 3600

The `timeout` parameter is the socket timeout value in seconds used for the local OPMN server for connection attempts and client connection writes. If the connection handshake to the local OPMN server takes longer than the configured timeout value, then the socket is closed and the connection resources are available. If a write on a client connection socket takes longer than the configured timeout value, then the socket is closed and the connection resources are available.

<topology>

Required: false
Default: none
Parents: notification-server
Attributes: none

The `topology` element contains the configuration information for the ONS topology within a cluster.

<nodes>

Required: false
Default: none
Parents: topology
Attributes: list

The `nodes` element provides a list of specific addresses for OPMN servers in the same cluster as the local OPMN server. The local OPMN server is included in the list. Multiple nodes elements may be configured.

list="list-of-nodes"

Required: true
Default: none
Valid Values: a list of nodes, with a comma separating each listed node. Each node represents an OPMN server in the cluster.

Each `node` entry represents an instance in the cluster to which the local OPMN server will attempt to connect to remote OPMN servers. The connection is needed to transfer information such as notifications, status, and user requests. If OPMN is unable to connect to a configured remote node, it will try again every 120 seconds. Duplicate entries are ignored.

Each `node` entry must be specified in the address:port format. The address must be a valid Transmission Control Protocol (TCP) IP address. Refer to Section 3.10, "IPv6 Support" for more information about supported IP connections for OPMN.

Each `node` value should correspond to an OPMN server running on the system. The OPMN server must be specified by address and configured to listen on the remote port specified by the `port` element.

The value of address should be the same as the one configured for the corresponding OPMN. If the system has multiple IP addresses or multiple host names, then the value of address must match the value configured for the remote attribute of the `ipaddr` element of the `notification server`.

The address portion of each node may consist of a list of host names or IP addresses for the same system. The lists are available only if the remote node `ipaddr` remote attribute is configured with the same set of information. The listed host names and IP addresses must be separated by a semi-colon.

The value of the `port` element must match the value configured for the remote attribute of the port element of the `notification-server` element.

For example:

- Configure host1 through host5:

```
<nodes list="host1.com:6200,host2.com:6200,host3.com:6200"/>
<nodes list="host4.com:6200,host5.com:6200"/>
```

- Configure a host that has two names:

```
<nodes list="hostnameA.com;hostnameB.com:6200"/>
```

<discover>

Required: false
Default: none
Parents: topology
Attributes: list

The `discover` element provides a list of discovery service addresses. The local OPMN server will use the discover element to find and connect with remote OPMN servers in the same cluster. You can configure multiple discover elements.

list="list-of-services"

Required: true
Default: none
Valid Values: a list of discovery multi-cast addresses or a discovery OPMN host addresses. A comma must be used to separate the list of services.

Each service entry represents a discovery resource through which the local OPMN server will locate remote OPMN servers in the same cluster. Each OPMN server in a discovered topology will automatically manage its connections to other OPMN servers in the same topology. A service may be one of two types: a multicast node or a host node.

The syntax for the multicast node is `*node`. `*node` is a multi-cast IP address (or host name) and port pair (refer to `nodes`). OPMN will treat each unique *node entry as a separate topology of instances to join, although typically any given OPMN server will participate in only a single discover topology.

For example:

```
<discover list="*225.0.0.37:8205"/>
```

The multicast address must be within the valid address range, which is 224.0.0.1 to 239.255.255.255. The asterisk (`*`) preceding the IP address is critical, because it informs OPMN that the value specified is a multicast address.

The syntax for the discover server node is node. node is an IP address (or host name) and OPMN remote port pair for an OPMN instance in the cluster that will serve as a discovery host in a topology. Each topology may have multiple discovery servers configured.

For example:

```
<discover list="host1.com:6200"/>
```

Within each discover topology, OPMN will maintain a circular connection of servers. For clusters with groups of instances in locations remote from one another, instances should be grouped in separate topologies based upon location, in which case, the separate topologies must be joined through the gateway mechanism.

For discovery topology loops with large numbers of instances, OPMN will automatically break the circular connection loop into smaller sub loops in order to minimize latency from passing a notification from server to server. By default, OPMN will auto-break a circular loop when the number of nodes exceeds 16. You can configure this auto-break threshold by prefacing the discovery list with a `[number]` prefix, where `number` is the maximum number of instances OPMN should allow in a topology loop before breaking it into multiple loops. The minimum value for number is eight.

For example:

- Set the auto-break number to 8 for a discovery multi-cast topology:

  ```
  <discover list="*[8]225.0.0.37:8205"/>
  ```

- Set the auto-break number to 8 for a discovery host topology:

  ```
  <discover list="[8]host1.com:6200,host2.com:6200"/>
  ```

<gateway>

Required: false
Default: none
Parents: topology
Attributes: list

The `gateway` element is used to connect isolated discovery topologies. You can configure multiple gateway elements.

list="list-of-gateways"

Required: true
Default: none
Valid Values: a list of gateways. Each gateway represents a link or set of links between two OPMN discovery topologies. Each listed gateway must be separated by a comma.

The gateway syntax is `source/target`.

The `source/target` are lists of node entries. Each node entry on the list is separated by an ampersand (&). For more information refer to `nodes` . Each OPMN server listed in the source will connect to each OPMN server listed in the target. If the target is omitted, then it is assumed to be the same as the source. OPMN servers that are not listed in the source ignore the gateway specification.

For example:

The OPMN server on host1a is part of a discovery topology that is isolated to subnet subA, and has been selected to be the gateway instance between that topology and the discovery topology isolated on subnet subB. The OPMN server on host1b is part of a discovery topology that is isolated to subnet subB, and has been selected to be the gateway instance between that topology and the discovery topology isolated on subnet subA.

```
<gateway list="host1a.subA.com:6200&host1b.subB.com:6200/"/>
```

With this configuration, OPMN server `host1a.subA.com:6200` will connect to OPMN server `host1b.subB.com:6200`, and OPMN server `host1b.subB.com:6200` will connect to OPMN server `host1a.subA.com:6200`.

For maximum redundancy you may wish to configure more than one host as the gateway for each discovery topology.

For example:

The OPMN server on host1a is part of a discovery topology that is isolated to subnet subA, and has been selected to be the gateway instance between that topology and the discovery topology isolated on subnet subB. The OPMN server on host1b is part of a discovery topology that is isolated to subnet subB, and has been selected to be the gateway instance between that topology and the discovery topology isolated on subnet subA.

This is the same as the previous example, but with two hosts configured from each subnet.

```
<gateway
list="host1a.subA.com:6200&host2a.subA.com:6200&host1b.subB.com:6200&host2b.subB.c
om:6200/"/>
```

However with this configuration the two instances on subnet A will always connect to one another and the two nodes on subnet B will always connect to one another, which may result in uneccessary connection overhead.

A more precise configuration requires two separate gateway elements:

```
<gateway
list="host1a.subA.com:6200&host2a.subA.com:6200/host1b.subB.com:6200&host2b.subB.c
om:6200"/>
<gateway
list="host1b.subB.com:6200&host2b.subB.com:6200/host1a.subA.com:6200&host2a.subA.c
om:6200"/>
```

Now the instances on the same subnet will only connect to one another if needed.

<process-manager>

Required: true
Default: none
Parents: <opmn>
Attributes: `insecure-remote-requests`

The `process-manager` contains the configuration definitions for the PM portion of OPMN.

insecure-remote-requests="boolean"

Required: false
Default: false
Valid Values: `true` or `false`

The `insecure-remote-request` attribute is a security check which disables requests until security features are configured. By default OPMN will only allow start, stop, restart, shutdown and reload requests rerouted from remote OPMN servers if ONS SSL is `enabled` and a wallet file is configured for authentication.

> **Note:** Setting the `insecure-remote-request` attribute to `true` overrides that security check and enables these requests to be issued remotely with no security features configured.
>
> Setting the insecure-remote-request attribute to `true` is a major security risk and should only be done for testing purposes with all connected OPMN servers behind a well secured fire wall or completely disconnected from any external network.

<process-modules>

Required: true
Default: none
Parents: <process-manager>
Attributes: none

Each `process module` is designed to support a specific set of `process-type` elements; it is only required if the `process-type` elements are configured. The PM dynamically loads in a library for each specified `process module`.

<module>

Required: true
Default: none
Parents: <process-modules>
Attributes: `path, tag, status, cron`

The `module` element is used to provide `process-type` specific support for the PM. Each module is implemented as a shared library. The module exports a set of standard functions and uses the PM `process module` API. A module must provide a list of the `process-types` it supports. Only one configured `process module` may list a specific `process-type`. No two modules can list the same `process-type`.

path="path"

Required: true
Default: none
Valid Values: The path name for the module shared library.

The `path` attribute must specify the path name of the shared library file. The library file has the system suffix of `.so` for Linux and `.dll` for Microsoft Windows. The suffix may be omitted and OPMN will automatically append it. *`ORACLE_HOME`* may be used when specifying the path name.

tag="tag-id"

Required: false
Default: The value specified by the path element.
Valid Values: A string uniquely identifying the module.

The tag attribute identifies the module. A module may report its tag value when logging errors to the PM log file or as part of the response to a request. While optional, it is a good idea to set this attribute to a meaningful value to help track any issues with process management.

status="state"

Required: false
Default: `enabled`
Valid Values: `critical, enabled, or disabled`

A module may be `enabled`, in which case PM loads in its shared library when it starts and calls the module's initialization functions, or `disabled` in which case the module entry is completely ignored. If the module `process-types` are configured in the `opmn.xml` file they must also be `disabled`. The `critical` state is the same as `enabled`, except that OPMN will terminate with a fatal error code if the module initialization fails.

cron="interval"

Required: false
Default: none
Valid Values: An integer.

Specify the interval in seconds between calls to the module's `cron` callback function. Configuring a `cron` interval for a module that does not support the `cron` callback is not allowed. Unless you have designed the module, you should neither add nor alter this attribute.

<module-data>

Required: false
Default: none
Parents: <module> , <ias-instance> , <ias-component> , <dependencies> , <process-set>
Attributes: none

The `module-data` blocks are used to define module specific name-value pairs that are meaningful only to a specific module. Each `module-data` block is organized into categories, which contain the name-value data pairs.

The `module-data` blocks can be defined for multiple elements within `the opmn.xml file`, and OPMN will create an aggregate `module-data` block at the `process-set` level that contains all values defined at or above it. If multiple definitions exist in this hierarchy with the same `category id` and `data id`, the value defined at the lowest level is used.

Table 6–5 illustrates the `module-data` defined at each level in the hierarchy (with the highest level displayed at the top) and the resultant union at the `process-set` level of all of the `module-data` definitions:

Table 6–5 module-data Hierarchy

Module	Definition

`ias-instance`	`<category id="CatA">`
	`  <data id= "DataAA" value="aaaa"/>`
	`</category>`

Table 6–5 (Cont.) module-data Hierarchy

Module	Definition
ias-component	<category id="CatA"> <data id= "DataAB" value="abab"/> </category> <category id="CatB"> <data id= "DataBA" value="baba"/> </category>
module	<category id="CatA"> <data id= "DataAC" value="acac"/> </category>
process-type	<category id="CatA"> <data id= "DataAA" value="XXXX"/> </category>
process-set	<category id="CatB"> <data id= "DataBB" value="bbbb"/> </category>
RESULT	<category id="CatA"> <data id= "DataAA" value="XXXX"/> <data id= "DataAB" value="abab"/> <data id= "DataAC" value="acac"/> </category> <category id="CatB"> <data id= "DataBA" value="baba"/> <data id= "DataBB" value="bbbb"/> </category>

<category>

Required: true
Default: none
Parents: <module-data>
Attributes: id

The category element is an organizational level within a module-data block.

id="id"

Required: true
Default: none
Valid Values: A string.

This id string identifies a data category. Each category id within a single module-data block must be unique, but multiple module-data blocks may contain the same data category ids, in which case the categories are considered to be related.

<data>

Required: true
Default: none
Parents: <category>
Attributes: id, value, process-conversion

A data name value definition within a module-data category.

id="id"

Required: true
Default: none
Valid Values: A string.

This string identifies a data attribute. Each `data id` within a single category must be unique, but multiple categories may contain the same data identifications. Data elements with the same identification as other data elements, that are defined in different categories with the same identification, are related.

value="value"

Required: true
Default: none
Valid Values: A string.

The value string associated with the data element `id`. Any environment variable defined anywhere within the scope of the `process-set` (any level at or above the `process-set`) in which the data value is defined (again, any level at or above the `process-set`) referenced within the value string as `$variable` or `%variable%` will be expanded to the variable value.

process-conversion="boolean"

Required: false
Default: `true`
Valid Values: `true` or `false`

The `process-conversion` attribute enables you to set the separator character. By default OPMN will convert slashes in the data value string to be those of the directory path separator character for the system on which OPMN is running (on UNIX each `\` character is converted to `/` and on Windows each `/` is converted to `\`). Set this attribute to false to disable this conversion.

Note that if process-conversion is true, OPMN uses the ^ character as an escape character to disable process conversion for the following character, and so `^/` on a Windows system will yield a `/` in the string. Specify two ^ characters if you need to specify the ^ character in the resultant string: ^^ yields ^.

<data id="routing-id">

Required: false
Default: none
Parents: category
Attributes: value

The `data id` defines the value for the routing-id module.

For more information about the `routing-id` refer to the Oracle HTTP Server Administrator's Guide.

value="routing id"

Required: true
Default: none
Valid Values: Any string.

