Contents

Preface ........................................................................................................................................................................... ix
  Audience ........................................................................................................................................................................... ix
  Documentation Accessibility ............................................................................................................................................... ix
  Related Documentation ................................................................................................................................................ ix
  Conventions ................................................................................................................................................................. ix

1 What’s New in OPMN?
  1.1 Integration with Oracle Instance ........................................................................................................................ 1-1
  1.2 Support with Oracle WebLogic Server .................................................................................................................. 1-1

2 Oracle Process Manager and Notification Overview
  2.1 What is OPMN? ....................................................................................................................................................... 2-1
  2.2 How OPMN Works .................................................................................................................................................... 2-2
    2.2.1 Oracle Notification Server ............................................................................................................................... 2-3
    2.2.2 Oracle Process Manager ................................................................................................................................. 2-3
    2.2.3 PM Modules ...................................................................................................................................................... 2-4
  2.3 What System Components Does OPMN Manage? ............................................................................................... 2-4
    2.3.1 Oracle Enterprise Manager Fusion Middleware Control Console ................................................................. 2-5

3 The opmn.xml File
  3.1 opmn.xml .............................................................................................................................................................. 3-1
  3.2 Automatic Restart .................................................................................................................................................. 3-2
  3.3 OPMN Log Files ...................................................................................................................................................... 3-2
  3.4 Security ................................................................................................................................................................. 3-3
  3.5 IPv6 Support ........................................................................................................................................................ 3-4

4 opmnctl Commands
  4.1 opmnctl ................................................................................................................................................................. 4-1
    4.1.1 opmnctl Syntax .................................................................................................................................................. 4-2
  4.2 opmnctl Command Quick Reference ........................................................................................................................ 4-2
  4.3 opmnctl Detailed Command Description ............................................................................................................... 4-3
    4.3.1 Command Definitions ........................................................................................................................................ 4-3
    4.3.1.1 Scope ........................................................................................................................................................... 4-3
    4.3.1.2 Attributes ...................................................................................................................................................... 4-3
5 Using OPMN

5.1 Starting OPMN .......................................................... 5-1
5.2 Starting and Stopping All System Components .......... 5-1
5.3 Starting and Stopping a System Component ................................................................. 5-1
5.4 Starting and Stopping all System Components of the Same Type ................................ 5-2

6 opmn.xml Common Configuration
6.1 Example of opmn.xml Elements and Attributes ............................................................. 6-1
6.1.1 Global Definitions and Syntax .................................................................................. 6-1
6.2 opmn.xml Element and Attribute Descriptions ............................................................. 6-3

A Configuring Custom Process
A.1 Custom Process Module Configuration ....................................................................... A-1
A.2 Custom Process Minimum Configuration ..................................................................... A-1
A.3 Custom Process Complete Configuration ...................................................................... A-1
A.3.1 Ping ......................................................................................................................... A-2
A.4 Custom Process Attribute Descriptions ..................................................................... A-3

B OPMN Troubleshooting
B.1 Problems and Solutions .............................................................................................. B-1
B.1.1 System Process Does Not Start ................................................................................ B-2
B.1.2 Determining if System Processes are Dying or Unresponsive .................................. B-2
B.1.3 opmnctl Command Execution Times Out ................................................................ B-3
B.1.4 System Component Automatically Restarted by OPMN ......................................... B-3
B.1.5 Unexpected opmnctl start Behavior ........................................................................ B-4
B.1.6 Disabled Element in the opmn.xml File ................................................................. B-4
B.1.7 Unable to Start OHS ............................................................................................... B-5
B.1.8 Unable to Stop Component ..................................................................................... B-5
B.1.9 globalInitNLS Error ............................................................................................... B-5
B.1.10 OPMN Start Up Consumes CPU Processing Capability ....................................... B-6
B.1.11 Error Messages During Start-up of OPMN............................................................. B-6
B.1.12 Increasing Size of opmn.log File .......................................................................... B-6
B.1.13 Cleanup of Oracle Instance or Component If It is in a Bad State ......................... B-7
B.1.14 Path Arguments Containing Quotation Marks on Windows Operating Systems... B-7
B.2 Diagnosing OPMN Problems ...................................................................................... B-8
B.2.1 OPMN log Files ...................................................................................................... B-8
B.2.1.1 opmn.log File Rotation ...................................................................................... B-8
B.2.1.2 Process Console log File Rotation ...................................................................... B-9
B.2.2 opmnctl debug ........................................................................................................ B-9
B.2.3 Oracle Enterprise Manager Fusion Middleware Control Console ......................... B-9
B.2.4 Troubleshooting with Event Scripts ...................................................................... B-9
B.2.5 opmn.xml Environment Variables ......................................................................... B-11
B.3 Need More Help? ......................................................................................................... B-11

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

This preface contains these topics:

- Audience
- Documentation Accessibility
- Related Documentation
- Conventions

Audience

The Oracle Process Manager and Notification Server Administrator’s Guide is intended for administrators of Oracle Fusion Middleware.

Documentation Accessibility

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

Access to Oracle Support

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

Related Documentation

For more information, see these Oracle resources:

- Oracle Fusion Middleware Documentation Library
- Oracle Fusion Middleware Platform-Specific Documentation

Conventions

The following text conventions are used in this document:
<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>boldface</strong></td>
<td>Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.</td>
</tr>
<tr>
<td><strong>italic</strong></td>
<td>Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.</td>
</tr>
<tr>
<td><strong>monospace</strong></td>
<td>Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.</td>
</tr>
</tbody>
</table>
What’s New in OPMN?

This chapter describes the new features of Oracle Process Manager and Notification Server (OPMN) available in Oracle Fusion Middleware Release 11g (11.1.1.1.3).

This chapter includes the following topics:

- Integration with Oracle Instance
- Support with Oracle WebLogic Server

1.1 Integration with Oracle Instance

Each $\textsc{ORACLE\_HOME}$ can support multiple instances, each with its own configuration and diagnostic directory tree ($\textsc{ORACLE\_INSTANCE}$). OPMN now operates from within an $\textsc{ORACLE\_INSTANCE}$.

There are two basic subdirectories within the $\textsc{ORACLE\_INSTANCE}$ base directory: $\textsc{ORACLE\_INSTANCE}/\text{config}$ and $\textsc{ORACLE\_INSTANCE}/\text{diagnostics}/\text{logs}$. Within each of these two directories each component has a specific directory tree with the format: component type/component name where component type is either the OPMN type value configured for the ias-component or the process-type id for the component, and component name is the id value configured for the ias-component.

For example, the default Oracle HTTP Server (OHS) install log configuration directory would therefore be:

$\textsc{ORACLE\_INSTANCE}/\text{config}/\text{OHS}/\text{ohs1}$ (UNIX)

$\textsc{ORACLE\_INSTANCE}\textbackslash\text{config}\textbackslash\text{OHS}\textbackslash\text{ohs1}$ (Microsoft Windows)

The OPMN component type is always OPMN and component name is always opmn.

1.2 Support with Oracle WebLogic Server

There is OPMN support on the admin server, which runs as a remote WebLogic Server, and perhaps something should be added here about that. See the document scoped notes for an overview and details about OPMN and WebLogic.

In most Oracle Fusion Middleware installations an Oracle instance is associated with an Oracle WebLogic Server domain for management of system components using the Oracle Enterprise Manager Fusion Middleware Control Console (Fusion Middleware Control Console). In these deployments, OPMN is integrated with the WebLogic domain to provide remote management of system components.

Unlike prior releases of Oracle Fusion Middleware, OPMN not only manages system components in the local Oracle instance but it is also integrated with the WebLogic...
domain to provide remote system component management using either the Fusion Middleware Control Console and or the WLST command line.

**Note:** For more information about Oracle Fusion Middleware management concepts refer to the *Oracle Fusion Middleware Administrator’s Guide.*
Oracle Process Manager and Notification Overview

This chapter provides an overview of Oracle’s Process Manager and Notification (OPMN) server for Oracle Fusion Middleware. It includes the following topics:

- What is OPMN?
- How OPMN Works
- What System Components Does OPMN Manage?

2.1 What is OPMN?

Oracle Process Manager and Notification (OPMN) is installed and configured with the following installation types:

- Web Tier Installation: OPMN manages Oracle HTTP Server and Oracle Web Cache
- Oracle Portal, Forms, Reports, and Discoverer Installation: OPMN manages Oracle Portal, Oracle Forms Services, Oracle Reports, Oracle Business Intelligence Discoverer, and Oracle EM Agent and Service Manager
- Oracle Identity Management Installation: OPMN manages Oracle Internet Directory and Oracle Virtual Directory

OPMN features the following functionality:

- Provides a command-line interface for process control and monitoring for single or multiple system components.
- Provides an integrated way to manage system components.
- Enables management of system subcomponents and sub-subcomponents.
- Channels all events from different system component instances to all system components that can utilize them.
- Solves interdependency issues between system 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 system process statistics and tasks.
- Provides automatic restart of system processes when they become unresponsive, terminate unexpectedly, or become unreachable as determined by ping and notification operations.
- Provides automatic death detection of system processes.
How OPMN Works

- Enables you to control an Oracle instance through modifications to the `opmn.xml` file.

An Oracle instance is a directory that contains a set of system components managed by a common OPMN.

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

On Microsoft Windows, OPMN is created as a service for each Oracle instance.

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

Use the Fusion Middleware Control Console and the `opmnctl` command line utility to start or stop system components.

**Note:** Refer to Chapter 4 for more information about the `opmnctl` command.

### 2.2 How OPMN Works

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

The core of OPMN consists of the following three components:

- **Oracle Notification Server**
- **Oracle Process Manager**
- **PM Modules**

Figure 2–1 shows the architecture of the core of OPMN.
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 Fusion Middleware. It operates according to a publish-subscribe model: a system 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.

Unlike previous versions of Oracle Fusion Middleware, ONS is not configured to discover other OPMN servers. OPMN does not need to connect to other OPMNs for normal operation because all administration is centralized in the WebLogic domain. The WebLogic domain establishes remote client connections to each OPMN server as needed.

2.2.2 Oracle Process Manager

Oracle Process Manager (PM) is the centralized process management mechanism used to manage system 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 system 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 system 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 is restarted by the first OPMN server process.

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

The Fusion Middleware Control Console also uses PM to manage processes.

The PM uses the Oracle Notification Server 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

### 2.2.3 PM Modules

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

The PM Modules:

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

See Also: Chapter 6 for the common configuration of the `opmn.xml` file.

### 2.3 What System Components Does OPMN Manage?

OPMN manages all system components including the Oracle EM Agent and Service Manager.

You can also configure OPMN to manage other processes (including Oracle and other third-party products) using the Custom PM Module. See Appendix A for more information about configuring a custom process.

---

**Note:** If you are installing Oracle HTTP Server on your certified Windows operating system, you should verify that the `msvcr90.dll` and `linkinfo.dll` library files exist on your system; these are required by Oracle HTTP Server.

For more information, see also “Missing Libraries Might Cause HTTPD to Exit Without Notice” in Oracle HTTP Server Release Notes.

If you are unable to the required libraries on your system, contact Microsoft support for further assistance.

OPMN manages the following Oracle Fusion Middleware components:

- OHS
What System Components Does OPMN Manage?