The Routing ID specifies a routing relationship between OC4J and Oracle HTTP Server. In other words, an Oracle HTTP Server routes to every OC4J that it shares a

routing ID with. Out of the box, the routing ID is specified as a module data under the `ias-instance` in the `opmn.xml` file. The value of the default `routing-id` is `g_rt_id`. Because of the hierarchical nature of the `opmn.xml` file, every component configured in the `opmn.xml` file inherits the configured routing ID. The user can configure a separate routing ID for any Oracle HTTP Server or OC4J by configuring a routing ID at a lower level in the `opmn.xml` file. As with most entities in the `opmn.xml` file, like entities configured at a lower level override those configured at a higher level.

<module-id>

Required: true
Default: none
Parents: <module>
Attributes: `id`

The `module-id` name defines the type of process and associates the configuration with a `process module`.

This identifier is used by each `process-type` to specify which module supports it. A module may be configured with multiple `module-ids`.

id="module-id"

Required: true
Default: none
Valid Values: A string.

<ias-instance>

Required: true
Default: none
Parents:<process-manager>
Attributes: `id`, *`ORACLE_HOME`*

The configuration definitions for an Oracle Application Server instance. Only one `ias-instance` is supported for each OPMN.

`id` = `ias-instance-id`; `name` = `ias-instance-name`; *`ORACLE_HOME`* = "path name"

id="ias-instance-name"

Required: true
Default: none
Valid Values: A string.

The string specifies the `ias-instance` id.

name="ias-instance-name"

Required: false
Default: `ias-instance-id`
Valid Values: A string.

The `name` string specifies the `ias-instance` name, and it is this value that is used to identify the instance for a scoped OPMN request.

ORACLE_HOME="path"

Required: true
Default: none
Valid Values: A path string.

This `ORACLE+_HOME` path attribute must be the *`ORACLE_HOME`* equivalent for the Oracle Application Server instance.

<environment>

Required: true
Default: *Refer to the following paragraph.*
Parents: ias-instance , `ias-component` ,`process-type` , `process-set`
Attributes: none

Like `module-data` blocks, `environment` blocks can be defined for multiple elements within the `opmn.xml` file, and OPMN will create an aggregate `environment` block at the `process-set` level that contains all values defined at, or above it. If multiple definitions exist in the hierarchy with the same `id`, the value defined at the lowest level is used.

Note: OPMN sets the following default environment variables at the `ias-instance` level, with the values extracted either from the `ias-instance` configuration or from the OPMN run time environment:

Linux: `ORACLE_HOME, ORACLE_NLS, OPMN_ENV_LC_ALL, OPMN_ENV_LANG, OPMN_ENV_NLS_LANG, SHELL`

Microsoft Windows: `COMSPEC, SYSTEM_DRIVE, and SYSTEM_ROOT`

<variable>

Required: true
Default: none
Parents: <environment>
Attributes: `id, value, append, process-conversion`

The variable element defines the environment variable name and value.

id="id"

Required: true
Default: none
Valid Values: A string.

The `id` is the environment variable name. An `environment id` may be duplicated within an `environment` block, with the last defined value taking priority over earlier definitions. The same `environment id` may be defined within `environment` blocks for different elements, and the value defined at the lowest level will take priority over values defined at higher levels.

value="value"

Required: true
Default: none
Valid Values: A string.

The environment value. Environment variables referenced within the value string as `$variable` or `%variable%` will be expanded to the variable value. The same environment variable may reference itself to use a definition defined at a higher level, or earlier within this same `environment` block.

You may use the Linux shell syntax for referencing an environment variable, `$variable` or `${variable}`, or the Microsoft Windows format *`%variable%`*. Referenced variables that have not been defined remain in place as referenced, and so `value="_notdefined_"` would remain unchanged.

For example, the following environment block yields a value for `accumulate` of "`foobar`".

```
<environment>
   <variable id="accumulate" value="foo">
   <variable id="accumulate" value="${accumulate}bar">
</environment>
```

append="boolean"

Required: false
Default: false
Valid Values: `true` or `false`

You can force OPMN to append the new environment variable value to the previously defined value, with the system library delimeter placed in between the two values (':' for Linux and ';' for Microsoft Windows) by specifying a value of `true` for this attribute. This is useful when assembling a value for a variable such as `CLASSPATH`.

For example, the following `environment` block yields a value for `CLASSPATH` of "`/foo:/bar`" on a Linux system.

```
<environment>
   <variable id="CLASSPATH" value="/foo">
   <variable id="CLASSPATH" value="/bar" append="true">
</environment>
```

process-conversion="boolean"

Required: false
Default: `true`
Valid Values: `true` or `false`

<rmd-definitions>

Default: none
Parents: ias-instance , `ias-component` ,`process-type` , `process-set` , process-manager
Attributes: none

The RMD definitions provide the ability to have process management commands issued based on system conditions according to a set of user-configured directives. The system conditions that can be taken into account include any DMS metrics that are available within the Oracle Application Server instance. This includes preset metrics that are always present as well as metrics that are defined and implemented by other applications.

RMD definitions are either:

- Hierarchical: if defined at the `ias-instance` level or lower. Hierarchical RMDs assume an association within the OPMN configuration components in which they are defined.
- Global: if defined at the process-manager level. Global RMDs require explicit OPMN component specifications.

<rmd>

Required: true
Default: none
Parents: rmd-definitions
Attributes: name, description, interval

A directive specifying a condition and a set of actions to take when the condition evaluates true and an optional set of `exceptions` to enact should any action fail.

name="name"

Required: true
Default: none
Valid Values: An identifier for the RMD.

For hierarchical RMDs the name must be unique within the hierarchical component in which it was defined. For global RMDs the name must be unique within the `global rmd-definitions` block.

description="description"

Required: false
Default: none
Valid Values: A string.

A description of the RMD.

interval="interval"

Required: false
Default: 30
Valid Values: An integer greater than or equal to five or the string `"joinons group"`.

The `interval` attribute specifies the period between conditional checks for the RMD. The actual time elapsed between checks may be somewhat greater than the configured `interval` depending upon system load. A smaller value yields more frequent checks of the RMD at the expense of additional system resources.

If `interval` is set to the string "`joinonsgroup`", then the RMD is evaluated when the ONS layer detects that the OPMN server has joined a new ONS topology group.

<conditional>

Required: true (false if `rmd` attribute interval="`joinonsgroup`")
Default: none
Parents: rmd
Attributes: CDATA

The `conditional` element contains the conditional statement to be evaluated by OPMN.

<![CDATA[statement]]/>

Required: true
Default: none
Valid Values: A string (the conditional statement).

The CDATA conditional statement describes a state of the system that will trigger an `action`. The statement consists of a logical combination of comparisons between values. Like all other aspects of OPMN, conditionals are case sensitive. The values that may be used can be the following types:

- DMS metrics available from the OPMN DMS tree
- Constant values (for example, 500000)
- Temporal values (for example, 5 p.m.)

DMS metrics are described based on their location in the OPMN DMS tree. The description can be a fully qualified path name, a hierarchical relative reference, or a global absolute reference.

A fully qualified path name to the target metric begins with a '/'. The fully qualified path name includes the full path name to the OPMN instance. The OPMN instance prefix is automatically added to fully qualified path names. For example:

```
/pm/host_statistics/freePhysicalMem
```

Metrics with fully qualified path names can be referenced by both hierarchical and global RMD `conditionals`.

If the RMD element is hierarchical, then the metric must use a relative path name form. For example:

```
[process-set].numProcs
```

which specifies the `numProcs` metric for any `process-set` in the hierarchy for which the RMD is evaluated. The exception to this rule is the `application` metric, in which the path name may be specified as either relative or absolute. The following component specifications are allowed for hierarchical RMD `conditionals`:

- `[ias-component]` Refers to the `ias-component` element within the hierarchy of the `opmn.xml` file.
- `[process-type]` Refers to the `process-type` element within the hierarchy of the `opmn.xml` file.
- `[process-set]` Refers to the `process-set` element within the hierarchy of the `opmn.xml` file
- `[process]` Refers to the `process` element within the run time hierarchy
- `[application]` Refers to the application element within the run time hierarchy
- `[application=<app>]` Refers to the application <app>.

If the RMD element is global, then the metric must be an absolute description of a starting point followed by a relative path name. For example:

```
[ias-component=HTTP_Server][process-set=HTTP_Server].numProcs
```

specifies the metric `numProcs` for the `process-set` HTTP_Server that belongs to the ias-component=HTTP_Server. The exceptions to this rule are:

- `process`: must always be specified as as relative
- `application`: may be specified as either relative or absolute.

The following component specifications are allowed for global RMD conditionals:

- `[ias-component=<comp>]` Refers to the ias-component `<comp>` in the `opmn.xml` file
- `[process-type=<ptype>]` Refers to the process-type `<ptype>` in the `opmn.xml` file
- `[process-set=<pset>]` Refers to the `process-set` `<pset>` in the `opmn.xml` file.
- `[process]` Refers to a process
- `[application]` Refers to an application
- `[application=<app>]` Refers to the application `<app>`.

As with OPMN process requests, components not specified in the directive are assumed to be wildcards; thus a global RMD referencing [`process-set`=home] would be evaluated for every `process-set` with the `id` "`home`" configured in the `opmn.xml` file.

A conditional cannot reference more than one OPMN logical component. For example, the same conditional cannot reference `[ias-component=HTTP_Server]`. Similarly, references to lower configuration components must not belong to a referenced higher component within the same conditional.

For hierarchical and global RMDs multiple bracketed sections can be specified but each subsequent section must further narrow the tree. That is, the order must progress from the top of the tree toward the leaf. For example, a `process` cannot appear before an `ias-component`.

Constant values must be simple numeric values, including decimal values. When comparing a constant value with a DMS metric, the constant must have the same units as the DMS metric it is being compared to. OPMN will convert the numeric values so that the values are similar.

Temporal values consist of a 24 hour format (hh:mm) time combined with an optional day of the week indicator. The day of the week abbreviations are `sun, mon, tue, wed, thu, fri` and `sat`. The day of the week indicators can be a single value, a comma separated list, or a dashed inclusive list. For example:

- `17:00` Daily at 5 PM
- `[mon,wed,fri].05:00` Monday, Wednesday, Friday at 5 AM
- `[mon-fri].21:45` Week days at 9:45 PM

The current time is represented by the key word `{time}`. A time comparison is evaluated false if the day of the week specified on the left of the operator does not include a day of the week specified on the right.

By default, the day of the week is only evaluated if the values are equal. If the day of the week evaluation is true then the time value is also evaluated.

You can force the comparison to be made against a single day of the week by appending the '@' character after the day, with Sunday (sun) having the value 0 and Saturday (sat) 6. For example:

- `{time} < [wed].12:00` Evaluates true only on Wednesday from 00:00 until noon
- `{time} < [wed@].12:00` Evaluates true from Sunday at 00:00 until Wednesday at noon

Comparisons between values include less than (<), less than, or equal to (<=), greater than (>), greater than or equal to (>=), equals (=), or not equals (!=).

String values may also be specified between quotes (using the single quote character """), but the only allowable operators are equals (=) and not equals (!=).

Comparisons may be logically combined using the operators and (&) or (|), and grouping (parentheses). The logical operator (!) can be placed before any logical group of comparisons to negate the result of the outcome.

The key word `{duration(interval)}` may also be used to indicate that a conditional should only trigger if it has evaluated as true over the specified time interval (expressed in seconds). It is important to note that the conditional is only evaluated on a periodic basis, and specifying the duration value will force the conditional to only trigger if the evaluations over the time period meet the conditional value. This is an approximation of ensuring that the condition holds true over the entire time period. The accuracy of this approximation depends on the ratio of the duration interval value to the evaluation configured interval. When a conditional is evaluated as true, all durations for that evaluation are reset. When a conditional evaluates false, any duration that was not encountered during the evaluation is also reset.

If interval is set to the string `joinonsgroup`, then the conditional is only evaluated when the ONS layer detects that the OPMN server has joined a new ONS topology group. If the conditional is omitted, then the RMD evaluation will always be true. If the conditional is defined, then `process` level references are not allowed and the duration key word should not be used. For example:

- If the heap size of a JVM has exceeded 500 Megabytes (global RMD):

  ```
  [process-set=home][process].heapSize > 500000
  ```

- If the time is 5 PM on a weekday:

  ```
  {time} = [Mon-Fri].1700
  ```

If the average request time is greater than 500 milliseconds for at least 60 seconds, and there are less then four processes running for the `process-set` at which the hierarchical RMD was configured:

```
[process].avgReqTime > 500 {duration(60)})&([process-set].numProcs < 4)
```

If there has been less than 50 MBs of free system memory for the last three minutes:

```
/pm/host_statistics.freePhysicalMemory < 50000 {duration(180)}
```

<action>

Required: true
Default: none
Parents: rmd
Attributes: value

The `action` element indicates an action to be taken if the `conditional` evaluates true.

value=request

Required: true
Default: none
Valid Values: A string (the request).

An `action` can be either a process management request, or a specified program to execute, when a RMD conditional evaluates true. An RMD may have multiple `actions`, and they are executed sequentially in the order in which they are defined.

If an `action` request returns a value other than success (200), or the executed program exits with a status other than 0, then the remaining `actions` for the RMD will be skipped, and any configured `exceptions` will be performed.

An RMD will not be evaluated while either its `actions` or `exceptions` are in progress.

The possible `actions` for an RMD are:

- `start`: perform a `start` request with the given argument list
- `restart`: perform a restart request with the given argument list
- `stop`: perform a stop request with the given argument list
- `exec` : execute the given program or script with the listed arguments

The targets for the `start`, `restart`, and `stop` requests are assumed to be relative to the OPMN components referenced in the RMD conditional. Key words representing these components should be used to narrow the scope of the request.

The following set of key words, with expansion, is provided:

- `{ias-component} ias-component=<comp>`
- `{process-type} process-type=<ptype>`
- `{process-set} process-set=<pset>`
- `{process} uniqueid=<uid>`
- `{application} application=<app>`

For `start`, `restart`, and `stop` `actions`, the key word {`process`} should not be used with the other key words because OPMN requests do not allow this combination.

In addition the `exec` `action` can use the `{pid}` key word, which expands to `pid=<pid>` if the `conditional` refers to a `process-set` or `process`.

Note that an `action` cannot reference a key word for an OPMN component at a level below the lowest OPMN component referenced in the `conditional` (if the `conditional` only references an ias-component, then the `action` can only use the {ias-component} key word, for example). The exception to this rule is that an `action` may reference {`process`} if the `conditional` references a `process-set`, but this will force the RMD to be evaluated at the `process` level and not the `process-set` level.

The results of the `action` requests are logged in the OPMN process manager log (the beginning and completion of the request with completion status at level 4, and full results at level 5). The `stdout` and `stderr` output of executable programs or scripts are sent to *`$ORACLE_HOME`*`/opmn/logs/rmd.out`. There is no rotation performed on the `rmd.out` file. Other programs and scripts should maintain and use their own log files and not print output using the `stdout` and `stderr` file descriptors.

The following are some `action` examples:

- Start another JVM in the home OC4J instance within a hierarchical RMD defined within the *`ORACLE_HOME`* instance (note the `process-set` must be configured with `minprocs/maxprocs`):
- `start {ias-component}{process-type}{process-set} numprocs=1`

Restart a JVM in the "home" instance within a hierarchical RMD defined within the "home" instance:

- `restart {process}`

Stop the entire `ias-component` referred to by the RMD `conditional`:

- `stop {ias-component}`

Execute the given program and pass in the referenced `process` UID and pid:

- `exec` *`$ORACLE_HOME`*`/mybin/report.sh "rmd triggered" {process} {pid}`

timeout=timeout

Required: false
Default: the configured (or default) timeout for the OPMN request for affected components (see following paragraph)
Valid Values: An integer > 0 and < 3600

A timeout value can be configured for each `action`. The default timeout for `start`, `restart`, and `stop` `actions` is the configured (or default) timeout for the OPMN request on these components. The default timeout for `exec` is the configured (or default) timeout for the OPMN request on these components. The default timeout for `exec` `action` is 30 seconds.

<exception>

Required: false
Default: none
Parents: rmd
Attributes: value

An action to be taken if the any of the `actions` definitions for the RMD fail.

The configuration is identical to `action`.

If an `exception` request returns a value other than success (200), or the executed program exits with a status other than 0, then the remaining `exceptions` for the RMD will be skipped.

<ias-component>

Required: true
Default: none
Parents:<ias-instance>
Attributes: `id, id-matching, status`

An `ias-component` is a logical grouping of `process-type` for administrative purposes.

id="component-id"

Required: true
Default: none
Valid Values: a string

The `id` attribute uniquely identifies the `ias-component` within the `ias-instance`.

id-matching="boolean"

Required: false
Default: false
Valid Values: `true` or `false`

By default OPMN requests that do not specify `ias-components` match all configured `ias-components`, unless the `id-matching` attribute for a component is set to `true`, in which case the request must explicitly include the `ias-component id` in order to affect the `ias-component` or any `process-type` or `process-set` configured for that `ias-component`.

status="state"

Required: false
Default: `enabled`
Valid Values: `enabled` or `disabled`

An `ias-component` may be `enabled`, in which case OPMN parses all of its configured attributes and elements and enables requests to operate upon it, or `disabled`, in which case the `ias-component` entry is completely ignored.

<dependencies>

Required: false
Default: none
Parents: <ias-component> , <process-type> , <process-set>
Attributes: none

OPMN uses `dependencies` to determine if a process should be started or not. Like `module-data` and `environment` block, dependencies blocks can be defined for multiple elements within the `opmn.xml` file, and OPMN will create an aggregate dependency list at the `process-set` level that contains all dependencies defined at or above it. If duplicate dependencies are defined at different levels, then duplicate checks on that dependency will be made before starting a process. OPMN will create an aggregate dependency list at the `process-set` level that contains all dependencies defined at or above it. If duplicate dependencies are defined at different levels, then duplicate checks on the dependency will be made before starting a process.

There are two primary types of dependencies: external and internal. External dependencies are for those resources not managed by OPMN. For example: Application Server Control Console.

An external program is executed by OPMN to perform the check on the resource. Internal dependencies are for OPMN-managed processes (unit), which may include processes managed on a remote OPMN.

OPMN maintains a cache of dependency states which contains the last known state of each dependency, and the time it was last checked. A `cache-timeout` parameter for each dependency enables users to specify how long to use its state in the cache, or if it should be used at all. Similarly, a general timeout parameter for each dependency will determine how long OPMN will wait for a status update from that dependency before aborting the dependency check and the process start.

OPMN checks dependencies in the order in which they are declared. The traversal of the list of dependencies concludes either with the full sequence of successful checks, the dependency is available, or the first check failure, the dependency is not available, or the dependency check timed out.

<database>

Required: false
Default: none
Parents: <dependencies>
Attributes: `db-connect-info, infrastructure-key, timeout, cache-timeout`

The `database` element specifies the database to check. Either `db-connect-info` or `infrastructure-key` is used to identify the database.

db-connect-info="connect"

Required: true if `infrastructure-key` is not specified.
Default: none
Valid Values: A string

The `db-connect-info` attribute is the string required to connect to the database. The string can be in either of the following formats:

- `<host>:<port>/<service name>`
- `(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp) (HOST=<host>) (PORT=<port>)) (CONNECT_DATA=(SERVICE_NAME=<service name>)))`

For example:

`pdsundev7:1521/asdb.us.oracle.com`

Database connection information can be found in `$ORACLE_HOME/network/admin/tnsnames.ora`.

infrastructure-key="key"

Required: true if `db-connect` is not specified.
Default: none
Valid Values: A string

The `infrastructure key` attribute is required to identify the database.

timeout="depend-timeout"

Required: false
Default: `1200`
Valid Values: An integer

The `timeout` attribute specifies in seconds how long OPMN will wait for a dependency check to complete. If the check takes longer than the configured timeout, then OPMN will consider the check to have failed.

cache-timeout="cache-timeout"

Required: false
Default: `600`
Valid Values: An integer

The `cache-timeout` attribute specifies how long in seconds OPMN will use the current "up" status for the dependency entry in the cache. If the last successful dependency check was within the prescribed number of seconds from the current check, then the dependency check is instantly flagged as successful, otherwise another dependency check will be performed. Note that the `cache-timeout` is only for the

last successful check of the dependency, and if the previous check failed, another access of the dependency will be performed for this check. A value of 0 indicates OPMN will always perform the check.

<OID>

Required: false
Default: none
Parents:<dependencies>
Attributes: `address, infrastructure, timeout, cache-timeout`

The `<OID>` element specifies the Oracle Internet Directory service to check either an address string for a specific Oracle Internet Directory, or that the OracleAS Infrastructure flag is set to `true` to use the default infrastructure Oracle Internet Directory.

address="address"

Required: true
Default: none
Valid Values: A string

The `address` element is the address string required to connect to Oracle Internet Directory.

infrastructure="boolean"

Required: true if `address` is not set.
Default: none
Valid Values: `true` or `false`

Use the default infrastructure Oracle Internet Directory for the Oracle Application Server instance.

<ssl>

Required: false
Default: none
Parents: <OID>
Attributes: `enabled, wallet-file, wallet-password`

The SSL information for the Oracle Internet Directory connection.

enabled="boolean"

Required: true
Default: none
Valid Values: `true` or `false`

To enable SSL on the Oracle Internet Directory connection, set the `enabled` attribute to `true`.

wallet-file="path"

Required: true
Default: none
Valid Values: A valid path name

The path name to the wallet file for authentication of the Oracle Internet Directory connection. The *$ORACLE_HOME* directory location may be used.

wallet-password="password"

Required: false
Default: none
Valid Values: A string

The `wallet-password` attribute is the password for the specified wallet-file.

timeout="depend-timeout"

Required: false
Default: `1200`
Valid Values: An integer

The `timeout` attribute specifies in seconds how long OPMN will wait for a dependency check to complete. If the check takes longer than the configured timeout, then OPMN considers the check to have failed.

cache-timeout="cache-timeout"

Required: false
Default: `600`
Valid Values: An integer

The `cache-timeout` attribute specifies how long in seconds OPMN will use the current "up" status for the dependency entry in the cache. If the last successful dependency check was within the prescribed number of seconds from the current check, then the dependency check is flagged as successful. Otherwise, OPMN performs another dependency check. The `cache-timeout` is only for the last successful check of the dependency. If the previous check failed, OPMN performs another access of the dependency check. A value of 0 indicates OPMN will always perform the check.

<OSSO>

Required: false
Default: none
Parents: <dependencies>
Attributes: `host, port, URI, timeout, cache-timeout`

The `OSSO` element specifies the OracleAS Single Sign-On service to check.

host="hostname"

Required: true
Default: none
Valid Values: A string

The `host` attribute is the hostname for the OracleAS Single Sign-On connection.

port="port"

Required: true
Default: none
Valid Values: A port number

The `port` attribute is the port for the OracleAS Single Sign-On connection.

URI="uri"

Required: true
Default: none
Valid Values: A string

The `URI` attribute is the URI for the OracleAS Single Sign-On connection.

<ssl>

Required: false
Default: none
Parents: <OSSO>
Attributes: `enabled, wallet-file, wallet-password`

The `ssl` element is the SSL information for the OracleAS Single Sign-On connection.

enabled="boolean"

Required: true
Default: none
Valid Values: `true` or `false`

The `enabled` attribute enables the SSL connection OracleAS Single Sign-On. To enable the connection set this attribute to `true`.

wallet-file="path"

Required: true
Default: none
Valid Values: A path name

The `wallet-file` attribute is the path name to the wallet file for authentication of the OracleAS Single Sign-On connection. The *`ORACLE_HOME`* value may be used.

wallet-password="password"

Required: false
Default: none
Valid Values: A string

The `wallet-password` is the password for the specified wallet-file.

timeout="depend-timeout"

Required: false
Default: `1200`
Valid Values: An integer

The `timeout` attribute specifies in seconds how long OPMN will wait for a dependency check to complete. If the check takes longer than the configured timeout, then OPMN considers the check to have failed.

cache-timeout="cache-timeout"

Required: false
Default: `600`
Valid Values: An integer

The `cache-timeout` attribute specifies how long in seconds OPMN will use the current "up" status for the dependency entry in the cache. If the last successful dependency check was within the prescribed number of seconds from the current check, then the dependency check is flagged as successful. Otherwise, OPMN performs another dependency check. The `cache-timeout` is only for the last successful check of the dependency. If the previous check failed, OPMN performs another dependency check. A value of 0 indicates OPMN will always perform the check.

<managed-process>

Required: false
Default: none
Parents: <dependencies>
Attributes: `ias-instance, ias-component, process-type, process-set, autostart, autostop, timeout, cache-timeout`

The `managed-process` attribute specifies the managed process to check. A process for `process-type` or `process-set` does not start unless the specified dependency managed process is active. Circular dependencies are detected and rejected for local managed processes, but not for remote managed processes; this may result in a dependency check deadlock, which times out.

ias-instance="ias-instance-id"

Required: false
Default: The `ias-instance` of the current `process-type` or `process-set`.
Valid Values: A string

The `ias-instance` for the managed process dependency. If the specified `ias-instance` is not managed by the current OPMN, it is assumed to be a remote managed process dependency.

ias-component="ias-component-id"

Required: true
Default: none
Valid Values: A string

The `ias-component` for the managed process dependency.

process-type="process-type-id"

Required: true
Default: none
Valid Values: A string

The `process-type-id` for the managed process dependency.

process-set="process-set-id"

Required: true
Default: none
Valid Values: A string

The `process-set-id` for the managed process dependency.

autostart="boolean"

Required: false
Default: false
Valid Values: `true` or `false`

The `autostart` attribute is used for the managed process dependency. If the process is not running when the check is performed, the `autostart` element will OPMN to attempt to start it.

autostop="boolean"

Required: false
Default: false
Valid Values: `true` or `false`

When the managed process dependency is stopped, then stop the managed process. The attribute is always `false` for remote managed process dependencies.

timeout="depend-timeout"

Required: false
Default: `1200`
Valid Values: An integer

The `timeout` attribute specifies, in seconds, how long OPMN will wait for a dependency check to complete. If the check takes longer than the configured timeout, then OPMN considers the check to have failed.

cache-timeout="cache-timeout"

Required: false
Default: `600`
Valid Values: An integer

The `cache-timeout` attribute is only used for a process managed by a remote OPMN. The `cache-timeout` attribute specifies how long in seconds OPMN will use the current "up" status for the dependency entry in the cache. If the last timeout dependency check was within the prescribed number of seconds from the current check, then the dependency check is instantly flagged as successful, otherwise OPMN performs another dependency check. Note that the `cache-timeout` is only for the last successful check of the dependency, and if the previous check failed, OPMN another access of the dependency will be performed for this check. A value of 0 indicates OPMN will always perform the check.

Note: The `cache-timeout` is only for the last successful check of the dependency, and if the previous check failed, OPMN will perform another dependency check.