- Oracle Web Cache
- Oracle Internet Directory
- Oracle Virtual Directory
- Oracle Forms
- Oracle Reports
- Oracle Business Intelligence Discoverer

2.3.1 Oracle Enterprise Manager Fusion Middleware Control Console

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

See Also: Oracle Fusion Middleware Administrator's Guide for more information about Oracle Fusion Middleware management.
This chapter provides an overview of the opmn.xml file for Oracle Fusion Middleware. It features the following topics:

- opmn.xml
- Automatic Restart
- OPMN Log Files
- Security
- IPv6 Support

### 3.1 opmn.xml

The ORACLE_INSTANCE/config/OPMN/opmn/opmn.xml file is the main configuration file for OPMN. The opmn.xml file contains information for PM and system component specific configuration. The opmn.xml file shows you which system components OPMN is managing on your system.

The opmn.xml file is divided into three main sections: log files, notification-server, and process-manager.

- **log files**: The log files configuration by default fully enables the standard log but disables the debug log.

- **notification server**: The notification-server configuration includes the OPMN listener ports as well as secure connection definition. Optional `<topology>` opmn.xml file element can be manually added for OPMN discovery.

- **process manager**: The process-manager configuration includes all information required for each managed component.

The opmn.xml file does not contain component-specific element names. Component specific management code is located in the PM modules which are 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 system component or just part of a component.

```xml
<ias-component>
  <process-type>
    <process-set>
```
<ias-component>: This entry represents the system 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 system component. <process-set> is an optional configuration element.

The opmn.xml file contains system component entries arranged in the hierarchical structure shown in the following Example 3–1.

**Example 3–1 Element Entries in opmn.xml File**

```
<iias-component id="ohs1">
  <process-type id="OHS">
    <process-set id="OHS">....
```

### 3.2 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 system process death
- Periodic ping requests to system processes
- Periodic status notification from system processes (reverse-ping)

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

OPMN automatically restarts system components that terminate unexpectedly. OPMN also restarts processes that are unresponsive according to the result of notification and ping operations.

**See Also:** Chapter 6 for more information about the common configuration of the opmn.xml file.

### 3.3 OPMN Log Files

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

**See Also:**

- *Oracle Fusion Middleware Administrator’s Guide* for additional diagnostic utilities
- *Section B.2.1* for more information about OPMN log files
3.4 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/.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). You may encounter the following:

- You attempt 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 is not able to provide the correct key to the OPMN server.

- You attempt to run an OPMN client from the wrong Oracle instance. (It is possible to have multiple Oracle instances set up on the same system.) If the other Oracle instances have OPMN configured to use the same local port then the system process request from the wrong Oracle instance reads the wrong `.formfactor` file.

- You have 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 runs, binds to the new port, and overwrites 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.

- Oracle Fusion Middleware and the Oracle Database both use ONS. When these two products are installed onto the same host, an ONS port conflict may arise if the port values configured for ONS are the same for both the Oracle Database and the Oracle Fusion Middleware component.

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

Because the Oracle Database is installed in a different location than that of Oracle Fusion Middleware, the Database ONS does not have access to the `.formfactor` file that was created when OPMN started up with the Oracle Fusion Middleware. 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. 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; preventing client access to the ONS server unless they possess the correct formfactor string.
To avoid having the Oracle Database listener contact the system OPMN server, change the default local and remote port values for the ONS server that was installed with the Oracle Database.

3.5 IPv6 Support

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 × 10^38) addresses.

As shown in Figure 3–1, for output, such as debug or log records, each IPv4 identifier is 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 is displayed as eight 16 bit fields for the address (each a four digit hexadecimal format) and a single 16 bit field for the port (the five digit decimal format).

**Figure 3–1  IPv4 and IPv6 Host Identifier**
This chapter provides an overview of opmnctl commands for Oracle Fusion Middleware components managed by OPMN. It features the following topics:

- **opmnctl**
- **opmnctl Command Quick Reference**
- **opmnctl Detailed Command Description**

### 4.1 opmnctl

opmnctl is the supported tool for starting and stopping all components in an Oracle instance, with the exception of the Fusion Middleware Control Console. opmnctl provides a centralized way to control and monitor system components from the command line.

If OPMN is configured to discover other Oracle instances, you can use opmnctl to execute control and monitoring commands across multiple Oracle instances simultaneously.

The location of the opmnctl script determines which opmnctl commands you can use.

The opmnctl command exists in two distinct directory location paths:

- **ORACLE_HOME/opmn/bin/opmnctl**: The opmnctl command `ORACLE_HOME/opmn/bin/opmnctl` location can only be used to create an Oracle instance or a component for an Oracle instance on the local system. opmnctl commands generated from this location cannot be used to manage system processes.

- **ORACLE_INSTANCE/bin/opmnctl**: The other opmnctl command which is located in the `ORACLE_INSTANCE/bin/` directory location provides a per Oracle instance instantiation of opmnctl. The opmnctl command in this location must be used for managing processes for this Oracle instance and can also be used for creating components for the Oracle instance.

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

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

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

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

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>verbose</td>
<td>Prints detailed execution message, if available.</td>
</tr>
<tr>
<td>command</td>
<td>Specifies an <code>opmnctl</code> command. Refer to Example 4–1 for a list of commands.</td>
</tr>
<tr>
<td>options</td>
<td>Specifies options for the command. Refer to Section 4.3.5.1.1 for a list of command options.</td>
</tr>
</tbody>
</table>

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**

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

| verbose | print detailed execution message if available |

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

<table>
<thead>
<tr>
<th>scope</th>
<th>command</th>
<th>options</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>startproc</td>
<td>[&lt;attr&gt;=&lt;val&gt; ..] - Start opmn managed processes</td>
</tr>
<tr>
<td>stopall</td>
<td>stopproc</td>
<td>[&lt;attr&gt;=&lt;val&gt; ..] - Stop opmn managed processes</td>
</tr>
<tr>
<td>shutdown</td>
<td>startup</td>
<td>[&lt;attr&gt;=&lt;val&gt; ..] - Shutdown opmn &amp; all managed processes</td>
</tr>
<tr>
<td>[scope]</td>
<td>startup</td>
<td>[options] - Get managed process status</td>
</tr>
<tr>
<td>[scope]</td>
<td>metric</td>
<td>[options] - Get DMS metrics for managed processes</td>
</tr>
<tr>
<td>[scope]</td>
<td>dmsdump</td>
<td>[dmsargs] - Get DMS metrics for opmn</td>
</tr>
<tr>
<td>[scope]</td>
<td>debug</td>
<td>[options] - Display opmn server debug information</td>
</tr>
<tr>
<td>[scope]</td>
<td>set</td>
<td>[options] - Set opmn log parameters</td>
</tr>
<tr>
<td>[scope]</td>
<td>launch</td>
<td>[options] - Query opmn log parameters</td>
</tr>
<tr>
<td>[scope]</td>
<td>phantom</td>
<td>[options] - Launch a configured target process</td>
</tr>
<tr>
<td>[scope]</td>
<td>ping</td>
<td>[options] - Register phantom processes</td>
</tr>
<tr>
<td>[scope]</td>
<td>validate</td>
<td>[file] - Validate the given opmn xml file</td>
</tr>
<tr>
<td>[scope]</td>
<td>help</td>
<td>[file] - Print brief usage description</td>
</tr>
<tr>
<td>[scope]</td>
<td>usage</td>
<td>[command] - Print detailed usage description</td>
</tr>
<tr>
<td>[scope]</td>
<td>createinstance</td>
<td>- Create an Oracle Instance</td>
</tr>
</tbody>
</table>

Note: Oracle recommends starting OPMN as the user that has installed Oracle Fusion Middleware.
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:

- Command Definitions
- Server Control Commands
- Process Control Commands
- Provisioning Commands
- Status Commands
- 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:

- Scope
- Attributes
- Verbose

4.3.1.1 Scope

The scope option specifies which Oracle instance the opmnctl command applies to. Unlike prior releases of Oracle Fusion Middleware, the force option in this release is not used and kept here for backward compatibility.

4.3.1.2 Attributes

syntax: <attribute>=<value>

The opmnctl attributes enable you to apply process control operations to specific system components.

For example, the following command starts all system processes configured for Oracle Web Cache:

opmnctl startproc ias-component=webcache1

Refer to Chapter 5 for additional opmnctl command examples.

Table 4–2 lists the attribute names and values that can be used with the opmnctl command:
The target attributes of `ias-instance`, `ias-component`, `process-type`, and `process-set` may be specified in a hierarchical manner, with a single `ias-component` within that `ias-instance`, a single `process-type` within the `ias-component`, and a single `process-set` within the `process-type`. Any of these attributes may be omitted, and treated as a wildcard (process-set is omitted, for example, then all process-sets under the specified `process-type` matches the request).

The target attributes may also be specified in a list of a single hierarchical type (a list of `ias-components`, for example).

### Table 4–2 opmnctl Attribute Names and Values

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Attribute Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ias-instance</code></td>
<td>Value should be the same as the value for the id attribute for the <code>&lt;ias-instance&gt;</code> element in the <code>opmn.xml</code> file.</td>
</tr>
<tr>
<td><code>ias-component</code></td>
<td>Value should be the same as the value for the id attribute for the <code>&lt;ias-component&gt;</code> element in the <code>opmn.xml</code> file.</td>
</tr>
<tr>
<td><code>process-type</code></td>
<td>Value should be the same as the value for the id attribute for the <code>&lt;process-type&gt;</code> element in the <code>opmn.xml</code> file.</td>
</tr>
<tr>
<td><code>process-set</code></td>
<td>Value should be the same as the value for the id attribute for the <code>&lt;process-set&gt;</code> element in the <code>opmn.xml</code> file.</td>
</tr>
<tr>
<td><code>mode</code></td>
<td>Value can either be <code>sync</code> or <code>async</code>. The default value is &quot;sync, meaning that this request operates synchronously, and waits for the operation to complete before returning.&quot; async indicates that the request returns immediately, while OPMN continues to perform the request until the operation finishes.</td>
</tr>
<tr>
<td><code>timeout</code></td>
<td>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 <code>startproc</code> operations. The request does continue for <code>restartproc</code> and <code>stopproc</code> operations. The timeout attribute overrides the configured timeout value for the request to take for each <code>ias-instance</code>, <code>ias-component</code>, <code>process-type</code>, and <code>process-set</code> value.</td>
</tr>
<tr>
<td><code>uniqueid</code></td>
<td>This value is assigned by OPMN after starting up. You can use this value when you execute the <code>opmnctl restartproc</code> and <code>opmnctl stopproc</code> commands.</td>
</tr>
<tr>
<td><code>sequential</code></td>
<td>If the value of the sequential attribute is <code>true</code>, each process or application targeted by the request is acted upon sequentially (one at a time). The request order of affected managed-process dependencies is honored.</td>
</tr>
<tr>
<td><code>report</code></td>
<td>If the value of the report attribute is <code>true</code>, OPMN reports the results of each part of the request as it completes (as each process starts, for example). The default behavior is for OPMN to wait until the entire request has completed before sending all of the results at once.</td>
</tr>
</tbody>
</table>

**Note:** If Oracle Internet Directory is managed by OPMN and an Oracle Internet Directory dependency (not a managed-process dependency for Oracle Internet Directory) is encountered as part of the request, and the request is expected to start both Oracle Internet Directory processes (such as an `opmnctl startall` request), the Oracle Internet Directory process will not start unless it has been configured in the `opmn.xml` file previously. Otherwise, the request times out with an Oracle Internet Directory dependency failure.
**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=ohs1
```

**Example 4–2 opmnctl verbose output**

```
opmnctl startproc: starting opmn managed processes...