<process-type>

Required: true
Default: none
Parents: <ias-component>
Attributes: `id, module-id, status, working-dir`

A `process-type` is a grouping of `process-sets` that are supported by the same module.

id="process-type-id"

Required: true
Default: none
Valid Values: a string

The `id` attribute uniquely identifies the `process-type` within the `ias-component`.

module-id="module-id"

Required: true
Default: none
Valid Values: a string

The `module-id` attribute must map directly map to the `module-id` attribute that supports the `process-type`.

status="state"

Required: false
Default: `enabled`
Valid Values: `enabled` or `disabled`

A `process-type` may be `enabled`, in which case OPMN parses all of its configured attributes and elements and enables requests to operate upon it, or `disabled`, in which case the `process-type` entry is completely ignored and treated as if it were not listed in `the opmn.xml file`.

working-dir="path"

Required: false
Default: *`ORACLE_HOME`*
Valid Values: A path name

The `working-dir` path name specifies the working location that is set for managed processes created by the `process-type` element. If a `process-set` also defines a `working-dir` attribute, then the process-set path name takes precedence over the `process-type` path name. The *`$ORACLE_HOME`* directory may be used.

service-failover="num"

Required: false
Default: 0
Valid Values: An integer value > 0.

A `process-type` may be configured as a `service-failover` (if `num` is not zero), which represents a process that exists num times somewhere in the Oracle Application Server cluster when it is up. The implementation is limited such that only one process of this type will run on a single instance, and so the maximum number of processes for a specific `service-failover` in the cluster can never be more than the number of participating instances in the cluster. If the value of num is greater than the number of instances participating in this `service-failover` in the cluster and the `service-failover` is active (it has been started), then each participant added to the cluster will automatically start its `service-failover` process until the total number cluster wide is num.

A `service-failover` process can run on any instance participating in the service, which means each instance must have the service configured with the same `ias-component id`, `process-type id` and `process-set id`. To target the

service itself, a request must specify both the `ias-component` and the `process-type` (it can also include the `process-set`).

A `service-failover` `process-type` can have only one `process-set`. Because the number of processes for a failover service is always 1, this `process-set` cannot specify `numprocs`, `minprocs`, or `maxprocs`.

A `service-failover` can be specified as a dependency (like any managed-process) or can specify dependencies. If specified as a dependency, the dependency check for a `service-failover` will evaluate true as soon as one process of this type is active anywhere in the cluster, regardless of the configured value for `num`.

service-weight="value"

Required: false
Default: 100
Valid Values: An integer value > 0.

The instances that run the actual `service-failover` processes are selected based upon the configured (or default) service-weight value. Instances with higher weights are selected over instances with lower weights. If a set of instances have the same weight for a service, then the configured number of instances are selected from the set to run the processes.

The `service-weight` attribute can only be specified if the `service-failover` attribute is set to a nonzero value.

<event-scripts>

Required: false
Default: none
Parents: <process-type> , <process-set>
Attributes: none

A configured `event script` is executed when a specific process related event has occurred. OPMN waits until the script completes or times out before proceeding with the next `action` for the process.

Table 6–6 shows event script arguments.

Table 6–6 Event Script Arguments

Option Name	Option Argument	Description
`-timeStamp`	`<time>`	An integer value for the current `time` on the system (in seconds).
`-instanceName`	`<instance-name>`	The `instance-name` of the managed process.
`-componentId`	`<component-id>`	The `component-id` of the managed process.
`-processType`	`<process-type-id>`	The `process-type` of the managed process.
`-processSet`	`<process-set-id>`	The `process-set` of the managed process.
`-processIndex`	`<index>`	The `process-index` of the managed process.
`-stderr`[1]	`<path>`	The path name for the `stderr` file pointer of the process.
`-stdout`[1]	`<path>`	The path name for the `stdout` file pointer of the process. Note: this argument will only be given for a pre-start script if the start is part of a process restart request.

Table 6–6 (Cont.) Event Script Arguments

Option Name	Option Argument	Description
-reason	<reason>	A string indicating the reason script was executed. The http_request indicates the process action is the result of the user HTTP request to OPMN. The non_http_request indicates the process action was initiated by OPMN itself.
-pid[2]	<process-id>	The operating system integer value given for the process-id.
-startTime[2]	<time>	An integer value for the system start time of the process (in seconds).

[1] This argument will only be given for a pre-start script if the start is part of a process restart request. The pre-start event is triggered only prior to performing a start. A restart operation may be composed of a stop operation followed by a start operation. A start operation can occur as an operation all by itself or as a sub-operation of a restart.

[2] This argument is only available with pre-stop or post-crash event scripts.

<pre-start>

Required: false
Default: none
Parents: <event-scripts>
Attributes: path

OPMN runs the pre-start script after any configured dependency checks have been performed (and passed) and before the process is actually started. The timeout for the pre-start script is the timeout value configured for starting the process itself, and any time consumed by the execution of this script counts toward the process start timeout. If the script times out, the process will not be started and any associated HTTP request will fail.

Be cautious when you execute any OPMN process requests such as start, stop or restart within an event script. These requests are serialized at the process-set level. If the script invokes a request on a process-set on which the current request (or another already queued request) is operating, then the script will hang until it times out.

path="path"

Required: true
Default: none
Valid Values: The path name to the executable script.

The path name must specify either an executable program for which OPMN has execute permission, or a script file for which OPMN has both read and executable permission. The *ORACLE_HOME* value may be used.

<pre-stop>

Required: false
Default: none
Parents: <event-scripts>
Attributes: path

OPMN runs the specified script before stopping the associated process. The timeout for the script is the value configured for stopping the process itself. Any time

consumed by the execution of the script counts toward the process stop timeout. If the script times out, any associated HTTP request will fail. However, OPMN will proceed with stopping the process.

Be cautious when you execute any OPMN process requests such as `start, stop,` or `restart`. These requests are serialized at the `process-set` level. If the script invokes a request on a `process-set` on which the current request (or another already queued request) is operating, then the script will hang until it times out.

path="path"

Required: true
Default: none
Valid Values: The path name to the executable script.

The `path` attribute must specify either an executable program for which OPMN has execute permission, or a script file for which OPMN has both read and executable permission. The *`$ORACLE_HOME`* directory may be used.

<post-crash>

Required: false
Default: none
Parents:<event-scripts>
Attributes: `path`

OPMN runs the specified script after the associated process has terminated unexpectedly. The timeout for the script is the timeout value configured for stopping the process itself. After the script has terminated OPMN schedules a replacement of the terminated process.

Be cautious when you execute any OPMN process requests such as `start`, stop or restart. These requests are serialized at the `process-set` level. If the script invokes a request on a `process-set` on which the current request (or another already queued request) is operating, then the script will hang until it times out.

path="path"

Required: true
Default: none
Valid Values: The path name to the executable script.

The `path` attribute must specify either an executable program for which OPMN has execute permission, or a script file for which OPMN has both read and executable permission. The *`$ORACLE_HOME`* directory may be used.

<start>

Required: false
Default: Refer to the values in the following paragraphs.
Parents: <process-type> , <process-set>
Attributes: `timeout, retry`

The `start` attribute contains the start parameters for a managed processes.

timeout="timeout"

Required: false
Default: `60`
Valid Values: An integer

The `timeout` attribute specifies the timeout value in seconds for the start of a managed process.

retry="num"

Required: false
Default: `0`
Valid Values: An integer

The `retry` attribute specifies the number of consecutive attempts that will be made to start the process for a single request.

<stop>

Required: false
Default: Refer to the values in the following paragraphs.
Parents: <process-type> , <process-set>
Attributes: `timeout`

The `stop` attribute specifies the stop parameters for managed processes.

timeout="timeout"

Required: false
Default: `30`
Valid Values: An integer

The `timeout` value in seconds for the stopping a managed process.

<restart>

Required: false
Default: Refer to the values in the following paragraphs.
Parents: <process-type> , <process-set>
Attributes: `timeout, retry`

The `restart` parameters for managed processes.

timeout="timeout"

Required: false
Default: `90`
Valid Values: An integer

The `timeout` value in seconds for the restart of a managed process.

retry="num"

Required: false
Default: `0`
Valid Values: An integer

The `retry` attribute is the number of consecutive attempts that will be made to restart the process for a single request.

<ping>

Required: false
Default: Refer to the values in the following paragraphs.

Parents: <process-type> , <process-set>
Attributes: `timeout, retry, interval`

The `ping` element is the ping parameters for managed processes.

timeout="timeout"

Required: false
Default: `20`
Valid Values: An integer

The `timeout` value in seconds for the ping of a managed process. Each module specifies a ping timeout.

retry="num"

Required: false
Default: `0`
Valid Values: An integer

The `retry` attribute is the number of consecutive ping failures that will be tolerated before the module declares the process unreachable and restarts it. Each module specifies ping retries.

interval="interval"

Required: false
Default: `20`
Valid Values: An integer

The `interval` attribute is the interval, in seconds, between each ping of a managed process.

<port>

Required: false
Default: none
Parents: <process-type>
Attributes: `id, range`

The `port` element provides a port management mechanism for modules to use. Each module uses the ports configured with `id`.

id="id"

Required: true
Default: none
Valid Values: A string

The `id` attribute identifies the range of ports for the `process-type`. Each module has its own list of required or optional `port` ids.

range="range"

Required: true
Default: none
Valid Values: A port range

The `port range` specifies which ports to use for the `id`.

Upon request from a module for a port number from the `id`, OPMN checks if a port in the range has been bound on the local system, and if it has not, it returns that port number back to the module. Syntax of the `port range` is a comma separated list of individual port numbers or a low-high range specification.

Examples:

Specify ports 5555, 6666, 7777, 8888, and 9999:

```
range="5555,6666,7777,8888,9999"
```

Specify ports 4000 through 4250 (inclusive):

```
range="4000-4250"
```

Specify ports 7000 through 7049, 7775, 7785, and 8050 through 8099:

```
range="7000-7049,7775,7785,8050-8099"
```

<process-set>

Required: true
Default: none
Parents: <process-type>
Attributes: `id, restart-on-death, numprocs, minprocs, maxprocs, status, working-dir, parallel-requests`

A `process-set` is the abstraction of a process within OPMN. All `module-data`, environment variables, and other configuration parameters are resolved into their final values at the `process-set` level.

id="process-set-id"

Required: true
Default: none
Valid Values: A string

The `id` attribute uniquely identifies the `process-set` within the `process-type`.

restart-on-death="boolean"

Required: false
Default: false
Valid Values: `true` or `false`

If a managed process terminates unexpectedly, that is, not stopped by a request, then OPMN will not automatically restart it. To enable automatic restarting of terminated managed processes set the attribute to `true`.

numprocs="num"

Required: true unless `minprocs` is configured; otherwise false
Default: none
Valid Values: An integer

Specifies the number of processes for OPMN to start for the `process-set`.

minprocs="min"

Required: true unless `numprocs` is configured; otherwise false
Default: none

Valid Values: An integer

Specifies the default number of processes for OPMN to start for this process set. If `minprocs` is configured, then `maxprocs` must be set with a value greater than or equal to the value for minprocs. If minprocs and `maxprocs` are configured, a specific number of processes may be given in an OPMN request for this process set. This attribute may not be specified if `numprocs` has been configured.

maxprocs="max"

Required: true if minprocs is configured; otherwise false
Default: none
Valid Values: An integer

The `maxprocs` attribute must be specified if minprocs has been configured, but cannot be specified if numprocs has been configured.

status="state"

Required: false.
Default: `enabled`
Valid Values: `enabled` or `disabled`

A `process-set` may be `enabled`, in which case OPMN parses all of its configured attributes and elements and enables requests to operate upon it, or `disabled`, in which case the `process-set` entry is complete ignored and treated as if it were not even listed in `the opmn.xml file`.

working-dir="path"

Required: false.
Default: *`$ORACLE_HOME`*
Valid Values: A path name

The `working-dir` path name specifies the working directory set for the managed processes created that belong to the `process-set`. The *`$ORACLE_HOME`* directory may be used.

parallel-requests="boolean"

Required: false
Default: false
Valid Values: `true` or `false`

OPMN serializes requests at the `process-set` level, such that only one request can execute on a given `process-set` at a time: each subsequent request must wait until the previous request completes before it can execute. This default behavior is disabled for a `process-set` when `parallel-requests` is set to `true`.

Note: When the `parallel-requests` attribute is enabled OPMN performs **no** serialization on requests for the `process-set` at all, which means conflicting requests may be issued and execute at virtually the same time, thus leaving processes in the `process-set` in unpredictable states; therefore when `parallel-requests` is set to `true` you must verify that conflicting requests are not issued at the same time for the `process-set` (this includes requests with implicit wild-cards for matching `process-sets`).

7

Configuring Oracle HTTP Server

This chapter describes Oracle HTTP Server configuration in the OPMN `opmn.xml` file.

It features the following topics:

Note:

- Only one Oracle HTTP Server can be configured for each *ORACLE_HOME*.
- Oracle Application Server 10*g* Release 3 (10.1.3) comes with Oracle HTTP Server based on Apache 1.3. This version of Oracle Application Server cannot be configured to run Oracle HTTP Server 2 based on Apache 2.
- To obtain Oracle HTTP Server 2 you must use the Oracle HTTP Server 2 standalone install. The Oracle HTTP Server 2 standalone installation is available on the companion CD. The Oracle HTTP Server 2 installation cannot be configured to run Oracle HTTP Server.