HTTP/1.1 200 OK
Content-Length: 661
Content-Type: text/html
Response: 1 of 1 processes started.

<?xml version='1.0' encoding='WINDOWS-1252'?>
<response>
<opmn id="stapk08:6704" http-status="200" http-response="1 of 1 processes started.">
  <ias-instance id="instance1">
    <ias-component id="ohs1">
      <process-type id="OHS">
        <process-set id="OHS">
          <process id="1792746301" pid="5106" status="Alive" index="1"
            log="/scratch/lbottlem/product/11.1.1/as1/instances/instance1/diagnostics/logs/OHS/ohs1/console~OHS~1.log"
            operation="request" result="success">
            <msg code="0" text=""></msg>
          </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.

- Server Control Commands on Microsoft Windows
- `opmnctl start`
- `opmnctl startall`
- `opmnctl stopall`
- `opmnctl shutdown`
- `opmnctl reload`
Output is not generated for the successful execution of an `opmnctl` server control command. Refer to Appendix B 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 Microsoft Windows Services you start or stop all system components on the local Oracle instance.

Use the Fusion Middleware Control Console and the `opmnctl` command line utility to start or stop system components.

#### 4.3.2.2 `opmnctl start`  
**Syntax:** `opmnctl start`

Use this command to start the OPMN server for a local Oracle instance without starting system processes.

---

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

---

**See Also:** Chapter 5 for OPMN command-line examples

#### 4.3.2.3 `opmnctl startall`

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

Use this command to start OPMN as well as the system processes for a local Oracle 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 start.

Enter the following command for additional detailed information:

`opmnctl usage startall`

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 system processes for the local Oracle 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 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 system components gracefully. Processes which will not stop gracefully is forcefully shutdown. After stopping all managed processes, the OPMN daemon shuts down itself.

The `opmnctl stopall` command should only be used when it is necessary to stop the OPMN daemon quickly in case of an emergency. 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 system components without stopping the OPMN daemon, consider using the `opmnctl stopproc` command without any arguments.

To restart the OPMN daemon without restarting any system components, 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 system processes.

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

Enter one of the following commands to obtain additional information:

```
opmnctl usage stopall
```

or

```
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 system processes for the local Oracle instance.

The `opmnctl shutdown` command quickly shuts down the OPMN daemon and system components for the local Oracle instance.

The `opmnctl shutdown` command is similar to the `opmnctl stopall` command but waits less time before initiating a forceful termination of system components. After all of the system components are stopped, the OPMN daemon shuts down 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 system components without stopping the OPMN daemon, consider using the `opmnctl stopproc` command without any arguments.

To restart the OPMN daemon without restarting any system components, 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 system component processes.

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

Enter one of the following commands to obtain additional information:

```
opmnctl usage stopall
```

or

```
opmnctl usage shutdown
```

### 4.3.2.6 opmnctl reload

**Syntax:** `opmnctl reload`

Use this command to trigger the OPMN to reread its configuration files. This command restarts the OPMN server without restarting any system processes managed by OPMN. The OPMN server for the Oracle 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:

```
opmnctl usage reload
```

**See Also:** Section 4.3.1 for OPMN command definitions

### 4.3.3 Process Control Commands

The `opmnctl` process control commands enable you to start, stop, or restart single or multiple system components. You can control a system 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:

- `opmnctl startproc`, `opmnctl restartproc` and `opmnctl stopproc`
- **Progressive Request Reports**
- **Sequential Requests**

Output is not generated for the successful execution of an `opmnctl` process control command. Refer to Appendix B if you receive any error messages during `opmnctl` command execution.
4.3.3.1 opmnctl startproc, opmnctl restartproc and opmnctl stopproc

Syntax:

```
opmnctl startproc [<attr>=<value>...]
opmnctl restartproc [<attr>=<value>...]
opmnctl stopproc [<attr>=<value>...]
```

Use these commands to start, restart, or stop system processes. The OPMN server for the Oracle instance must be up and running.

You can use attributes for these commands. If no attribute is supplied when executing `startproc`, `stopproc`, and `restartproc`, all system processes except components with `id-matching="true"` in the opmn.xml file is started.

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 system component 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 system component in the `uid` column of the generated output:

  ```
opmnctl status -l
  ```

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

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

The following command restarts Oracle Web Cache at the `<process-type>` level:

```
opmnctl restartproc ias-component=webcache1 process-type=WebCache
```

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

```
opmnctl stopproc ias-component=ohs1
```
Enter one of the following commands to obtain additional information:

```
opmnctl usage startproc
```

or

```
opmnctl usage restartproc
```

or

```
opmnctl usage stopproc
```

See Also:
- Section 4.3.1 for OPMN command definitions
- Chapter 5 for OPMN command-line examples

### 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 attempts to start 4 processes, OPMN reports back to the user the result of each process start attempt as soon as it completes.

### 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 runs 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
```

**Figure 4–1 Processes in Parallel**

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

For example the following command:
% opmnctl startproc sequential=true

starts all of the managed system processes sequentially.

*Figure 4–2  Managed Process Sequential Request #1*

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

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

*Figure 4–3  Managed Process Sequential Request #2*

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

*Figure 4–4  Managed Process Sequential Request #3*

### 4.3.4 Provisioning Commands

The `opmnctl` provisioning commands are used for provisioning system components within an Oracle instance. Use these commands to:

- Create and delete an Oracle instance or system component.
- Register and unregister an Oracle instance with a WebLogic domain for management using the Fusion Middleware Control Console.
- Update an Oracle instance or system component registration with the WebLogic domain.

---

**Note:** The `opmnctl` provisioning commands must use the following naming conventions:

1. Name must begin with a letter and contain only alphanumerics, hyphen(-) and underscore(_) characters.
2. Name must be no more than 30 characters long.
3. Name must be at least 4 characters long.

This section includes the following topics:
- General Syntax
- Alternative Syntax for Provisioning Commands
- Common Arguments
- Commands

### 4.3.4.1 General Syntax

`opmnctl` provisioning commands are invoked using the following syntax:

```
opmnctl  command  [-argName1  value1]  [-argName2  value2]  ...
```

The following provisioning commands are supported:
- createinstance
- createcomponent
- deleteinstance
- deletecomponent
- registerinstance
- unregisterinstance
- updateinstanceregistration
- updatecomponentregistration

### 4.3.4.2 Alternative Syntax for Provisioning Commands

Provisioning commands can also be invoked with the syntax used by other `opmnctl` commands:

```
opmnctl  command  [argName1=value1]  [argName2=value2]  ...
```

This alternate syntax follows the Attributes style (refer to Section 4.3.1.2) and accepts argument names presented in lower case. The alternate syntax is provided to allow greater uniformity between `opmnctl` commands.

Any `opmnctl` provisioning commands can be entered using either syntax.

For example, the following commands are equivalent:

```
opmnctl createcomponent -componentType OHS -componentName ohs1
opmnctl createcomponent componentType=OHS componentName=ohs1
```
Both syntax styles can be combined within the same command.

4.3.4.3 Common Arguments
The following arguments generally apply to all provisioning commands, with the exception with component specific arguments that apply only to component-related commands:

- **Adminserver Arguments**
- **Logging Arguments**
- **Oracle Instance Arguments**
- **Component Specific Arguments**

4.3.4.3.1 Adminserver Arguments
administrative server (Adminserver) arguments denote how opmnctl should contact the Adminserver of the Oracle WebLogic Server domain.

The Adminserver values are stored in the Oracle instance as defaults for subsequent commands. opmnctl does not use or require the Adminserver values for a non-registered Oracle instance. See Section 4.3.4.4.1 about the createinstance command for creating a non-registered instance.

The Adminserver arguments are:

- **-adminHost**: the Weblogic Adminserver host. The default value is the last successful command for the Oracle instance or local host.
- **-adminPort**: the Weblogic Adminserver port. The default value is the last successful command for the Oracle instance; or 7001.
- **-adminProtocol**: Weblogic Adminserver protocol. The default value is the last successful command for the Oracle instance; or t3.
- **-adminUsername**: the Weblogic Adminserver login user name. The default value is the last successful command for the Oracle instance; otherwise you are prompted to type in their login user name.
- **-adminPasswordFile**: the location of the file containing the administrative user (admin user) password. The default value is the last successful command for the Oracle instance; otherwise you are prompted to type in your admin user password.

4.3.4.3.2 Logging Arguments
By default, opmnctl logs the detailed provisioning errors and exceptions in the ORACLE_INSTANCE/diagnostics/logs/OPMN/opmn/provision.log file. For certain errors during execution of the createinstance command, specifically when the provision.log file has not been created yet, the detailed exception is indicated on the console. Log file locations can be specified using the -logFile argument.

- **-logFile**: the location of provisioning log file.
- **-logLevel**: the logging level of provisioning log. For example, INFO or FINER.

4.3.4.3.3 Oracle Instance Arguments
The -oracleInstance and -instanceName arguments are used primarily for the createinstance command. After the Oracle instance is created, the ORACLE_INSTANCE/bin/opmnctl directory provides a default Oracle instance home. The Oracle instance name is determined automatically.

The Oracle instance arguments are:
- **-oracleInstance**: the Oracle instance root directory. This directory is only required when the opmnctl command is not executed from the Oracle instance home.

- **-instanceName**: this command option is used for assigning an Oracle instance name when the Oracle instance is created or to verify the Oracle instance name afterward.

For the `createinstance` command, if the `instanceName` is not provided directly, the last directory name of the `-oracleInstance` is used as the Oracle instance name.

### 4.3.4.3.4 Component Specific Arguments

Commands that target a single component can accept arguments specific to the component type. For example, the `-wcAdminPasswordFile` argument is accepted when creating an Oracle Web Cache component but not when creating an OHS component.

These arguments are syntactically optional but must be understood to ensure proper configuration.

For details covering these specific arguments and how they should be used, refer to the *Oracle Fusion Middleware Administrator’s Guide*.

A list of the arguments applicable to a system component can be displayed by including the component type when executing the `opmnctl usage` command.

For example:

```
opmnctl usage createcomponent -componentType OHS
```

### 4.3.4.4 Commands

This section describes the `create` series of commands. It includes the following sections:

- **createinstance**
- **createcomponent**
- **deleteinstance**
- **deletecomponent**
- **registerinstance**
- **unregisterinstance**
- **updateinstanceregistration**
- **updatecomponentregistration**

#### 4.3.4.4.1 createinstance

The `createinstance` command creates an Oracle instance and registers it. It uses the following arguments:

- **Adminserver**
- **Logging**
- **Oracle Instance**

  - **-adminRegistration**: the registration status for the created Oracle instance. Supported values are **ON** (registered) and **OFF** (non-registered). The default value is **ON**.

  - **-opmnRemoteHost**: the OPMN remote host. The default value is the computer canonical host name.
- **-opmnLocalPort**: the OPMN local port value. The default port value is 6700 if the port is available.

- **-opmnRemotePort**: the OPMN remote port value. The default port value is 6701 if the port is available.

For example:

```
opmnctl createinstance -oracleInstance /foo/bar/inst1 -adminHost myadminserver -adminPort 7001
```

**or**

```
opmnctl createinstance -oracleInstance /foo/bar/inst1 -adminRegistration OFF
```

### 4.3.4.4.2 createcomponent

The `createcomponent` command creates a component within an Oracle instance. The created component is consistent with the Oracle instance in regard to its registration state.

The `createcomponent` command uses the following arguments:

- **Adminserver**
- **Logging**
- **Oracle Instance**
- **Component Specific Arguments**
  - **-componentType**: the system component type
  - **-componentName**: the system component name

For example:

```
opmnctl createcomponent -componentType OHS -componentName ohs1 -proxyPort 8888
```

It is important to note that different system components feature different sets of arguments that are available for configuration. In the above command example, the `-proxyPort` usage is specific to the OHS component. For more information about component specific arguments refer to Section 4.3.4.3.4.

### 4.3.4.4.3 deleteinstance

The `deleteinstance` command deletes an Oracle instance and unregisters the Oracle instance and associated system components. A successful Oracle instance deletion removes all files of the Oracle instance except for the default `provision.log` file.

For example:

```
opmnctl deleteinstance
```

The `deleteinstance` command uses the following arguments:

- **Adminserver**
- **Logging**
- **Oracle Instance**
- **-force**: The `-force` argument forces the `opmnctl deleteinstance` command to proceed regardless of discrepancies in the Oracle instance. The command is enabled only with a value of `true`. The default value is `false`.

Please note that the `-force` option circumvents most pre-validation checks. Therefore, Oracle recommends using the `-force` option with caution. An example
is shown in section **Force deleteinstance.**

**Force deleteinstance**
The `deleteinstance` command operates in a forced mode if the option `-force true` is included in the command. When used, this option directs `opmnctl` to proceed with the operation regardless of warnings and errors detected for the Oracle instance. Because the Oracle instance may be in a bad or non-responsive state, Oracle recommends explicitly providing non-required values that are associated and appropriate for the Oracle instance (for example, `-instanceName`, `-adminHost`, `-adminPort`) to ensure that the proper values are used.

During the course of cleanup, the forced `deleteinstance` command displays warnings and exceptions consistent with the damaged state of the Oracle instance. These warnings and exceptions are provided as visual feedback for the inconsistencies encountered and do not necessarily indicate that further corrective action is needed.

For example:

```
opmnctl deleteinstance -force true -instanceName instancel -adminHost myadminserver -adminPort 7001
or
	npmctl deleteinstance -force true -instanceName instancel
```

4.3.4.4.4  **deletecomponent**  The `deletecomponent` command deletes a system component.

The `deletecomponent` command uses the following arguments:

- Adminserver
- Logging
- Oracle Instance
- `-componentName`: The system component name.

For example:

```
opmnctl deletecomponent -componentName ohs1
```

4.3.4.4.5  **registerinstance**  The `registerinstance` command switches the Oracle instance to a registered state and updates the Adminserver value.

The `registerinstance` command uses the following arguments:

- Adminserver
- Logging
- Oracle Instance

For example:

```
opmnctl registerinstance -adminHost myhostname -adminPort 8000
```

4.3.4.4.6  **unregisterinstance**  The `unregisterinstance` command switches the Oracle instance to a non-registered state and updates the Adminserver value.

For example:
opmnctl unregisterinstance

The unregisterinstance command uses the following arguments:

- **Adminserver**
- **Logging**
- **Oracle Instance**
- **-force**: directs the `opmnctl unregisterinstance` command to proceed regardless of discrepancies in the Oracle instance. This option is only enabled with a value of true. The default value is false.

Please note that the `-force` option circumvents most pre-validation checks. Therefore, Oracle recommends using the `-force` option with caution. An example is shown in section Forced unregisterinstance

**Forced unregisterinstance**

The `unregisterinstance` command can operate in a forced mode when the option `-force true` is included.

This option directs `opmnctl unregisterinstance` to unregister the Oracle instance but limits the scope of these changes exclusively to the Adminserver. In other words, it unregisters an Oracle instance name from the Adminserver without examining or interacting with an Oracle instance (except optionally for writing to the `provision.log` file).

This mode can readily break registration associations. For example, executing this command with the name of an operational registered Oracle instance would put Adminserver and the Oracle instance out of synchronization. Afterward, the Adminserver no longer lists the Oracle instance or associated system components; however, in this example, the Oracle instance would still be configured as registered and but would fail most provisioning commands. Therefore, Oracle recommends using the `-force` option with caution.

For example:

```
opmnctl unregisterinstance -force true -instanceName instance1 -adminHost myadminserver -adminPort 7001
```

### 4.3.4.4.7 updateinstanceregistration

The `updateinstanceregistration` command updates information registered on the Adminserver for the Oracle instance. Specifically, the `updateinstanceregistration` command updates the registered OPMN remote port, OPMN remote host and OPMN wallet from the current OPMN settings.

The `updateinstanceregistration` command uses the following arguments:

- **Adminserver**
- **Logging**
- **Oracle Instance**

For example:

```
opmnctl updateinstanceregistration
```

### 4.3.4.4.8 updatecomponentregistration

The `updatecomponentregistration` command updates settings registered on the Adminserver for the subject system component.
These settings include the component proxy port, component proxy wallet, component mbean properties and Fusion Middleware Control Console component properties, plus other values specific to the type of component.

The `updatecomponentregistration` command uses the following arguments:

- **Adminserver**
- **Logging**
- **Oracle Instance**
- **Component Specific Arguments**
  - `-componentName`: the system component name

For example:

```
opmnctl updatecomponentregistration -componentName ohs1 -proxyPort 8889 ....
```

### 4.3.5 Status Commands

The `opmnctl status` commands enable you to determine the status of system component processes.

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

- `opmnctl status`
- `opmnctl metric`
- `opmnctl dmsdump`
- `opmnctl ping`
- `opmnctl set`
- `opmnctl query`

#### See Also:

- Section 4.3.1.2 for information about attributes
- Section 4.3.5.1.1 for information about the options for the status command of `opmnctl`
- Section 4.3.5.1.2 for information about the `opmnctl status -port` command

#### 4.3.5.1 `opmnctl status`

**Syntax:** `opmnctl status [options]`

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

The output is a text table. Each row in the table represents one system process.

You can customize the status command in the following ways:

- Change the information displayed about each system 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, “opmnctl Status Output” shows the output after entering the `opmnctl status` command for the AppSrv1 Oracle instance on host comp1 for the domain yourcompany.com:

**Example 4–3  opmnctl Status Output**

```bash
opmnctl status
```

**Processes in Instance: Instance1**

<table>
<thead>
<tr>
<th>Monitor</th>
<th>MONITOR</th>
<th>6720</th>
<th>Alive</th>
</tr>
</thead>
<tbody>
<tr>
<td>webcache1</td>
<td>WebCache-admin</td>
<td>N/A</td>
<td>Down</td>
</tr>
<tr>
<td>webcache1</td>
<td>WebCache</td>
<td>N/A</td>
<td>Down</td>
</tr>
<tr>
<td>ohs1</td>
<td>OHS</td>
<td>5106</td>
<td>Alive</td>
</tr>
</tbody>
</table>

4.3.5.1.1  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.

```bash
opmnctl status -l
```

For example, the command outputs the information shown in Example 4–4 (some columns are not shown to improve readability).

**Example 4–4  opmnctl status -l output**

<table>
<thead>
<tr>
<th>ias-component</th>
<th>process-type</th>
<th>pid</th>
<th>status</th>
<th>uid</th>
<th>memused</th>
<th>uptime</th>
<th>ports</th>
</tr>
</thead>
</table>

The `uid` information enables you to stop or restart an individual Oracle Fusion Middleware process.

For example, the following command stops the OHS process-type:

```bash
opmnctl stopproc uniqueid=1792746301
```

- `-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 `
`.

- `-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%pid5%sta8.

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

<table>
<thead>
<tr>
<th>Format String Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;statname&gt;</td>
<td>This must be one of the following:</td>
</tr>
<tr>
<td></td>
<td>• clu: system farm name</td>
</tr>
<tr>
<td></td>
<td>• ins: Oracle instance name</td>
</tr>
<tr>
<td></td>
<td>• cmp: system component ID</td>
</tr>
<tr>
<td></td>
<td>• prt: process-type ID</td>
</tr>
<tr>
<td></td>
<td>• prs: process-set ID</td>
</tr>
<tr>
<td></td>
<td>• idx: index of process in process-set</td>
</tr>
<tr>
<td></td>
<td>• pid: operating system process ID</td>
</tr>
<tr>
<td></td>
<td>• uid: OPMN uniqueid</td>
</tr>
<tr>
<td></td>
<td>• typ: name for this kind of process</td>
</tr>
<tr>
<td></td>
<td>• sta: process status</td>
</tr>
<tr>
<td></td>
<td>• stm: start time (ms)</td>
</tr>
<tr>
<td></td>
<td>• utm: up time (ms)</td>
</tr>
<tr>
<td></td>
<td>• cpu: cpu time (ms)</td>
</tr>
<tr>
<td></td>
<td>• mem: memory used (in KB)</td>
</tr>
<tr>
<td></td>
<td>• por: port list</td>
</tr>
<tr>
<td></td>
<td>• fpr: full detail port list</td>
</tr>
</tbody>
</table>

| <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**

webcache1@WebCache-admin@1544@Alive
webcache1@WebCache@1545@Alive
ohs1@OHS@1543@Alive

**See Also:** Section 4.3.1 for command definitions

Enter the following command for additional detailed information:
opmnctl usage status

4.3.5.1.2 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:

```bash
opmnctl status -port
```

displays:

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

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.

4.3.5.2 opmnctl metric

The opmnctl metric command enables you to print the DMS statistics for a system component process. The opmnctl metric has the following command and argument structure:

```bash
opmnctl metric op=<op> [ <identifier>=<value> ] dmsargs=<dmsargs>
```

The opmnctl metric command uses the following command arguments:

- **op=<op>:** Specifies the operation, where <op> can be either list or query.
  
  A list operation does not use additional arguments. The target OPMN returns a list of managed system processes for which DMS metrics are available through the metric request.
  
  A query operation returns the metric tree for the managed process or processes qualified by the <identifier>=<value> arguments (all available processes are assumed if no identifier arguments are specified).

- **local=<bool>:** When set to true, this option limits the command to operate within the local Oracle instance. This is useful when OPMN has been configured to discover other Oracle instances.

- **<identifier>=<value>:** Specifies which managed process or processes to query for DMS metrics. Multiple identifiers can be specified, and each is used to match the managed process with address properties (the list is inclusive).

The four standard properties are:

- **PROCESS_UID=<uid>:** Where <uid> is the unique id for the target process. This value is obtained by the metric list operation, the opmnctl status command, or any similar method that is used to display OPMN DMS metrics.

- **PROCESS_UID=0:** This property always specifies OPMN itself.

- **INSTANCE_UID=<instance>@<uid>:** Where <instance> is the Oracle instance name for the target Oracle instance and <uid> is the unique id for the target process on the Oracle instance. This identifier is only valid if <instance> is the local Oracle instance name, or all OPMN servers in the farm are configured to connect to one another.

- **COMPONENT_NAME=<component-name>:** Where <component-name> is the id value for the <ias-component> in the opmn.xml file. COMPONENT_NAME=opmn always specifies OPMN itself.
- **COMPONENT_TYPE=<{component-type}>**: Where `{component-type}` is the type value for the `<ias-component>` in the `opmn.xml` file, or if that is not defined, the id value for the `<process-type>` element. `COMPONENT_TYPE=OPMN` always specifies OPMN itself.

- **dmsarg=<dmsargs>**: Specifies the DMS specific options for the metric query.
  <dmsargs> is a single argument that may include multiple options for the DMS layer, each following the `<attr>=<value>` format, and each option separated by an `&`.

  For example: `format=raw, format=xml&name=/ProcessInfo`

  Refer to the *Oracle Fusion Middleware Performance Guide* for additional metric options.