7.1 Oracle HTTP Server Process Module Configuration

The following lines load and identify the Oracle HTTP Server process module. Management of Oracle HTTP Server processes by the process module are identified by the `module id`.

```
<module path="ORACLE_HOME/opmn/lib/libopmnohs.so">
   <module-id="OHS"/>
</module>
```

7.2 Oracle HTTP Server Minimum Configuration

The following lines represent the minimum configuration for Oracle HTTP Server. Default values are assigned to all other configuration elements and attributes for Oracle HTTP Server.

```
<ias-component id="HTTP_Server">
   <process-type id="HTTP_Server" module-id="OHS">
      <process-set id="HTTP_Server" numprocs="1"/>
   </process-type>
</ias-component>
```

7.3 Oracle HTTP Server Complete Configuration

The following lines show a complete configuration for Oracle HTTP Server. It contains all possible configuration elements and attributes for Oracle HTTP Server.

```
<ias-component id="HTTP_Server" status="enabled" id-matching="false">
   <process-type id="HTTP_Server" module-id="OHS">
      <process-set id="HTTP_Server" restart-on-death="true" numprocs=1>
         <module-data>
            <category id="start-parameters">
               <data id="config-file" value="/myconfs/httpd.conf"/>
               <data id="start-mode" value="ssl-disabled"/>
               <data id="command-line" value="-D MyDefine"/>
               <data id="routing-id" value="routing id">
            </category>
            <category id="ping-parameters">
               <data id="ping-url" value="/"/>
            </category>
            <category id="restart-parameters">
               <data id="reverseping-timeout" value="345"/>
               <data id="no-reverseping-failed-ping-limit" value="3"/>
               <data id="reverseping-failed-ping-limit" value="6"/>
            </category>
         </module-data>
            <start timeout="300" retry="3"/>
            <stop timeout="300"/>
            <restart timeout="300"/>
            <ping timeout="30" interval="30"/>
      </process-set>
   </process-type>
</ias-component>
```

7.4 Oracle HTTP Server Attribute Descriptions

This section describes the attributes that are specific for Oracle HTTP Server.

The Oracle HTTP Server attributes are described with the following format:

- **Title**: This is the attribute name and value being defined. For example, **id="HTTP_Server"**.
- **Required**: This field defines whether or not the attribute is required in the component definition.
- **Default**: This defines the default value assigned to the attribute. The default value appears in the installed version of the `opmn.xml` file or is assigned internally if the attribute is not present.

- **Valid values:** If applicable, this field defines the valid values for the attribute. For example, `HTTP_Server`.
- **Path**: This field defines in which elements the attribute can appear. For example, `ias-component/process-type/process-set`

id="HTTP_Server"

Required: true
Default: none
Valid values: `HTTP_Server`
Path: `ias-component`
Path: `ias-component/process-type`
Path: `ias-component/process-type/process-set`

The `id` attribute is required and cannot be changed. The `id` must match the `targets.xml` entry or Application Server Control Console will not work.

module-id="OHS"

Required: true
Default: none
Valid values: `OHS`
Path: `ias-component/process-type`

The `module-id` attribute defines the type of process. It associates the configuration with a process module. The `OHS module-id` can be configured for managing Apache 1.3.

numprocs=1

Required: true
Default: none
Valid values: `1`
Path: `ias-component/process-type/process-set`

The number of Oracle HTTP Server Instances to start. Only valid value is 1.

The `numprocs` attribute gives the number of Oracle HTTP Server instances to start. The only valid value is 1.

id="start-parameters"

Required: false
Default: none
Path: `ias-component/process-type/process-set/module-data/category`

The `start-parameters` category contains the parameters that are relevant for the startup of Oracle HTTP Server.

id="config-file"

Required: false
Default: *`ORACLE_HOME`*`/Apache/Apache/conf/httpd.conf`
Valid values: any full path to an existing configuration file
Path:
`ias-component/process-type/process-set/module-data/category/data`

The `config-file id` is an start command option which specifies the `httpd.conf` for starting Oracle HTTP Server. The `config-file id` is not supported on Microsoft Windows

id="start-mode"

Required: false
Default: `ssl-enabled`
Valid values: `ssl-enabled/ssl-disabled`
Path: `ias-component/process-type/process-set/module-data/category/data`

This option specifies whether Oracle HTTP Server will be started with `ssl enabled`.

id="command-line"

Required: false
Default: none
Valid values: any valid command line options to Oracle HTTP Server
Path: `ias-component/process-type/process-set/module-data/category/data`

This `id` option specifies extra command lines to append to the Oracle HTTP Server command line.

id="routing-id"

Required: false
Default: none
Parents: category
Attributes: value

The `routing-id` defines the value for the routing ID module.

You can configure the `routing-id` element for Oracle HTTP Server in both the `opmn.xml` file and in the configuration for Oracle HTTP Server. However, if the `routing-id` element is configured in both locations, OPMN will not start.

value="routing id"

Required: true
Default: none
Valid Values: Any string.

The Routing ID specifies a routing relationship between OC4J and Oracle HTTP Server. In other words, an Oracle HTTP Server routes to every OC4J that it shares a routing ID with. Out of the box, the routing ID is specified as a module data under the `ias-instance` in the `opmn.xml` file. The value of the default `routing-id` is `g_rt_id`. Because of the hierarchical nature of the `opmn.xml` file, every component configured in the `opmn.xml` file inherits the configured routing ID. You can configure a separate routing ID for any Oracle HTTP Server or OC4J by configuring a routing ID at a lower level in the `opmn.xml` file. As with most entities in the `opmn.xml` file, like entities configured at a lower level override those configured at a higher level.

id="ping-parameters"

Required: false
Default: none
Path: `ias-component/process-type/process-set/module-data/`

`category`

The `ping parameters` category contains the parameters that configure how OPMN pings Oracle HTTP Server.

id="ping-url"

Required: false
Default: `/`
Valid values: the path portion of an url; for example: `http://127.0.0.1/<path>`
Path: `ias-component/process-type/process-set/module-data/category/data`

The `ping-url id` specifies the URL at which OPMN pings Oracle HTTP Server.

id="restart-parameters"

Required: false
Default: none
Path: `ias-component/process-type/process-set/module-data/category`

The `restart-parameters` category is used for defining parameters that will be used in death-detection.

id="reverseping-timeout"

Required: false
Default: 300 seconds
Valid values: Any reasonable timeout value
Path: `ias-component/process-type/process-set/module-data/category/data`

The `reverseping-timeout` value is the maximum allowable time between two notifications arriving from an Oracle HTTP Server process. As part of death-detection, the Oracle HTTP Server module performs forward pings on the Oracle HTTP Server process. In the event that forward pings start failing, the reverse pings are taken into account in death-detection and Oracle HTTP Server processes are restarted.

id="no-reverseping-failed-ping-limit"

Required: false
Default: 1
Valid values: Any reasonable value that reflects the tolerance that OPMN should have for failed forward pings when reverse pings are also failing. This tolerance is used by OPMN to determine when the process should be declared as unresponsive and replaced.
Path:
`ias-component/process-type/process-set/module-data/category/data`

This `id module data` element defines the tolerance for failed forward pings in the event that reverse pings are also not being received (within the timeout period specified by the `reverseping-timeout` data element). After the number of ping failures equals this limit, the process is deemed unresponsive and restarted by OPMN.

id="reverseping-failed-ping-limit"

Required: false
Default: 3
Valid values: Any reasonable value that reflects the tolerance that OPMN should have for failed forward pings when reverse pings are being received. This tolerance is used by OPMN to determine when the process should be declared as unresponsive and replaced.
Path: `ias-component/process-type/process-set/module-data/category/data`

This `module data` element defines the tolerance for failed forward pings when reverse pings are succeeding. After the number of ping failures equals this limit, the process is deemed unresponsive and restarted by OPMN.

7.5 Oracle HTTP Server 2

The Oracle HTTP Server 2 based on Apache 2 (OHS2) module ID is used to manage the version of Apache 2 shipped with Oracle Application Server 10*g* on the companion CD.

module-id="OHS2"

Required: true
Default: none
Valid values: `OHS2`
Path: `ias-component/process-type`

The `module-id` attribute defines the type of process. It associates the configuration with a process module.

id="mpm"

Required: false
Default: `prefork`
Valid values: `worker/prefork`
Path: `ias-component/process-type/process-set/module-data/category/data`

This option specifies what threading model `OHS2` should use. This option is only valid for `OHS2` and is only valid on Linux platforms.

Oracle does not support the perchild MPM. Microsoft Windows uses the `mpm_winnt` MPM.

7.6 Generic Apache (Linux only)

The Oracle HTTP Server process module can be configured to manage generic Apache processes. Follow these steps to configure the Oracle HTTP Server process module to manage generic Apache:

- Update the module definition to include the generic Apache `module-id`. The module definition should look similar to:

```
<module path="ORACLE_HOME/opmn/lib/libopmnohs.so">
   <module-id id="OHS" />
   <module-id id="GENERIC_APACHE" />
```

```
    </module>
```

- Alter your HTTP_Server component to manage generic Apache. A GENERIC_APACHE process has one required parameter specified as `module-data`. It is "`apache-home`" in the "`start-parameters`" `module-data` category. The "`apache-home`" specifies the installation directory of the generic Apache. The following example shows a generic Apache configuration:

```
<ias-component id="HTTP_Server">
   <process-type id="HTTP_Server" module-id="GENERIC_APACHE">
      <module-data>
         <category id="start-parameters">
            <data id="apache-home" value="/private1/apbuild/runapache_1.3.27"/>
         </category>
      </module-data>
      <process-set id="HTTP_Server" numprocs="1"/>
   </process-type>
</ias-component>
```

8

Configuring OC4J

This chapter describes OC4J configuration in the OPMN `opmn.xml` file.

It features the following topics:

- Section 8.1, "OC4J Process Module Configuration"
- Section 8.2, "OC4J Minimum Configuration"
- Section 8.3, "OC4J Complete Configuration"
- Section 8.4, "OC4J Attribute Descriptions"

8.1 OC4J Process Module Configuration

The following lines load and identify the OC4J process module. Management of OC4J processes by the process module are identified by the `module id`.

```
<module path="ORACLE_HOME/opmn/lib/libopmnoc4j.so">
   <module-id id="OC4J" />
</module>
```

8.2 OC4J Minimum Configuration

The following lines represent the minimum configuration for OC4J. Default values are assigned to all other configuration elements and attributes for OC4J.

```
<ias-component id="OC4J">
   <process-type id="home" module-id="OC4J">
      <port id="ajp" range="3301-3400" />
      <port id="rmi" range="3101-3200" />
      <port id="jms" range="3201-3300" />
      <process-set id="default-group" numprocs="1"/>
   </process-type>
</ias-component>
```

8.3 OC4J Complete Configuration

The complete configuration example in this section showcases the attributes of OC4J configuration that you can control. It contains all possible configuration elements and attributes that can be used with this component.

```
<ias-component id="OC4J" status="enabled" id-matching="false">
   <environment>
      <variable id="LD_LIBRARY_PATH" value="ORACLE_HOME/lib"
      append="true" />
   </environment>
```

```
      <process-type id="home" module-id="OC4J">
      <module-data>
         <category id="start-parameters">
            <data id="java-options" value="-DTestVar=TestVal"/>
            <data id="oc4j-options" value=" "/>
            <data id="config-file" value="/my/config/dir/server.xml"/>
            <data id="java-bin" value="/my/javalocation/jdk/bin/java"/>
            <data id="routing-id" value="MYRoutingID">
         </category>
         <category id="stop-parameters">
            <data id="java-options" value="-DTestVar=TestVal"/>
         </category>
         <category id="restart-parameters">
            <data id="reverseping-timeout" value="345"/>
            <data id="no-reverseping-failed-ping-limit" value="3"/>
            <data id="reverseping-failed-ping-limit" value="6"/>
         </category>
         <category id="urlping-parameters">
            <data id="/j2ee/servlet/Spy" value="200"/>
         </category>
         <category id="security-parameters">
            <data id="wallet-file" value="file:/private/user/ssl_cert/client_cert"/>
            <data id="wallet-password" value ="welcome1"/>
         </category>
      </module-data>
         <start timeout="300" retry="3"/>
         <stop timeout="300"/>
         <restart timeout="300"/>
         <port id="default-web-site" range="12501-12600" protocol="ajp"/>
         <port id="rmi" range="12401-12500"/>
         <port id="jms" range="12601-12700"/>
      <process-set id="default-island" restart-on-death="true" numprocs="1"/>
   </process-type>
</ias-component>
```

8.4 OC4J Attribute Descriptions

This section describes the attributes that are specific for OC4J. This section also provides attribute descriptions of the attributes.

The OC4J attributes are described with the following format:

- **Title**: This is the attribute name and value being defined. For example, **id="OC4J"**.
- **Required**: This field defines whether or not the attribute is required in the component definition.
- **Default**: This defines the default value assigned to the attribute. The default value appears in the installed version of the `opmn.xml` file or is assigned internally if the attribute is not present.
- **Valid values:** If applicable, this field defines the valid values for the attribute. For example, `OC4J`.
- **Path**: This field defines in which elements the attribute can appear. For example, `ias-component`.

id="OC4J"

Required: true
Default: none

Valid values: `OC4J`
Path: `ias-component`

The `id` name is required and cannot be changed. The `id` must match the entry in the `targets.xml` file or Application Server Control Console will not work.

environment

Required: false
Default: none
Path: `ias-component`

The environment element can be specified at multiple levels within the Oracle Application Server component. This concept is important for the OC4J module because OC4J processes can be part of other Oracle Application Server components. In those cases, the required environment may have to be specified at the `ias-component/process-type` level.

process-type

Required: true
Default: none
Path: `ias-component`

For OC4J processes, the `process-type` element is administratively equivalent to an OC4J instance.

module-id="OC4J"

Required: true
Default: none
Path: `ias-component/process-type`

The `module-id` associates the process with a module. For OC4J processes, this `id` has to match the `module-id` specified in the process module configuration for the OC4J module.

port

Required: true
Default: none
Path: `ias-component/process-type`

The OC4J processes will not be started up unless port elements for `ajp` and `rmi` ports are configured. An OC4J process can be pinged at the `ajp` ports; therefore one of the ports has to be configured. If both type of ports are configured, then the `ajp` port is used for pinging. In addition to the `ajp` and `rmi` ports, you can also configure other port types that will be passed in to the OC4J process at the command line during process startup.

id="ajp"

Required: true
Default: none
Path: `ias-component/process-type/port`

A `port` element defining `ajp` `port` values is required.

id="rmi"

Required: true
Default: none
Path: `ias-component/process-type/port`

A `port` element defining `rmi` `port` values is required.

id="jms"

Required: true
Default: none
Path: `ias-component/process-type/port`

A `port` element defining `jms` `port` values is required.

range

Required: true
Default: none
Valid values: range of ports, individual port numbers or 0
Path: `ias-component/process-type/port`

This attribute is used to specify valid port ranges, comma separated list of ports or a mix of both. For port selection by the operating system to select ports, specify 0 and the OC4J process will use a port provided by the system.