### 4.3.5.3 opmnctl dmsdump

**Syntax:** `opmnctl 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 system components. If no attributes are specified, performance data for all system component processes for your system 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:

```
opmnctl dmsdump 'table=opmn_ons&format=xml'
```

outputs 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 Fusion Middleware Performance Guide*.

### 4.3.5.4 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,

```
opmnctl ping 10
```

designates pinging of OPMN 10 times until the ping command succeeds.

### 4.3.5.5 opmnctl set

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

The `opmnctl set` command sets the logging configuration for OPMN.
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 `standard.log` or the `debug.log`. The standard log is configured to include all possible log messages by default and it is recommended that users not change its settings. Refer to Appendix B for more information about OPMN troubleshooting.

---

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

---

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

**4.3.5.5.1 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**

<table>
<thead>
<tr>
<th>ONS element</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>all subcomponents</td>
</tr>
<tr>
<td>local</td>
<td>local information</td>
</tr>
<tr>
<td>listener</td>
<td>listener information</td>
</tr>
<tr>
<td>discover</td>
<td>discover (server or multicast) information</td>
</tr>
<tr>
<td>servers</td>
<td>remote servers currently up and connected to the farm</td>
</tr>
<tr>
<td>topology</td>
<td>current farm wide server connection topology</td>
</tr>
<tr>
<td>server</td>
<td>remote server connection information</td>
</tr>
<tr>
<td>client</td>
<td>client connection information</td>
</tr>
<tr>
<td>connect</td>
<td>generic connection information</td>
</tr>
<tr>
<td>subscribe</td>
<td>client subscription information</td>
</tr>
<tr>
<td>message</td>
<td>notification receiving and processing information</td>
</tr>
<tr>
<td>deliver</td>
<td>notification delivery information</td>
</tr>
<tr>
<td>special</td>
<td>special notification processing</td>
</tr>
<tr>
<td>internal</td>
<td>internal resource information</td>
</tr>
</tbody>
</table>
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]

yields all ons subcomponents excluding topology, while the following:

ons![topology,all]

yields all ons subcomponents including topology.

### 4.3.5.6 opmnctl query

Syntax: opmnctl 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 B.2.1, "OPMN log Files" for more information.

### 4.3.6 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 4.3.6.1, "opmnctl help"**
- **Section 4.3.6.2, "opmnctl usage"**
- **Section 4.3.6.3, "opmnctl validate"**

#### 4.3.6.1 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**

```plaintext
opmnctl help

usage: opmnctl [verbose] [scope] <command> [options]

verbose: print detailed execution message if available

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

<table>
<thead>
<tr>
<th>scope</th>
<th>command</th>
<th>options</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>start</td>
<td>- Start opmn</td>
</tr>
<tr>
<td></td>
<td>startall</td>
<td>- Start opmn &amp; all managed processes</td>
</tr>
<tr>
<td></td>
<td>stopall</td>
<td>- Stop opmn &amp; all managed processes</td>
</tr>
<tr>
<td></td>
<td>shutdown</td>
<td>- Shutdown opmn &amp; all managed processes</td>
</tr>
<tr>
<td></td>
<td>startproc</td>
<td>[&lt;attr&gt;=&lt;val&gt;...] - Start system managed processes</td>
</tr>
<tr>
<td></td>
<td>restartproc</td>
<td>[&lt;attr&gt;=&lt;val&gt;...] - Restart managed processes</td>
</tr>
<tr>
<td></td>
<td>stopproc</td>
<td>[&lt;attr&gt;=&lt;val&gt;...] - Stop managed processes</td>
</tr>
<tr>
<td></td>
<td>reload</td>
<td>- Trigger opmn to reread opmn.xml</td>
</tr>
<tr>
<td></td>
<td>status</td>
<td>[&lt;options&gt;] - Get managed process status</td>
</tr>
<tr>
<td></td>
<td>dmsdump</td>
<td>[&lt;attr&gt;=&lt;val&gt;...] - Get DMS stats</td>
</tr>
<tr>
<td></td>
<td>set</td>
<td>[&lt;attr&gt;=&lt;val&gt;...] - Set opmn log parameters</td>
</tr>
<tr>
<td></td>
<td>query</td>
<td>[&lt;attr&gt;=&lt;val&gt;...] - Query opmn log parameters</td>
</tr>
<tr>
<td></td>
<td>launch</td>
<td>[&lt;attr&gt;=&lt;val&gt;...] - Launch a configured target process</td>
</tr>
<tr>
<td></td>
<td>phantom</td>
<td>[&lt;attr&gt;=&lt;val&gt;...] - Register phantom processes</td>
</tr>
<tr>
<td></td>
<td>ping</td>
<td>[&lt;max_retry&gt;] - Ping local opmn</td>
</tr>
<tr>
<td></td>
<td>validate</td>
<td>[&lt;filename&gt;] - Validate the given xml file</td>
</tr>
<tr>
<td></td>
<td>config</td>
<td>[options] - Modify the opmn xml file</td>
</tr>
<tr>
<td></td>
<td>help</td>
<td>- Print brief usage description</td>
</tr>
<tr>
<td></td>
<td>usage</td>
<td>- Print detailed usage description</td>
</tr>
<tr>
<td></td>
<td>createinstance</td>
<td>- Create an Oracle Instance</td>
</tr>
</tbody>
</table>
```
createcomponent - Create a specified component
deleteinstance - Delete an instance and components
deletecomponent - Delete a specified component
registerinstance - Register with admin server
unregisterinstance - Unregister with admin server
updateinstanceregistration - Update instance registration
updatecomponentregistration - Update component registration

4.3.6.2 opmnctl usage
Syntax: opmnctl usage [<command>]

The usage command displays help for any one specified opmnctl command. With no argument the opmnctl usage command displays the same information as opmnctl help.

The command can be one or more of the following:

- start
- startall
- startproc
- stopall
- stopproc
- restartproc
- reload
- shutdown
- ping
- status
- metric
- dmsdump
- debug
- set
- query
- launch
- phantom
- ping
- validate
- help
- usage
- createinstance
- createcomponent
- deleteinstance
- deletecomponent
- registerinstance
- unregisterinstance
- updateinstanceregistration
- updatecomponentregistration
- help

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

```bash
opmnctl usage stopall
```

**Example 4–7  opmnctl usage stopall output**

```bash
opmnctl stopall [sequential=true] [report=true]
```

Stop opmn daemon and managed processes for local Oracle instance.

This request first tries to stop all 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 managed processes without stopping the opmn daemon, consider using the stopproc command without any target arguments.

If the value of the sequential attribute is "true", each process will be stopped upon sequentially (one at a time).

If the value of the report attribute is "true", opmn will report the results of the attempt to stop each processes as it completes. The default behavior is for opmn to wait until the entire request has completed before sending all of the results at once.

To restart the opmn daemon without restarting any managed processes, consider using 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.

### 4.3.6.3 opmnctl validate

**Syntax:** `opmnctl validate [<filename>]`

The `opmnctl validate` command validates the XML syntax of the `opmn.xml` file. The default `ORACLE_INSTANCE/config/OPMN/opmn/opmn.xml` is validated when `ORACLE_INSTANCE/bin/opmnctl` is used. The `<filename>` can be specified by either the relative or absolute path.
This chapter provides command-line examples on how to use OPMN for Oracle Fusion Middleware. It features the following topics:

- Starting OPMN
- Starting and Stopping All System Components
- Starting and Stopping a System Component
- Starting and Stopping all System Components of the Same Type

### 5.1 Starting OPMN

OPMN does not depend on any other system 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 system components:

```bash
opmnctl start
```

### 5.2 Starting and Stopping All System Components

Use the following command to **start** all system components of an Oracle instance:

```bash
opmnctl startproc
```

Use the following command to **stop** all system components of an Oracle instance:

```bash
opmnctl stopproc
```

### 5.3 Starting and Stopping a System Component

Use the following command to **start** a system component named **ohs1**:

```bash
opmnctl startproc ias-component=ohs1
```

Use the following command to **stop** a system component named **ohs1**:

```bash
opmnctl stopproc ias-component=ohs1
```
5.4 Starting and Stopping all System Components of the Same Type

Use the following command to **start** all system components of the type **OID** in an Oracle instance:

```
opmnctl startproc process-type=OID
```

Use the following command to **stop** all system components of the type **OID** in an Oracle instance:

```
opmnctl stopproc process-type=OID
```
This chapter provides common configuration examples, and descriptions of elements and attributes for the OPMN opmn.xml file.

It contains the following topics:

- Example of opmn.xml Elements and Attributes
- 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 system components.

6.1.1 Global Definitions and Syntax

All paths, module data values, and environment values may reference the following global variables, which is expanded to their platform specific values:

<table>
<thead>
<tr>
<th>Global Variable</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORACLE_HOME</td>
<td>Full path to ORACLE_HOME</td>
</tr>
<tr>
<td>ORACLE_INSTANCE</td>
<td>Full path to ORACLE_INSTANCE</td>
</tr>
<tr>
<td>:</td>
<td>The library path (or class path) separator (&quot;:&quot; on Linux or &quot;;&quot; on Microsoft Windows, for example).</td>
</tr>
<tr>
<td>EXE_EXT</td>
<td>The executable file extension (&quot;&quot; on Linux or &quot;.exe&quot; on Microsoft Windows, for example)</td>
</tr>
<tr>
<td>SO_EXT</td>
<td>The shared library extension (&quot;.so&quot; on Linux or &quot;.dll&quot; on Microsoft Windows, for example)</td>
</tr>
<tr>
<td>SHELL_EXT</td>
<td>The shell file extension (&quot;.sh&quot; on Linux or &quot;.bat&quot; on Microsoft Windows, for example)</td>
</tr>
</tbody>
</table>

Each of the variables defined above must be referenced using the same syntax as an environment variable in Linux or Windows. For example any of these references, ORACLE_HOME, (ORACLE_HOME), or %ORACLE_HOME% yields the full path to ORACLE_HOME.

OPMN also automatically converts path separators in any path, module data value, or environment value based upon the platform on which OPMN is running. For example, /oracle/instances/ias1 would remain the same on Linux, but on Microsoft Windows OPMN would convert it to \oracle\instances\ias1.
Example 6–1  Common Configuration Elements and Attributes

```xml
<opmn>
  <log path="path" comp="comp-codes" debug="comp-codes" rotation=-size="kBytes" rotation-hour="HOD"/>
  <notification-server interface="type">
    <ipaddr remote="ip; ip" request="ip; ip"/>
    <port local="port" remote="port" request="port"/>
    <ssl enabled="boolean" wallet-file="path" wallet-password="password" ssl-versions="versions"
      ciphers="ciphers"/>
    <tune io-timeout="timeout" io-idle="interval" timeout="timeout"/>
  </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>
  </process-manager>
  <ias-instance="id="ias-instance-name" name="ias-instance-name" ORACLE_HOME="path">
    <environment>
      <variable id="id" value="value" append="boolean" process-conversion="boolean"/>
    </environment>
    <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"/>
      <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>
    <process-type="process-type-id" module-id="module-id" status="state" working-dir="path">
      <!-- environment-->
      <!-- module-data-->
      <!-- dependencies-->
      <event-scripts>
        <pre-start path="path"/>
        <pre-stop path="path"/>
        <post-crash path="path"/>
      </event-scripts>
      <start timeout="timeout" retry="num"/>
      <stop timeout="timeout"/>
    </process-type">
  </ias-instance>
</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 system components. 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_INSTANCE/diagnostics/logs/OPMN/opmn/opmn.log
Valid Values: The path name for the OPMN log file.
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 default comp-codes for log already include all possible log messages, and it is recommended that you do not change the value.

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 6–2 lists the ONS component codes.

<table>
<thead>
<tr>
<th>ONS Attribute</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>all subcomponents</td>
</tr>
<tr>
<td>local</td>
<td>local information</td>
</tr>
<tr>
<td>listener</td>
<td>listener information</td>
</tr>
<tr>
<td>discover</td>
<td>discover (server or multicast) information</td>
</tr>
<tr>
<td>servers</td>
<td>remote servers currently up and connected to the farm</td>
</tr>
<tr>
<td>topology</td>
<td>current farm wide server connection topology</td>
</tr>
<tr>
<td>server</td>
<td>remote server connection information</td>
</tr>
<tr>
<td>client</td>
<td>client connection information</td>
</tr>
<tr>
<td>connect</td>
<td>generic connection information</td>
</tr>
<tr>
<td>subscribe</td>
<td>client subscription information</td>
</tr>
<tr>
<td>message</td>
<td>notification receiving and processing information</td>
</tr>
<tr>
<td>deliver</td>
<td>notification delivery information</td>
</tr>
<tr>
<td>special</td>
<td>special notification processing</td>
</tr>
<tr>
<td>internal</td>
<td>internal resource information</td>
</tr>
<tr>
<td>secure</td>
<td>SSL operation information</td>
</tr>
<tr>
<td>workers</td>
<td>worker threads</td>
</tr>
</tbody>
</table>
The following Table 6–3 lists the PM component codes.

**Table 6–3 PM Component Codes**

<table>
<thead>
<tr>
<th>PM Attribute</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>all subcomponents</td>
</tr>
<tr>
<td>requests</td>
<td>HTTP (user) requests</td>
</tr>
<tr>
<td>remote</td>
<td>remote HTTP requests</td>
</tr>
<tr>
<td>scheduler</td>
<td>scheduler thread and resource information</td>
</tr>
<tr>
<td>monitor</td>
<td>monitor thread information</td>
</tr>
<tr>
<td>workers</td>
<td>worker threads</td>
</tr>
<tr>
<td>process</td>
<td>managed processes</td>
</tr>
<tr>
<td>depend</td>
<td>dependency processing</td>
</tr>
<tr>
<td>internal</td>
<td>internal resources</td>
</tr>
<tr>
<td>schedjobs</td>
<td>periodic scheduled jobs</td>
</tr>
<tr>
<td>procjobs</td>
<td>for each process scheduled jobs</td>
</tr>
<tr>
<td>dms</td>
<td>DMS processing</td>
</tr>
<tr>
<td>modules</td>
<td>process module information. Only modules which call the modLog() (or modDebug for the debug log) function yields 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.</td>
</tr>
</tbody>
</table>

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] yields all ons subcomponents excluding topology, while ons[!topology,all] yields all ons subcomponents including topology.

**debug="comp-codes"**

Required: false
Default: none
Valid Values: A list of comp-codes for debug logging. A semi-colon must be used to separate the items on the comp-codes list.

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

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

See `comp="comp-codes"` for the list of valid comp-codes and syntax.

**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 closes 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 closes the log, renames it with a time stamp suffix, and then creates a new log file. This attribute may be used with rotation-size.

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 supports both the IPv6 and IPv4 network interfaces (see Section 3.5 for information about 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 ipaddr 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 IPv4 or IPv6 format) or host name to which ONS binds 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 binds 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 IPv4 or IPv6 format) or host name to which ONS binds 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 binds 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, ssl-versions, and ssl-ciphers.

The ssl element is used for ONS to ONS security and authentication configuration.

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.

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.

ssl-versions="versions"
Required: false (for Oracle SSL only)
Default: See the text below.
Valid values: A comma separated list of SSL versions.

The list of versions to use for SSL connections. The following list specifies the valid versions that may be configured, and is the default set:

- TLSv1.0
- TLSv1.1
- TLSv1.2
- SSLv3.0

ssl-ciphers="ciphers"
Required: false (for Oracle SSL only)
Default: See the text below.
Valid values: A comma separated list of cipher suites.

The list of cipher suites to use for SSL connections. The following list specifies the valid cipher suites that may be configured, and those flagged with an asterisk (*) compose the default set.

- SSL_RSA_WITH_AES_256_GCM_SHA384 *
- SSL_RSA_WITH_AES_128_GCM_SHA256 *
- SSL_RSA_WITH_AES_256_CBC_SHA256 *
SSL_RSA_WITH_AES_128_CBC_SHA256 *
SSL_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 *
SSL_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 *
SSL_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 *
SSL_ECDHE_ECDSA_WITH_AES_256_CBC_SHA *
SSL_ECDHE_ECDSA_WITH_AES_128_CBC_SHA *
SSL_RSA_WITH_AES_256_CBC_SHA *
SSL_RSA_WITH_AES_128_CBC_SHA *
SSL_RSA_WITH_3DES_EDE_CBC_SHA *
SSL_RSA_WITH_RC4_128_SHA
SSL_RSA_WITH_RC4_128_MD5

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 times-out the connection and close the socket. The io-timeout value is the timeout period remote OPMN (or ONS) servers uses 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.

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 continues 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.

\[ \text{timeout}=\text{timeout} \]

Required: false  
Default: 20  
Valid Values: an integer value \( \geq 1 \) and \( \leq 3600 \)

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.

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 only allows 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 automatically appends 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 terminates 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 creates 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 6–4 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>
<thead>
<tr>
<th>Module</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ias-instance</td>
<td>&lt;category id=&quot;CatA&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;data id= &quot;DataAA&quot; value=&quot;aaaa&quot;/&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/category&gt;</td>
</tr>
<tr>
<td>ias-component</td>
<td>&lt;category id=&quot;CatA&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;data id= &quot;DataAB&quot; value=&quot;abab&quot;/&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/category&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;category id=&quot;CatB&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;data id= &quot;DataBA&quot; value=&quot;baba&quot;/&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/category&gt;</td>
</tr>
<tr>
<td>module</td>
<td>&lt;category id=&quot;CatA&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;data id= &quot;DataAC&quot; value=&quot;acac&quot;/&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/category&gt;</td>
</tr>
<tr>
<td>process-type</td>
<td>&lt;category id=&quot;CatA&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;data id= &quot;DataAA&quot; value=&quot;XXXX&quot;/&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/category&gt;</td>
</tr>
<tr>
<td>process-set</td>
<td>&lt;category id=&quot;CatB&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;data id= &quot;DataBB&quot; value=&quot;bbbb&quot;/&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/category&gt;</td>
</tr>
</tbody>
</table>
Table 6–4 (Cont.) module-data Hierarchy

<table>
<thead>
<tr>
<th>Module</th>
<th>Definition</th>
</tr>
</thead>
</table>
| 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 `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%` is 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 converts 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 yields a "/" in the string. Specify two ^ characters if you need to specify the ^ character in the resultant string: ^^ yields ^.

**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`

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

`id = ias-instance-id; name = ias-instance-name`

**id="ias-instance-name"**
Required: true
Default: none
Valid Values: A string.
The string specifies the ias-instance id.

```xml
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 Oracle 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 creates 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.

OPMN provides a default set of environment variables for each managed process, and a set of Java system properties for managed Java processes.

Environmental variables are described in Table 6–5:

<table>
<thead>
<tr>
<th>Environmental Variable</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORACLE_HOME</td>
<td>Full path to the oracle home</td>
</tr>
<tr>
<td>ORACLE_INSTANCE</td>
<td>Full path to the Oracle instance home</td>
</tr>
<tr>
<td>INSTANCE_NAME</td>
<td>The name of the Oracle instance</td>
</tr>
<tr>
<td>PROCESS_UID</td>
<td>The unique process id</td>
</tr>
<tr>
<td>OPMN_INDEX</td>
<td>Synonym of PROCESS_INDEX</td>
</tr>
<tr>
<td>PROCESS_INDEX</td>
<td>The index of the process</td>
</tr>
<tr>
<td>OPMN_INDEX</td>
<td>Synonym of PROCESS_INDEX</td>
</tr>
<tr>
<td>COMPONENT_NAME</td>
<td>The name of the component</td>
</tr>
<tr>
<td>COMPONENT_TYPE</td>
<td>The type of the component</td>
</tr>
<tr>
<td>COMPONENT_CONFIG_PATH</td>
<td>Absolute path to the configuration files on disk for your component</td>
</tr>
<tr>
<td>COMPONENT_LOG_PATH</td>
<td>Absolute path to the log files on disk for your component</td>
</tr>
<tr>
<td>ORACLE_NLS</td>
<td>Defaults to ORACLE_NLS from the OPMN environment.</td>
</tr>
<tr>
<td>OPMN_ENV_LC_ALL</td>
<td>Defaults to OPMN_ENV_LC_ALL from the OPMN environment</td>
</tr>
<tr>
<td>OPMN_ENV_LANG</td>
<td>Defaults to OPMN_ENV_LANG from the OPMN environment</td>
</tr>
<tr>
<td>OPMN_ENV_NLS_LANG</td>
<td>Defaults to OPMN_ENV_NLS_LANG from the OPMN environment</td>
</tr>
</tbody>
</table>

---

**Table 6–5  Environmental Variables**
System Properties set for all managed Java processes (flagged as such to OPMN by the process module) are described in Table 6–6:

Table 6–6  System Properties

<table>
<thead>
<tr>
<th>Environmental Variable</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>oracle.process.uid</td>
<td>The unique process id</td>
</tr>
<tr>
<td>oracle.opmn.uid</td>
<td>Synonym of oracle.process.uid</td>
</tr>
<tr>
<td>oracle.process.index</td>
<td>The index of the process</td>
</tr>
<tr>
<td>oracle.opmn.index</td>
<td>Synonym of oracle.process.index</td>
</tr>
<tr>
<td>oracle.home</td>
<td>Full path to the oracle home</td>
</tr>
<tr>
<td>oracle.instance</td>
<td>Full path to the Oracle instance home</td>
</tr>
<tr>
<td>oracle.instance.name</td>
<td>The name of the Oracle instance</td>
</tr>
<tr>
<td>oracle.component.name</td>
<td>The name of the component</td>
</tr>
<tr>
<td>oracle.component.type</td>
<td>The type of the component</td>
</tr>
<tr>
<td>oracle.component.configpath</td>
<td>Absolute path to the configuration files on disk for your component</td>
</tr>
<tr>
<td>oracle.component.logpath</td>
<td>Absolute path to the log files on disk for your component</td>
</tr>
</tbody>
</table>
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 takes 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% is 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 or from the environment of OPMN.