<process-set>

Required: true
Default: none
Path: `ias-component/process-type`

For OC4J processes, the `process-set` element is administratively equivalent to an OC4J group.

id="start-parameters"

Required: false
Default: none
Path: `ias-component/process-type/process-set/module-data/category`

The `start-parameters id` is a category that collects all of the parameters that are relevant for the startup of an OC4J process.

id="java-options"

Required: false
Default: none
Valid values: any options acceptable to Java
Path: `ias-component/process-type/process-set/module-data/category/data`

OC4J requires that some `java-options` be passed to start and stop commands. These options are derived internally by OPMN, are not part of the opmn.xml configuration, and cannot be overridden. Additional `java-options` may be specified using this `module data` element.

id="oc4j-options"

Required: false
Default: none
Valid values: any options acceptable to the OC4J executable
Path: `ias-component/process-type/process-set/module-data/category/data`

OC4J processes require options to be passed in as part of the start or stop commands to function correctly. These options cannot be overridden. In addition to these options, other options can be passed in through this `module data` element. There is no default value for this data element.

id="config-file"

Required: false
Default: *ORACLE_HOME*`/j2ee/<process-type id>/config/server.xml`
Valid values: any full path to an existing configuration file
Path: `ias-component/process-type/process-set/module-data/category/data`

The configuration file is an OC4J option in the start command. The default value for this data element is built from the *ORACLE_HOME* variable and OC4J instance name (`process-type id`).

id="java-bin"

Required: false
Default: *ORACLE_HOME*`/jdk/bin/java`
Valid values: Full path to `java.exe`
Path: `ias-component/process-type/process-set/module-data/category/data`

The default value is the complete path to Java that is available in the installation. You can specify alternate paths to the Java executable. However, a valid version of Java will have to be used for the process to start up and work correctly.

id="routing-id"

Required: false
Default: none
Parents: category
Attributes: value

The `routing-id` defines the value for the routing ID module.

value="routing id"

Required: true
Default: none
Valid Values: Any string.

The Routing ID specifies a routing relationship between OC4J and Oracle HTTP Server. In other words, an Oracle HTTP Server routes to every OC4J that it shares a routing ID with. Out of the box, the routing ID is specified as a module data under the `ias-instance` in the `opmn.xml` file. The value of the default `routing-id` is `g_rt_id`. Because of the hierarchical nature of the `opmn.xml` file, every component configured in the `opmn.xml` file inherits the configured routing ID. You can configure a separate routing ID for any Oracle HTTP Server or OC4J by configuring a routing ID

at a lower level in the `opmn.xml` file. As with most entities in the `opmn.xml` file, those that are configured at a lower level override those configured at a higher level.

id="stop-parameters"

Required: false
Default: none
Path: `ias-component/process-type/process-set/module-data/category`

The `stop-parameters id` is a category that includes all the parameters that are relevant for stopping an OC4J process.

id="restart-parameters"

Required: false
Default: none
Path: `ias-component/process-type/process-set/module-data/category`

The `restart-parameters` category is used for defining parameters that will be used in death-detection.

id="reverseping-timeout"

Required: false
Default: `300 seconds`
Valid values: Any reasonable timeout value
Path: `ias-component/process-type/process-set/module-data/category/data`

The `reverseping-timeout` value is the maximum allowable time between two notifications arriving from an OC4J process. As part of death-detection, the OC4J module performs forward pings on the process also. In the event that forward pings start failing, the reverse pings are taken into account in death-detection and restart.

id="no-reverseping-failed-ping-limit"

Required: false
Default: `1`
Valid values: Any value that reflects the tolerance that OPMN should have for failed forward pings when reverse pings are also failing. This tolerance is used by OPMN to determine when the process should be declared as unresponsive and replaced.
Path: `ias-component/process-type/process-set/module-data/category/data`

This `module data` element defines the tolerance for failed forward pings in the event that reverse pings are also not being received (within the timeout period specified by reverseping-timeout data element). After the number of ping failures equals this limit, the process is deemed unresponsive and restarted by OPMN.

id="reverseping-failed-ping-limit"

Required: false
Default: `3`
Valid values: Any reasonable value that reflects the tolerance that OPMN should have for failed forward pings when reverse pings are being received. This tolerance is used by OPMN to determine when the process should be declared as unresponsive and replaced.

Path: `ias-component/process-type/process-set/module-data/category/data`

This `module data` element defines the tolerance for failed forward pings when reverse pings are succeeding. After the number of ping failures equals this limit, the process is deemed unresponsive and restarted by OPMN.

id="urlping-parameters"

Required: false
Default: Not Applicable
Valid values: Not Applicable
Path: `ias-component/process-type/process-set/module-data/category`

The `"urlping-parameters" id` enables you to specify URLs for ping operations as part of OC4J process ping operations. The data under this category consists of the URL and a valid HTTP return code. `AJP13` protocol is used to directly connect to the OC4J process and the HTTP return code is validated against the configured code. If there are multiple URLs configured, failure in pinging any one of them will be considered a ping failure and the process will be restarted after the ping failures limit is exceeded.

id="/j2ee/servlet/Spy"

Required: false
Default: Not Applicable
Valid values: Any valid URL on the OC4J process.
Path: `ias-component/process-type/process-set/module-data/category/data`

This is the URL in the OC4J process that will be pinged.

value="200"

Required: false
Default: Not Applicable
Valid values: Any valid HTTP return code.
Path: `ias-component/process-type/process-set/module-data/category/data`

The following is the HTTP code that results from ping operations to the configured URL.

```
<category id="security-parameters">
   <data id="wallet-file"value="file:/private/user/ssl_cert/client_cert"/>
   <data id="wallet-password"value="welcome1"/>
</category>
```

id="security-parameters"

Required: false
Default: Not Applicable
Valid values: Not Applicable
Path: `ias-component/process-type/process-set/module-data/category/`

The OC4J process module can perform pings over SSL. The `"security-parameters" id` is a category that enables you to specify the wallet file and password for such communication.

id="wallet-file"

Required: false
Default: Not Applicable
Valid values: Not Applicable
Path: `ias-component/process-type/process-set/module-data/category/data`

The `data id` whose value is the path to the wallet file (not including the filename).

value="file:/private/user/ssl_cert/client_cert"

Required: false
Default: Not applicable
Valid values: Path to a wallet file (not including the filename).
Path: `ias-component/process-type/process-set/module-data/category/data`

The path to the wallet file (not including the filename). The data in the wallet file is used in SSL authentication during pinging.

id="wallet-password"

Required: false
Default: Not applicable
Valid values: Not applicable
Path: `ias-component/process-type/process-set/module-data/category/data`

The `data id` that specifies the wallet password.

value ="welcome1"

Required: false
Default: Not applicable
Valid values: The valid wallet password.
Path: `ias-component/process-type/process-set/module-data/category/data`

This value specifies the password for the wallet.

9

Configuring Oracle Application Server Port Tunnel

This chapter describes Oracle Application Server Port Tunnel (OracleAS Port Tunnel) configuration in the OPMN `opmn.xml` file.

It features the following topics:

- Section 9.1, "OracleAS Port Tunnel Process Module Configuration"
- Section 9.2, "OracleAS Port Tunnel Minimum Configuration"
- Section 9.3, "OracleAS Port Tunnel Complete Configuration"
- Section 9.4, "OracleAS Port Tunnel Attribute Descriptions"

See Also: *Oracle HTTP Server Administrator's Guide*

9.1 OracleAS Port Tunnel Process Module Configuration

The following lines load and identify the OracleAS Port Tunnel process module. Management of OracleAS Port Tunnel processes by the process module are identified by the `module id`.

```
<module path="ORACLE_HOME/opmn/lib/libopmniaspt.so">
   <module-id id="IASPT" />
</module>
```

9.2 OracleAS Port Tunnel Minimum Configuration

The following lines represent the minimum configuration for OracleAS Port Tunnel. Default values are assigned to all other configuration elements and attributes for OracleAS Port Tunnel.

```
<ias-component id="IASPT">
   <process-type id="IASPT" module-id="IASPT">
      <process-set id="IASPT" numprocs="1"/>
   </process-type>
</ias-component>
```

9.3 OracleAS Port Tunnel Complete Configuration

The following example represents the complete configuration for OracleAS Port Tunnel. It contains all possible configuration elements and attributes that can be used with OracleAS Port Tunnel.

```
<module path="ORACLE_HOME/opmn/lib/libopmniaspt.so">
   <module-id id="IASPT" />
</module>
<ias-component id="IASPT" status="enabled" id-matching="false">
   <process-type id="IASPT" module-id="IASPT">
      <port id="ajp" range="6701-6703"/>
   <process-set id="IASPT" restart-on-death="true" id="ajp"/>
   </process-type>
</ias-component>
```

9.4 OracleAS Port Tunnel Attribute Descriptions

This section describes the attributes that are specific for OracleAS Port Tunnel.

The OracleAS Port Tunnel attributes are described with the following format:

- **Title**: This is the attribute name and value being defined. For example, **id="IASPT"**.
- **Required**: This field defines whether or not the attribute is required in the component definition.
- **Default**: This defines the default value assigned to the attribute. The default value appears in the installed version of the `opmn.xml` file or is assigned internally if the attribute is not present.
- **Valid values:** If applicable, this field defines the valid values for the attribute. For example, `IASPT`.
- **Path**: This field defines in which elements the attribute can appear. For example, `ias-component`.

id="IASPT"

Required: true
Default: none
Valid values: `IASPT`
Path: `ias-component`
Path: `ias-component/process-type`
Path: `ias-component/process-set`

The `id` name is required and cannot be changed. The `id` name must match the entry in the `targets.xml` file.

module-id="IASPT"

Required: true
Default: none
Path: `ias-component/process-type`

The `module-id` name defines the type of process and associates the configuration with a process module.

id="ajp"

Required: false
Default: none
Valid values: `ajp`
Path: `ias-component/process-type/port`

The `id` value should be used together with `range` in `port` property to specify the `ajp` ports to be used by the OracleAS Port Tunnel server. If the `id` is specified, the `port` number configured in the `iaspt.conf` file is overwritten.

range="6701-6703"

Required: false
Default: none
Valid values: Any single port or a range of ports
Path: `ias-component/process-type/port`

The `range` value should be used together with `ajp` in `port` property to specify the `ajp` ports to be used by OracleAS Port Tunnel servers.

numprocs="3"

Required: true
Default: none
Valid values: Any number
Path: `ias-component/process-type/process-set`

This attribute tells how many OracleAS Port Tunnel server processes to be started. The `ajp` range should be configured in the `port` property if the value is 1. If the value is greater than 1, `ajp` range has to be configured to specify enough ports for each OracleAS Port Tunnel server process. Typically, the value is 1 port for each process.

10

Configuring Custom Process

This chapter describes custom process configuration in the OPMN `opmn.xml` file.

It features the following topics:

- Section 10.1, "Custom Process Module Configuration"
- Section 10.2, "Custom Process Minimum Configuration"
- Section 10.3, "Custom Process Complete Configuration"
- Section 10.4, "Custom Process Attribute Descriptions"

10.1 Custom Process Module Configuration

The following lines load and identify the custom process module. Management of custom processes by the process module are identified by the `module id`.

```
<module path="ORACLE_HOME/opmn/lib/libopmncustom.so">
   <module-id id="CUSTOM" />
</module>
```

10.2 Custom Process Minimum Configuration

The following lines represent the minimum configuration for a custom process. Default values are assigned to all other configuration elements and attributes for the custom process.

```
<ias-component id="Custom">
   <process-type id="Custom" module-id="CUSTOM">
   <process-set id="Custom" numprocs="1">
      <module-data>
         <category id="start-parameters">
            <data id="start-executable" value="Your start executable here" />
         </category>
      </module-data>
   </process-set>
   </process-type>
</ias-component>
```

10.3 Custom Process Complete Configuration

Example 10–1 show a complete configuration for a custom process. It contains all possible configuration elements and attributes for a custom process.

A custom process can be part of any other Oracle Application Server component. In such cases, the `process-type` element in Example 10–1 must be part of the component configuration.

10.3.1 Ping

The custom module provides the framework for pinging a custom process in one of two ways:

- HTTP ping
- script ping

The type of ping can be configured by specifying the appropriate data in the `ping-parameters` category. The sample Oracle Application Server configuration example Example 10–1 shows a custom process using HTTP ping. Example 10–2 is an example of script ping that you can substitute into the component configuration.

Example 10–1 Custom Process Complete Configuration

```
<ias-component id="Custom" status="enabled" id-matching="false">
   <environment>
      <variable id="TEST_ENV_VARIABLE" value="/your/test/value"
      append="false"/>
   </environment>
   <process-type id="Custom" module-id="CUSTOM">
   <process-set id="Custom" restart-on-death="true" numprocs="1">
      <module-data>
         <category id="start-parameters">
            <data id="start-executable" value="Your start executable here" />
            <data id="start-args" value="Your start args here" />
         </category>
         <category id="stop-parameters">
            <data id="stop-executable" value="Your stop executable here" />
            <data id="stop-args" value="Your stop args here" />
         </category>
         <category id="restart-parameters">
            <data id="restart-executable" value="Your restart executable here"/>
            <data id="restart-args" value="Your restart args here" />
         </category>
         <category id="ping-parameters">
            <data id="ping-type" value="http" />
            <data id="ping-url" value="/your/ping/url" />
            <data id="ping-host" value="abc.company.com" />
            <data id="ping-port" value="7777" />
            <data id="ping-limit" value="3" />
            <data id="ping-timeout" value="300" />
         </category>
         <category id="ready-parameters">
            <data id="use-ping-for-ready" value="false" />
         </category>
      </module-data>
   </process-set>
   </process-type>
</ias-component>
```

Pinging with a script can be configured as shown in Example 10–2.