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".

```xml
<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 delimiter 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

ias-component
Required: true
Default: none
Parents: <ias-instance>
Attributes: id, id-matching, status, type

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.

**type="component-type"**

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

The `type` attribute allows you to specify a relationship among `ias-components` within the same `ias-instance`.

When configured the `component-type` is used for the value of the per process `COMPONENT_TYPE` environment variable (or the `oracle.component.type` system property for Java processes), and for the `component-type` element of the paths specified by `COMPONENT_CONFIG_PATH` and `COMPONENT_LOG_PATH` (or `oracle.component.configpath` and `oracle.component.logpath`). If `component-type` is not configured, then the `process-type id` attribute is used in the composition of these strings.

**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 creates 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 are made before starting a process. OPMN creates 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 are made before starting a process.

There are two primary types of dependencies: external and internal. External dependencies are for resources not managed by OPMN. For example: Fusion Middleware Control Console.

An external program is executed by OPMN to perform the check on the resource. Internal dependencies are for system component 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 determines how long OPMN waits 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.
opmn.xml Element and Attribute Descriptions

**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>)))`

**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 waits 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 uses 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 is 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 is performed for this check. A value of 0 indicates OPMN always performs 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 an address string for a specific Oracle Internet Directory. This value is set to true to use the default 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 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.

**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 waits 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 uses 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 always performs the check.

OSSO

Required: false
Default: none
Parents: <dependencies>
Attributes: host, port, URI, timeout, cache-timeout

The OSSO element specifies the Oracle Single Sign-On service to check.

host="hostname"
Required: true
Default: none
Valid Values: A string

The host attribute is the host name for the Oracle Single Sign-On connection.

port="port"
Required: true
Default: none
Valid Values: A port number

The port attribute is the port for the Oracle Single Sign-On connection.

URI="uri"
Required: true
Default: none
Valid Values: A string

The URI attribute is the URI for the Oracle 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 Oracle Single Sign-On connection.

**enabled=boolean**
- Required: true
- Default: none
- Valid Values: true or false

The `enabled` attribute enables the SSL connection to Oracle 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 Oracle 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 waits 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 uses 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 always performs 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 tries to get 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 waits 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 uses 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 access of the dependency is performed for this check. A value of 0 indicates OPMN always performs the check.

---

**Note:** The cache-timeout is only for the last successful check of the dependency, and if the previous check failed, OPMN performs 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: none
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.

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 6–7 shows event script arguments.

<table>
<thead>
<tr>
<th>Option Name</th>
<th>Option Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-timeStamp</td>
<td>&lt;time&gt;</td>
<td>An integer value for the current time on the system (in seconds).</td>
</tr>
<tr>
<td>-instanceName</td>
<td>&lt;instance-name&gt;</td>
<td>The instance-name of the managed process.</td>
</tr>
<tr>
<td>-componentId</td>
<td>&lt;component-id&gt;</td>
<td>The component-id of the managed process.</td>
</tr>
<tr>
<td>-processType</td>
<td>&lt;process-type-id&gt;</td>
<td>The process-type of the managed process.</td>
</tr>
<tr>
<td>-processSet</td>
<td>&lt;process-set-id&gt;</td>
<td>The process-set of the managed process.</td>
</tr>
<tr>
<td>-processIndex</td>
<td>&lt;index&gt;</td>
<td>The process-index of the managed process.</td>
</tr>
<tr>
<td>-stderr</td>
<td>&lt;path&gt;</td>
<td>The path name for the stderr file pointer of the process.</td>
</tr>
</tbody>
</table>
Table 6–7 (Cont.) Event Script Arguments

<table>
<thead>
<tr>
<th>Option Name</th>
<th>Option Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-stdout¹</td>
<td>&lt;path&gt;</td>
<td>The path name for the stdout file pointer of the process. Note: this argument is only given for a pre-start script if the start is part of a process restart request.</td>
</tr>
<tr>
<td>-reason</td>
<td>&lt;reason&gt;</td>
<td>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.</td>
</tr>
<tr>
<td>-pid²</td>
<td>&lt;process-id&gt;</td>
<td>The operating system integer value given for the process-id.</td>
</tr>
<tr>
<td>-startTime²</td>
<td>&lt;time&gt;</td>
<td>An integer value for the system start time of the process (in seconds).</td>
</tr>
</tbody>
</table>

¹ This argument is only 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 is not started and any associated HTTP request fails.

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 hangs 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.

**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 fails. However, OPMN proceeds 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 hangs until it times out.

```xml
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.

---

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 hangs until it times out.

```xml
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.

---

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.

```xml
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 is 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 is 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

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 is 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: false
Default: id="component-id" numprocs="1"
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.

If a process-set is not configured, then one is automatically created using the same id as the ias-component and numprocs of 1.

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: true
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 disable automatic restarting of terminated managed processes set the attribute to false.

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_INSTANCE  
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.

**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.
launch-targets

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

The launch-targets block is composed of a list of launch-target definitions. There can be only one launch-targets block.

launch-target

Required: false
Default: none
Parents: <launch-target>
Attributes: id

The definition of a process which OPMN starts via the launch request.

id="id"
Required: true
Default: none
Valid values: A string.

The identifier for this launch-target. Each launch-target must have a unique id. This identifier is specified by the target argument of the launch request.

eexec

Required: true
Default: none
Parents: <launch-target>
Attributes: path

The executable path for the launch-target. Only one exec element may be configured per launch-target.

path="path"
Required: true
Default: none
Valid values: A path to an executable.
Specify the full path to the executable file.

argument

Required: false
Default: none
Parents: <launch-target>
Attributes: value

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).
An argument always passed to the configured executable process. Multiple argument elements may be configured per launch-target, and the arguments are passed to the executable in the order in which they are configured.

Any additional arguments passed in via the launch request are passed in after the configured arguments.

**value**="argument"
Required: true
Default: none
Valid values: A string.

Specify a single argument for the executable; spaces are not argument delimiters-use multiple argument elements to define multiple arguments.

**timeout**
Required: false
Default: 120
Parents: <launch-target>
Attributes: value

The timeout used for each io made from the launch request thread. Only one timeout element may be configured per launch-target.

**value**="seconds"
Required: true
Default: none
Valid values: An integer value between 5 and 3600.

The io timeout value in seconds.

**max-concurrency**
Required: false
Default: 5
Parents: launch-target
Attributes: value

The maximum number of concurrent processes for this launch-target. Only one max-concurrency element may be configured per launch-target.

**value**="maximum"
Required: true
Default: none
Valid values: An integer value between 1 and 250.

The maximum number of concurrent processes for this launch-target.
This chapter describes custom process configuration in the OPMN `opmn.xml` file.

It features the following topics:

- Custom Process Module Configuration
- Custom Process Minimum Configuration
- Custom Process Complete Configuration
- Custom Process Attribute Descriptions

A.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`.

```xml
<module path="ORACLE_HOME/opmn/lib/libopmncustom.so">
  <module-id id="CUSTOM" />
</module>
```

A.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.

```xml
<iis-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>
</iis-component>
```

A.3 Custom Process Complete Configuration

Example A–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 system component. In such cases, the process-type element in Example A–1 must be part of the component configuration.

### A.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 configuration example Example A–1 shows a custom process using HTTP ping. Example A–1 is an example of script ping that you can substitute into the component configuration.

**Example A–1  Custom Process Complete Configuration**

```xml
<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" />
          <data id="append-req-args" value="Arguments appended to the end of the start command"/>
          <data id="java-proc" value="true"/>
        </category>
        <category id="stop-parameters">
          <data id="stop-executable" value="Your stop executable here" />
          <data id="stop-args" value="Your stop args here" />
          <data id="append-req-args" value="Arguments appended to the end of the stop command"/>
        </category>
        <category id="restart-parameters">
          <data id="restart-executable" value="Your restart executable here"/>
          <data id="restart-args" value="Your restart args here" />
          <data id="append-req-args" value="Arguments appended to the end of the restart command"/>
        </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 A–2.

**Example A–2 Ping Type Script**

```xml
<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 determines that the process did not start correctly and stop it.

---

### A.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 Appendix A for configuring a custom process.
- 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="java-proc"**
- Required: false
- Default: false
- Valid values: true or false
- Path: ias-component/process-type/process-set/module-data/category/data

This data element specifies if the process is a Java process. If set to true, OPMN adds the default managed process system properties to the start-args.

**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="append-req-args"
Required: false
Default: none
Path: ias-component/process-type/process-set/module-data/category

The value of append-req-args is appended to the end of the constructed start command.

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 executes 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="append-req-args"
  Required: false
  Default: none
  Path: ias-component/process-type/process-set/module-data/category

  The value of append-req-args is appended to the end of the constructed restart command.

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, integrator
  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 A–1 for information about complete custom process configuration
  ■  Example A–2 for information about the 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 is 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 is not performed. If an invalid hostname is specified, the process-set is 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 is not performed. If an invalid port is specified, the process-set is 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 waits for a ping response. If a response is not obtained within the timeout period, the ping attempt is 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.
This chapter describes some troubleshooting tips for OPMN. It features the following topics:

- Problems and Solutions
- Diagnosing OPMN Problems
- Need More Help?

B.1 Problems and Solutions

This section describes some of the common problems encountered when using OPMN. It features the following topics:

- System Process Does Not Start
- Determining if System Processes are Dying or Unresponsive
- opmncntl Command Execution Times Out
- System Component Automatically Restarted by OPMN
- Unexpected opmncntl start Behavior
- Disabled Element in the opmn.xml File
- Unable to Start OHS
- Unable to Stop Component
- globalInitNLS Error
- OPMN Start Up Consumes CPU Processing Capability
- Error Messages During Start-up of OPMN
- Increasing Size of opmn.log File
- Cleanup of Oracle Instance or Component If It is in a Bad State
B.1.1 System Process Does Not Start

Problem
Unable to start a system process using OPMN.

Solution
Try the following if you are unable to start a system 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 Oracle Application Server or Oracle Fusion Middleware for starting OPMN or system components.

- Most managed processes now have their own log directory, and OPMN uses that if it is present as the location for the process log file.

  ORACLE_INSTANCE/diagnostic/logs/<type>/<ias-component>/

If the default directory for a managed process does not exist, OPMN places the console log in its own log directory:

  ORACLE_INSTANCE/diagnostics/logs/OPMN/opmn/

For example, the standard output log for Oracle HTTP Server may be

  ORACLE_INSTANCE/diagnostics/logs/OHS/ohs1/console-OHS-1.log

- Verify the dependency requirements for the system process you are attempting to start.

- Verify the element values for the system 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 system component you are attempting to start.

B.1.2 Determining if System Processes are Dying or Unresponsive

Problem
Your system processes are dying or unreachable.

Note: Currently you can create an ORACLE_INSTANCE and OHS components without any issues, but when you run opmnctl startall, you get the following error:

[2014-10-24T17:29:53-06:00] [opmn] [ERROR:1] [32] [internal] libopmnoraclesl.dll LoadLibrary failed: 14001

[2014-10-24T17:29:53-06:00] [opmn] [ERROR:1] [] [ons-secure] Cannot Load library: libopmnoraclesl.dll

This is as per the Webtier requirement of msvcr90.dll.
Solution
If your system processes are dying or unreachable:

- Review the system component specific output in the ORACLE_INSTANCE/diagnostics/logs/OPMN/opmn.

  Look at the ORACLE_INSTANCE/diagnostics/logs/OPMN/opmn/opmn.log for system processes. Look for process crashed or process unreachable messages. OPMN automatically restarts system processes that die or become unresponsive.

See Also: Section B.2.1 for more information about 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 B.2.4 for information about troubleshooting with event scripts

B.1.3 opmnctl Command Execution Times Out

Problem
The time it takes to execute an opmnctl command is dependent on the type of system process and available computer hardware. Because of this the time it takes to execute an opmnctl command may not be readily apparent.

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 allows OPMN to wait for process to come up. This functionality is available with the startproc command which starts all the relevant processes configured in the opmn.xml file.

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_INSTANCE/diagnostics/logs/ for the system process that is not starting.

4. Review the component-specific log file for the system component that is not starting. For example, ORACLE_INSTANCE/diagnostics/logs/OHS/ohs1/console-OHS-1.log.