Example 10–2 Ping Type Script

```
<category id="ping-parameters">
```

```
   <data id="ping-type" value="script" />
   <data id="script-executable" value="Ping executable here" />
   <data id="script-args" value="Ping arguments here " />
</category>
```

You can use pinging (when available) for determining the readiness of a process. This indicates that OPMN needs confirmation that a managed process has started successfully after creation. Processes can inform OPMN of their ready status in various ways. The custom module enables these processes to communicate readiness through pinging. If you configure pinging for a custom process, you can also use this mechanism to determine if the process is ready. You can choose not to configure any mechanism for determining readiness in which case the custom module just assumes that the process started successfully.

Note: The ready ping, if configured, is created soon after the process is created. If the process takes a while to initialize and respond to pings, then using pinging for determining readiness is not appropriate. This is because if the process does not respond to the "ready ping", OPMN will determine that the process did not start correctly and stop it.

10.4 Custom Process Attribute Descriptions

This section describes the attributes that are specific for a custom process. This section also provides attribute descriptions of the attributes.

The custom process attributes are described with the following format:

- **Title**: This is the attribute name and value being defined. For example, **id="Custom"**.
- **Required**: This field defines whether or not the attribute is required in the component definition.
- **Default**: This defines the default value assigned to the attribute. The default value appears in the installed version of the `opmn.xml` file or is assigned internally if the attribute is not present.
- **Valid values:** If applicable, this field defines the valid values for the attribute. For example, `custom`.
- **Path**: This field defines in which elements the attribute can appear. For example, `ias-component`.

id="Custom"

Required: true
Default: none
Valid values: Any `id` of your choice
Path: `ias-component`
Path: `ias-component/process-type`
Path: `ias-component/process-type/process-set`

This `id` is required and can be any name you choose. The `id` cannot be a duplicate of existing names.

module-id="CUSTOM"

Required: true

Default: none
Valid values: The same as the `module-id` specified in Section 10.1, "Custom Process Module Configuration".
Path: `ias-component/process-type`

The `module-id` associates the process with a module. For Custom processes, this `id` has to match the `module-id` specified in Process Module Configuration for the Custom module.

id="start-parameters"

Required: true
Default: none
Path: `ias-component/process-type/process-set/module-data/category`

The `start-parameters` category contains child elements specifying the start executable and start arguments.

id="start-executable"

Required: true
Default: none
Valid values: a valid executable to run
Path:
`ias-component/process-type/process-set/module-data/category/data`

This data element specifies the name of the executable to be started.

id="start-args"

Required: false
Default: none
Valid values: Valid arguments to the executable specified by start-executable data element.
Path:
`ias-component/process-type/process-set/module-data/category/data`

The value of this data element should be a string containing all the arguments for the start executable. Multiple data elements with this `id` should not be specified.

id="stop-parameters"

Required: false
Default: none
Path: `ias-component/process-type/process-set/module-data/category`

The `stop-parameters` category contains child elements specifying the stop executable and stop arguments. If this category is not configured, OPMN stops the process with the kill command.

id="stop-executable"

Required: false
Default: none

Path:
`ias-component/process-type/process-set/module-data/category/data`

This data element specifies the name of the executable to be used for stopping the process.

id="stop-args"

Required: false
Default: none
Path:
`ias-component/process-type/process-set/module-data/category/data`

The value of this data element should be a string containing all the arguments to the stop executable. Multiple data elements with this `id` should not be specified.

id="restart-parameters"

Required: false
Default: none
Path: `ias-component/process-type/process-set/module-data/category`

The `restart-parameters` category contains child elements specifying the restart executable and restart arguments. This category needs to be configured if the process has an explicit restart command. In the absence of a restart command, a stop followed by the start command will be executed whenever the process needs to be restarted.

When restart data is specified, OPMN assumes that the process ID of the process remains the same after a restart. If there is no explicit restart command available for the process, a stop followed by a start is issued. In this scenario, a process ID change is acceptable.

id="restart-executable"

Required: false
Default: none
Valid values: A valid restart executable name
Path:
`ias-component/process-type/process-set/module-data/category/data`

This data element specifies the name of the executable to be used for restarting the process.

id="restart-args"

Required: false
Default: none
Valid values: valid arguments to the restart executable
Path:
`ias-component/process-type/process-set/module-data/category/data`

The value of this data element should be a string containing all the arguments to the restart executable. Multiple data elements with this `id` should not be specified.

id="ping-parameters"

Required: false
Default: none
Path: `ias-component/process-type/process-set/module-data/category`

Custom processes that are pinged through the HTTP protocol must specify this category. This `module data` category consists of all the data required to perform such a ping.

id="ping-type"

Required: false
Default: none
Valid values: `http`, `script`
Path:
`ias-component/process-type/process-set/module-data/category/data`

Custom processes that wish to be pinged have to specify this `module data`.

See Also:

- Example 10–1, "Custom Process Complete Configuration"
- Example 10–2, "Ping Type Script"

id="ping-url"

Required: false
Default: /
Valid values: Any valid URL
Path:
`ias-component/process-type/process-set/module-data/category/data`

This data element is used to specify the URL at which the process will be pinged. The listed parameters are used for HTTP pings.

id="ping-host"

Required: false
Default: none
Valid values: A valid hostname to which a custom process is bound.
Path:
`ias-component/process-type/process-set/module-data/category/data`

This data element is used to specify the host name to which a custom process is bound. If this data is not specified, pinging will not be performed. If an invalid hostname is specified, the `process-set` will be `disabled`.

id="ping-port"

Required: false
Default: none
Valid values: A valid port at which a custom process is listening for HTTP requests
Path:
`ias-component/process-type/process-set/module-data/category/`

`data`

The port at which a custom process is listening. If this data is not specified, pinging will not be performed. If an invalid port is specified, the `process-set` will be `disabled`.

id="ping-limit"

Required: false
Default: `3`
Valid values: Any reasonable value that reflects the tolerance that OPMN should have for failed pings. This tolerance is used by OPMN to determine when the process should be declared unresponsive and restarted.
Path: `ias-component/process-type/process-set/module-data/category/data`

This `module data` element defines the tolerance for failed pings. After the number of ping failures reaches this limit, the process is deemed unresponsive and restarted by OPMN.

id="ping-timeout"

Required: false
Default: `300 seconds`
Valid values: Any reasonable timeout value
Path: `ias-component/process-type/process-set/module-data/category/data`

The timeout value specified with this data element is used as the maximum time OPMN will wait for a ping response. If a response is not obtained within the timeout period, the ping attempt will be considered a failure.

id="script-executable"

Required: false
Default: none
Valid values: A valid script executable
Path: `ias-component/process-type/process-set/module-data/category/data`

This data element specifies the name of the executable to be used for pinging the process. An exit value of 0 from this executable is considered success. All other values indicate a ping failure.

id="script-args"

Required: false
Default: none
Valid values: valid arguments to the ping executable
Path: `ias-component/process-type/process-set/module-data/category/data`

The value of this data element should be a string containing all the arguments to the ping executable. Multiple data elements with this id should not be specified.

id="ready-parameters"

Required: false

Default: none
Path: `ias-component/process-type/process-set/module-data/category`

The `module data` category to indicate if pinging should be used to determine that a custom process started successfully.

id="use-ping-for-ready"

Required: false
Default: false
Valid values: `true` or `false`
Path: `ias-component/process-type/process-set/module-data/category/data`

The value of this data element determines if pinging should be used to determine if a process is available.

A

OPMN Troubleshooting

This chapter describes some troubleshooting tips for OPMN. It features the following topics:

- Section A.1, "Problems and Solutions"
- Section A.2, "Diagnosing OPMN Problems"
- Section A.3, "Need More Help?"

A.1 Problems and Solutions

This section describes some of the common problems encountered when using OPMN. It features the following topics:

- Section A.1.1, "Oracle Application Server Process Does Not Start"
- Section A.1.2, "Determining if Oracle Application Server Processes are Dying or Unresponsive"
- Section A.1.3, "opmnctl Command Execution Times Out"
- Section A.1.4, "Oracle Application Server Component Automatically Restarted by OPMN"
- Section A.1.5, "Unexpected opmnctl start Behavior"
- Section A.1.6, "Disabled Element in the opmn.xml File"
- Section A.1.7, "Unable to Start OC4J"
- Section A.1.8, "Unable to Stop Component"
- Section A.1.9, "globalInitNLS Error"
- Section A.1.10, "Start Remote Hosts of a Cluster Independently"
- Section A.1.11, "OPMN Start Up Consumes CPU Processing Capability"
- Section A.1.12, "Error Messages During Start-up of OPMN"
- Section A.1.13, "Disable, or Reconfigure, Firewall When Creating Topology Using Multi-Cast Address Configuration"

A.1.1 Oracle Application Server Process Does Not Start

Problem

Unable to start an Oracle Application Server process using OPMN.

Solution

Try the following if you are unable to start an Oracle Application Server process using OPMN:

- Verify and if necessary, correct, the command input. Confirm the spelling and choice of option for the command you are entering.

  Note: Do not use command line scripts or utilities from previous versions of Oracle9*i* Application Server or Oracle Application Server for starting OPMN or Oracle Application Server components.

- Review the standard out output log for the Oracle Application Server process. Output from the process console is located in the *ORACLE_HOME*/opmn/logs directory. For example, the standard output log for Oracle HTTP Server may be HTTP_Server~1.
- Verify the dependency requirements for the Oracle Application Server process you are attempting to start.
- Verify the element values for the Oracle Application Server component in the opmn.xml file. Use the opmnctl validate command to verify configuration of opmn.xml file. You may have mis-configured the opmn.xml for the Oracle Application Server component you are attempting to start.

A.1.2 Determining if Oracle Application Server Processes are Dying or Unresponsive

Problem

Your Oracle Application Server processes are dying or unreachable.

Solution

If your Oracle Application Server processes are dying or unreachable:

- Review the Oracle Application Server component specific output in the *ORACLE_HOME*/opmn/logs.

  Look at the *ORACLE_HOME*/opmn/logs/opmn.log for Oracle Application Server processes. Look for process crashed or process unreachable messages. OPMN automatically restarts Oracle Application Server processes that die or become unresponsive.

  See Also: Section A.2.1, "OPMN log Files"

- Create event scripts for any pre-stop or post-crash events. The event scripts could be used to create a specific log file or send you an email about a failure.

  See Also: Section A.2.4, "Troubleshooting with Event Scripts"

A.1.3 opmnctl Command Execution Times Out

Problem

The time it takes to execute an opmnctl command is dependent on the type of Oracle Application Server process and available computer hardware. Because of this the time it takes to execute an opmnctl command may not be readily apparent.

The default start time out for OC4J is approximately five minutes. If an OC4J process does not start-up after an `opmnctl` command, OPMN will wait approximately an hour before timing out and aborting the request.

Solution

To verify successful execution of the `opmnctl` command, try the following:

1. Increase the `start` element `timeout` attribute for the component that is not starting. Set the timeout in the `opmn.xml` file at a level that will allow OPMN to wait for process to come up. This functionality is available with the `startproc` command which will start all the relevant processes configured in `opmn.xml`.
2. Check the `start` element in the `opmn.xml` file and change the `retry` attribute to a higher increment of time.
3. Look at the *`ORACLE_HOME`*`/opmn/logs/` for the Oracle Application Server process that is not starting.
4. Review the component-specific log file for the Oracle Application Server component that is not starting. For example, *`ORACLE_HOME`*`/opmn/logs/OC4J~home~default_group~1`.

See Also: Chapter 6, "opmn.xml Common Configuration"

A.1.4 Oracle Application Server Component Automatically Restarted by OPMN

Problem

An Oracle Application Server component is automatically restarted by OPMN.

Solution

If an Oracle Application Server component is automatically restarted by OPMN, try the following:

- Review the message for the Oracle Application Server component in the *`ORACLE_HOME`*`/opmn/logs/opmn.log` file.
- Verify that the ping timeout for the Oracle Application Server component is sufficient. An Oracle Application Server component that receives a lot of activity may require an increase in the length of time for the timeout. Increase the ping timeout element in the Oracle Application Server component `opmn.xml` file.

A.1.5 Unexpected opmnctl start Behavior

Problem

Occasionally, there is unexpected behavior when you use the `opmnctl start` command to start OPMN; either only OPMN is started or OPMN makes a best effort to start Oracle Application Server OPMN-managed processes. Typically, this unexpected behavior is due to turning-off or rebooting your computer without first shutting down OPMN. When you restart your computer, all OPMN-managed processes are started.

Solution

Oracle recommends that you shutdown OPMN before shutting down your computer. Use the `opmnctl stopall` command to stop OPMN and OPMN-managed processes.

On the Microsoft Windows operating system, you can use the Windows services control panel to stop OPMN and OPMN-managed processes.

Note: OPMN keeps a record on disk of the expected status of the processes it manages. If a computer goes down while OPMN is running, upon restart OPMN will use the information cached on disk and make a best effort attempt to automatically restart all processes that were running at the time the system went down. This may catch some users off guard who start only OPMN and notice that processes managed by OPMN have also been started even though an explicit request to start those processes has not been issued. You can suppress this automatic process recovery by removing all files located in the *ORACLE_HOME*/opmn/logs/states directory before attempting to start OPMN.

The states directory and its contents should not be modified by the user if OPMN or any process managed by OPMN is running. Oracle recommends not modifying the /states directory.

A.1.6 Disabled Element in the opmn.xml File

Problem

Unable to start an Oracle Application Server process.

Solution

If you are unable to start an Oracle Application Server process, check if an element in the Oracle Application Server opmn.xml file is disabled. If an element in the opmn.xml file is disabled OPMN will generate an output message of "Missing" or "Disabled".

A.1.7 Unable to Start OC4J

Problem

If you have multiple Oracle Application Server installations on one host and you start them at the same time (for example, to start a cluster), OPMN may become unresponsive. You may receive an error message such as:

```
"failed to restart a managed process after the maximum retry limit"
```

This may occur when two Oracle homes on the same host use the same port ranges for RMI, JMS, and AJP ports. An OC4J instance in one Oracle home is trying to use the same port as an OC4J instance in a different Oracle home.

Port allocation for all OC4J instances within Oracle Application Server is controlled by OPMN; there can be overlapping port ranges within a single opmn.xml file. However, when two OPMN processes on a host start at the same time, there is no coordination between them on port usage.

Solution

To coordinate port usage, assign unique port ranges to each Oracle home. The OPMN process in one Oracle home and the OPMN in a different Oracle home will not attempt to use the same port numbers when assigning OC4J ports, and will not attempt to bind to the same port.

It is also recommended that you increase the maximum number of retries for starting OC4J instances. If you have identical port ranges in two Oracle homes and increase the number of times OPMN attempts to restart a process, OPMN will eventually select a port that works. This technique ultimately does not eliminate the problem, because there is the possibility that OPMN will not find a port that works in the number of port connection attempts that you have specified in the opmn.xml file.

Note: *Oracle Application Server Administrator's Guide*

A.1.8 Unable to Stop Component

Problem

If you are unable to stop Oracle Application Server components or OPMN-managed processes using the `opmnctl stop` or `opmnctl stopall` commands, the component or process was most likely not started using OPMN. The component or process might have been started using a startup script or utility.

Solution

Oracle Application Server components and OPMN-managed processes should never be started or stopped manually. Do not use command line scripts or utilities from previous versions of Oracle Application Server for starting and stopping Oracle Application Server components.

Use the Application Server Control Console and the `opmnctl` command line utility to start or stop Oracle Application Server components and OPMN-managed processes.

See Also: Chapter 5, "Using OPMN"

A.1.9 globalInitNLS Error

Problem

You may receive a `globalInitNLS` error when executing the `opmnctl` command. The following error message is displayed:

```
"globalInitNLS: NLS boot file not found or invalid -- default linked-in boot block
used XML parser init: error 201."
```

Solution

This error occurs when the `ORA_NLS33` environmental variable is set. This environmental variable should not be set.

A.1.10 Start Remote Hosts of a Cluster Independently

Problem

Starting a cluster of remote hosts using Application Server Control Console will result in an unknown status. This occurs because ONS is bound to the local host IP address and it is not reachable from remote hosts.

Solution

Oracle recommends starting each member of the cluster independently to effectively monitor and obtain the status from remote hosts. Additionally, make sure ONS is not bound to local host IP address.

A.1.11 OPMN Start Up Consumes CPU Processing Capability

Problem

On some computers, when OPMN starts up, it consumes large amounts of CPU processing capability. This can vary from approximately 50% to 60% of your computer's CPU processing capabilities. In affected computers, the OPMN CPU processing consumption will continue until OPMN is shutdown.

Solution

The following are some possible causes for the excessive CPU processing consumption:

- the installation environment used multibyte text character sets such as Japanese.
- the multi-cast address for all ONS servers is mis-configured in the `opmn.xml` file.

  ONS uses this address to discover all other instances in the cluster

A.1.12 Error Messages During Start-up of OPMN

Problem

When trying to start OPMN using the `opmnctl start` or `opmnctl startall` commands you receive the following error messages:

```
pingwait exits with 1220384
```

or

```
pingwait exits with 1220396
```

These error messages are generated when there are syntax errors in the *ORACLE_HOME*`/opmn/conf/opmn.xml` that need to be corrected.

Solution

If you encounter these error messages do the following:

- run the following command (with the complete directory path to the `opmn.xml` file):

  ```
  prompt > opmnctl validate opmn.xml
  ```

- remove all empty tags from the `opmn.xml` file.

A.1.13 Disable, or Reconfigure, Firewall When Creating Topology Using Multi-Cast Address Configuration

Problem

When setting up a network of Oracle Application Server instances to form a topology using the multi-cast address configuration for all ONS servers, some of the instances are not recognized by OPMN.

Solution

If you are planning to network multiple Oracle Application Server instances to form a topology, by using the multi-cast address configuration for all ONS servers in the `opmn.xml` file, you must disable, or reconfigure, the firewall before initiating networking with other Oracle Application Server instances.

If the firewall is not disabled, or re configured, the multi-cast information for setting up the network may not get through and the topology will not be setup correctly. All of the OPMN ports must be allowed to accept incoming notifications.

A.2 Diagnosing OPMN Problems

There are several methods for troubleshooting any problems you may have using OPMN:

- Section A.2.1, "OPMN log Files"
- Section A.2.2, "opmnctl debug"
- Section A.2.3, "Oracle Enterprise Manager 10g Application Server Control Console"
- Section A.2.4, "Troubleshooting with Event Scripts"
- Section A.2.5, "opmn.xml Environment Variables"

A.2.1 OPMN log Files

The OPMN log files enable you to troubleshoot difficulties you might have in execution and use of OPMN and OPMN-managed processes. OPMN and OPMN-managed processes generate log files during processing. You can review the following generated log files to verify successful or unsuccessful execution of an OPMN command:

- *ORACLE_HOME*`/opmn/logs/opmn.out`: contains the standard output (`stdout`) and standard error (`stderr`) logs of OPMN. Also referred to as the OPMN "console log". After a certain point in OPMN initialization, nothing else will be written to this file. Only a small set of messages will ever appear in this file; therefore, this file may not be present if you conduct a search through the log file directories.
- Process control log files (*ORACLE_HOME*`/opmn/logs/`): contain the standard output and standard error of OPMN managed processes. OPMN creates a log file for each component and assigns a unique concatenation of the Oracle Application Server component with a number. For example, the standard output log for OC4J may be `OC4J~home~default_group~1`. When a process terminates and is replaced by a new process, console log output from the previous process is preserved and the replacement process appends to the end of the console log file. The process specific console logs are the first and best resource for investigating problems related to starting and stopping components.
- *ORACLE_HOME*`/opmn/logs/opmn.log`: tracks command execution and operation progress. It contains messages useful for monitoring the operations of the OPMN server. Output written to the `opmn.log` file contains the exit status of a child OPMN process. A status code of 4 indicates a normal reload of OPMN. All other status codes indicate an abnormal termination of the child OPMN process. The `opmn.log` file is configured using the `<log>` attribute in the `opmn.xml` file. Refer to Chapter 6, "opmn.xml Common Configuration" for more information.

- *ORACLE_HOME*/opmn/logs/opmn.dbg: contains OPMN debug log messages (English only) for ONS and PM. Review the error codes and messages that are shown in the opmn.dbg file. The PM portion of OPMN generates and outputs the error messages in this file. The opmn.dbg file tracks command execution and operation progress. The level of detail that gets logged in the opmn.dbg can be modified by configuration of the <debug> element in the opmn.xml file.

  Refer to Chapter 6, "opmn.xml Common Configuration" for examples of debug levels.

  Use the opmn.dbg file to debug the ONS portion of OPMN or for early OPMN errors. The ONS portion of OPMN is initialized before PM. Therefore, errors that occur early in OPMN initialization will show up in the opmn.dbg file.

  Enable usage of the opmn.dbg file only after conferring with Oracle Support. The opmn.dbg file is used by Oracle Support to debug and diagnose OPMN issues. Messages that are contained in the opmn.dbg file are typically not readily comprehensible to the user.

A.2.1.1 opmn.log and opmn.dbg File Rotation

OPMN enables you to rotate the opmn.log and opmn.dbg files based on parameters of file size, specific time, or both, as a basis for file rotation. You can enable rotation by configuring the rotation-size and rotation-hour attributes of the <log> and <debug> tags in the opmn.xml file. When either the log file grows to a specified size or the specified time of the day is reached, or a combination of both parameters, the OPMN logging mechanism will close the file, rename the file with a unique time stamp suffix, and then create a new opmn.log or opmn.dbg file.

The OPMN console log file (opmn.out) is not rotated; this file is typically very small in size. Once OPMN surpasses an point of initialization, output is no longer generated to the console output file; therefore, only a relatively small set of messages will appear in this file.

A.2.1.2 Process Console log File Rotation

At process startup, before handing off an existing console log file to a managed process, OPMN checks the size against a configured limit (rotation-size attribute of the <log> tag). If the file size exceeds the limit, OPMN will rename the existing file to include a time stamp, and then create a new file for the managed process. If the rotation-size attribute is not configured, OPMN will not be able rotate the process console log file.

A.2.2 opmnctl debug

Use the opmnctl debug command to verify the status of an Oracle Application Server process and whether any actions are pending. This command generates output that can be used in conjunction with contact to your local Oracle support to diagnose your OPMN problem.

The syntax for the opmnctl debug command is:

```
opmnctl [<scope>] debug [comp=pm|ons] [interval=<secs> count=<num>]
```

where @scope is the optional scope for the request.

Output is generated following execution of the opmnctl debug command. Oracle recommends that you contact Oracle support to use the generated output to assist in diagnosis of your problem.

The attributes (<attr>) name for this command are either comp, interval, or count. The value for comp can be either ons or pm, representing ONS and PM, respectively. If comp is not specified, then both ons and pm debug information is reported. For example, the following command outputs debug information for ONS.

```
prompt > opmnctl debug comp=ons
```

You can specify the interval in seconds and number of requests sent to OPMN to assist in the debugging process. The values of <interval> and <count> must always be specified together. Values for them should be integers greater than 0. For example, the following command, outputs debug information at an interval of 5 seconds 3 times.

```
prompt > opmnctl debug comp=pm interval=5 count=3
```

Contact your local Oracle support to assist you in using the opmnctl debug command to diagnose your OPMN problem.

A.2.3 Oracle Enterprise Manager 10*g* Application Server Control Console

Application Server Control Console provides a graphical interface that enables diagnosis of Oracle Application Server components in your network and enterprise. Application Server Control Console features a log page. The log page enables you to view all of the Oracle Application Server log files in one place and trace problems across multiple log files. Application Server Control Console uses an API that contacts OPMN.

You can use Application Server Control Console to enable or disable Oracle Application Server components: You can disable components so they do not start when you start an Oracle Application Server instance.

See Also: *Oracle Application Server Administrator's Guide*

A.2.4 Troubleshooting with Event Scripts

You can create your own event scripts that record Oracle Application Server process event activities. You can create a script that records events prior to the start or stop of Oracle Application Server processes, as well as an unscheduled system crash.

Refer to the <event-scripts> element description in Chapter 6, "opmn.xml Common Configuration".

Example A–1 shows a pre-start event script.

Example A–1 Pre-start Event Script

```
#!/bin/sh
echo
echo =------------------------------------------------------------
echo =---------- PRE-START EVENT SCRIPT ----------
echo =------------------------------------------------------------

timeStamp="N/A"
instanceName="N/A"
componentId="N/A"
processType="N/A"
processSet="N/A"
processIndex="N/A"
stderrPath="N/A"  # not available w/pre-start unless part of restart
stdoutPath="N/A"  # not available w/pre-start unless part of restart
reason="N/A"
```

```
pid="N/A"         # only available with pre-stop, post-crash
startTime="N/A"   # only available with pre-stop, post-crash

while [ $# -gt 0 ]; do
     case $1 in
        -timeStamp)    timeStamp=$2; shift;;
        -instanceName) instanceName=$2; shift;;
        -componentId)  componentId=$2; shift;;
        -processType)  processType=$2; shift;;
        -processSet)   processSet=$2; shift;;
        -processIndex) processIndex=$2; shift;;
        -stderr)       stderrPath=$2; shift;;
        -stdout)       stdoutPath=$2; shift;;
        -reason)       reason=$2; shift;;
        -pid)          pid=$2; shift;;
        -startTime)    startTime=$2; shift;;
        *) echo "Option Not Recognized: [$1]"; shift;;
        esac
        shift
done

echo timeStamp=$timeStamp
echo instanceName=$instanceName
echo componentId=$componentId
echo processType=$processType
echo processSet=$processSet
echo processIndex=$processIndex
echo stderr=$stderrPath
echo stdout=$stdoutPath
echo reason=$reason
echo pid=$pid
echo startTime=$startTime
```

Note: The pre-start event script example, Example A–1, will not work for the Microsoft Windows operating system; however, you can create a script, with a `.bat` suffix, with similar functionality.

Use the full path to the `.bat` file when adding the necessary configuration information to the `opmn.xml` file,.

A.2.5 opmn.xml Environment Variables

The environment variable used to launch OPMN server is not inherited by the Oracle Application Server process started by OPMN server. OPMN sets the environment variables at the `ias-instance` level, with the values extracted either from the `ias-instance` configuration or from the OPMN run time environment.

See Also: Chapter 6, "opmn.xml Common Configuration"

A.3 Need More Help?

You can find more solutions on Oracle *MetaLink* (`http://metalink.oracle.com`). If you do not find a solution for your problem, log a service request.

See Also:

- *Oracle Application Server Release Notes*, available on the Oracle Technology Network:

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

Index

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W