  See Also: Chapter 6 for more information about the common configuration of the opmn.xml file

B.1.4 System Component Automatically Restarted by OPMN

Problem
A system component is automatically restarted by OPMN.

Solution
If a system component is automatically restarted by OPMN, try the following:

- Review the message for the system component in the ORACLE_INSTANCE/diagnostics/logs/OPMN/opmn/opmn.log file.
- Verify that the ping timeout for the system component is sufficient. A system 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 system component `opmn.xml` file.

### B.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 system 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 system component processes are started.

**Solution**

Oracle recommends that you shutdown OPMN before shutting down your computer. Use the `opmnctl stopall` command to stop OPMN and system component processes.

On the Microsoft Windows operating system, you can use the Windows services control panel to stop OPMN and system components.

---

**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 uses 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 setting the following attribute for the `process-manager` element in the `opmn.xml` file:

```
restore-procs-on-reboot="true"
```

---

### B.1.6 Disabled Element in the `opmn.xml` File

**Problem**

Unable to start an system process.

**Solution**

If you are unable to start an system process, check if an element in the system `opmn.xml` file is disabled. If an element in the `opmn.xml` file is disabled OPMN generates an output message of "Missing" or "Disabled".
B.1.7 Unable to Start OHS

**Problem**
If you have multiple Oracle Fusion Middleware installations on one host and you start them at the same time (for example, to start a farm), 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 instances on the same host use the same ports. An OHS component in one Oracle instances is trying to use the same port as an OHS component in a different Oracle instance. When two OPMN processes on a host start at the same time, there is no coordination between them on port usage.

Port allocation for all OHS instances within Oracle Fusion Middleware 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**
Make sure the ports assigned to OHS in each Oracle instance are unique and not used by any other process on the computer.

It is also recommended that you increase the maximum number of retries for starting Oracle instances. If you have identical port ranges in two Oracle homes and increase the number of times OPMN attempts to restart a process, OPMN eventually selects 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.

B.1.8 Unable to Stop Component

**Problem**
If you are unable to stop system components 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**
System components should never be started or stopped manually. Do not use command line scripts or utilities from previous versions of Oracle Application Server or Oracle Fusion Middleware for starting and stopping system components.

Use the Fusion Middleware Control Console and the opmnctl command line utility to start or stop system components.

**See Also:** Chapter 5 for OPMN command-line examples

B.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.

### B.1.10 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 continues until OPMN is shutdown.

**Solution**
One of the possible cause for the excessive CPU processing consumption is the installation environment used multibyte text character sets such as Japanese.

### B.1.11 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_INSTANCE/config/OPMN/opmn/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):
  
  `opmnctl validate opmn.xml`
- remove all empty tags from the `opmn.xml` file.

### B.1.12 Increasing Size of opmn.log File

**Problem**
If you install Oracle Fusion Middleware on a computer that contains a previous installation of the Oracle Database or Oracle Fusion Middleware, the `ORACLE_INSTANCE/diagnostics/logs/OPMN/opmn/opmn.log` file increases in size to over 4100000 KB due to continuous logging. The file may contain the following error message:

"[ons-connect] Local connection 127.0.0.1,6100 invalid form factor "
Solution
Change the request port for the opmn.xml file to a value greater than the 6100. For example:

```xml
<port local='6202' remote='6302' request='6105'/>
```

This error is most often caused by a conflict with a previous Oracle Database or Oracle Fusion Middleware installation on the same computer. The above occurs due to entries in the oraInventory directory of an existing or previous installation of Oracle Database or Oracle Fusion Middleware.

B.1.13 Cleanup of Oracle Instance or Component If It is in a Bad State

Problem
Your Oracle instance or component is providing some incorrect defaults for your opmnctl commands.

Solution
Use the opmnctl deleteinstance command with the -force true option to clean up the Oracle instance or component.

For example:

```
opmnctl deleteinstance -oracleInstance /scratch/bsong/demo1/inst1 -instanceName inst1 -adminHost myhost -adminPort myport -force true
```

Use discretion when utilizing the -force option. When you use this option with the opmnctl deleteinstance command a number of safeguards that protect the integrity of the Oracle instance or component are overruled. Misuse of this option may produce undesirable results.

Explicitly listing all applicable arguments (for example, oracleInstance, instanceName, adminHost, and adminPort) with the opmnctl deleteinstance command can help attain the desired results.

During the course of executing cleanup, a forced command displays warnings or exceptions consistent with the damaged state of the Oracle instance or component. These warning are provided as visual feedback for the inconsistencies encountered and do not necessarily indicate that further corrective action is needed.

B.1.14 Path Arguments Containing Quotation Marks on Windows Operating Systems

Problem
The path argument provided to opmnctl contains quotes ("), and everything that appears after the first quote (including other flags intended for opmn) is included in the ORACLE_Instance name. This is a Windows operating system issue only.

For example:

```
opmnctl createinstance -oracleInstance %T_WORK%"instl" -adminRegistration OFF
```

In this example, the name of the ORACLE_INSTANCE will be:

```
inst1-adminRegistration OFF
```
Solution
The quotes must encompass the entire argument or the command.
For example:

```
opmnctl createinstance '%T_WORK%\inst1' -adminRegistration OFF
```

In the corrected example, the name of the ORACLE_INSTANCE will be:

```
inst1
```

B.2 Diagnosing OPMN Problems

There are several methods for troubleshooting any problems you may have using OPMN:

- OPMN log Files
- `opmnctl debug`
- Oracle Enterprise Manager Fusion Middleware Control Console
- Troubleshooting with Event Scripts
- `opmn.xml` Environment Variables

B.2.1 OPMN log Files

The OPMN log files enable you to troubleshoot difficulties you might have in execution and use of OPMN and system component processes. OPMN and system component 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_INSTANCE/diagnostics/OPMN/opmn/opmn.out`: contains the standard output (stdout) and standard error (stderr) logs of OPMN. Also referred to as the OPMN “console log”. Only a small set of messages ever appears in this file; therefore, this file may not be present if you conduct a search through the log file directories. Output written to the `opmn.out` file contains the exit status of a child OPMN process.

- `ORACLE_INSTANCE/opmn/logs/opmn.log`: tracks command execution, operation progress and debug information if enabled. It contains messages useful for monitoring the operations of the OPMN server. The `opmn.log` file is configured using the `<log>` attribute in the `opmn.xml` file. Refer to Chapter 6 for more information about the common configuration of the `opmn.xml` file.

B.2.1.1 opmn.log File Rotation

OPMN enables you to rotate the `opmn.log` file 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>` tag 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 closes the file, rename the file with a unique time stamp suffix, and then create a new `opmn.log` or `debug.log` 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 appears in this file.

B.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 renames 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 is not able rotate the process console log file.

B.2.2 opmnctl debug
Use the opmnctl debug command to verify the status of a system 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 debug [comp=pm|ons] [interval=<secs> count=<num>]
```
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.

```
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.

```
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.

B.2.3 Oracle Enterprise Manager Fusion Middleware Control Console
Fusion Middleware Control Console provides a graphical interface that enables diagnosis of system components in your network and enterprise. Fusion Middleware Control Console features a log page. The log page enables you to view all of the system log files in one place and trace problems across multiple log files. Fusion Middleware Control Console uses an API that contacts OPMN.

See Also: Oracle Fusion Middleware Administrator’s Guide

B.2.4 Troubleshooting with Event Scripts
You can create your own event scripts that record system process event activities. You can create a script that records events prior to the start or stop of system processes, as well as an unscheduled system crash.
Refer to the event-scripts element description in Chapter 6.

Example B–1 shows a pre-start event script.

Example B–1  Pre-start Event Script

```bash
#!/bin/sh
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

timeStamp=$timeStamp
instanceName=$instanceName
componentId=$componentId
processType=$processType
processSet=$processSet
processIndex=$processIndex
stderrPath=$stderrPath
stdoutPath=$stdoutPath
reason=$reason
pid=$pid
startTime=$startTime
```
Note: The pre-start event script example, Example B-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.

B.2.5 opmn.xml Environment Variables

The environment variable used to launch OPMN server is not inherited by the system 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 for more information about the common configuration of the opmn.xml file

B.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 Fusion Middleware Release Notes, available on the Oracle Technology Network:
http://www.oracle.com/technology/documentation/
A
Application Server Control, 2-2, 4-6, B-9
async, 4-4, 4-9
attribute name, 4-9
attribute syntax, 4-3

C
cache-timeout, 6-25
cache-timeout attribute, 6-22
category element, 6-13
clu, 4-20
cmp, 4-20
command, 4-2
command definitions
  opmnctl, 4-3
comp attribute, 4-23
comp-codes, 6-4
complete configuration
  custom process, A-1
component codes, ONS, 6-4
component codes, PM, 6-5
component-id, 6-18
count, B-9
cpu, 4-20
cron, 6-12
custom id, A-3
custom module
  ping, A-2
CUSTOM module-id, A-4
custom process
  complete configuration, A-1
  minimum configuration, A-1
custom process module, A-1

D
data element, 6-13
database element, 6-20
db-connect-info, 6-20
dependencies element, 6-19
dmsdump
  opmnctl, 4-22

E
enabled, 6-8
environment element, 6-15
event script arguments, 6-26
event scripts, B-9
event-scripts element, 6-26
example elements, 6-1

F
-fmt option, 4-20
fmtlist, 4-20
-fmtlist syntax, 4-20
form factor key, 3-3
.formfactor file, 3-3
fpr, 4-20
-fsep option, 4-19

H
help
  opmnctl, 4-2, 4-25
HTTP ping, A-2

I
ias-component, 4-8, 4-9
ias-component element, 6-18
ias-component-id, 6-24
ias-instance attribute, 6-24
ias-instance element, 6-14
ias-instance-id, 6-24
idx, 4-20
infrastructure, 6-21
infrastructure-key, 6-20
ins, 4-20
insecure-remote-requests, 6-10
interface, 6-6
interval, B-9
io-idle, 6-9
io-timeout, 6-9
ipaddr element, 6-19
IPv4, 3-4
IPv6, 3-4
justification, 4-20

-l option, 4-19
log, 6-3
log files
  OPNM, B-8
log page, B-9

managed-process element, 6-23
max_retry, 4-22
maxprocs, 6-32
mem, 4-20
minimum configuration
  custom process, A-1
minprocs, 6-32
mode, 4-9
module, 6-11
module element, 6-11
module-data element, 6-12
module-id element, 6-14
modules
  PM, 2-4

-noheaders option, 4-19
notification-server element, 6-6
numprocs, 6-31

OID element, 6-20
ONS, 6-7
ONS (Oracle Notification Server), 2-3
ONS component codes, 6-4
OPMN, 2-1, 2-2, 2-4
  functionality, 2-1
log files, B-8
OPMN daemon, 4-6, 4-9
OPMN local listener, 3-3
OPMN Log Files, 3-2
opmnctl, 2-2, 4-1, 4-3, 4-6
  help, 4-2, 4-25
  options, 4-19
  ping, 4-22
  process control commands, 4-8
  quick reference, 4-2
  stopall, 4-6
  syntax, 4-2
  usage, 4-7, 4-8
  validate, 4-27
opmnctl attributes, 4-3
opmnctl command definitions, 4-3
opmnctl commands, 4-2
opmnctl debug, B-9
opmnctl dmsdump, 4-22
opmnctl metric, 4-21
opmnctl query, 4-24
opmnctl reload, 4-7, 4-8
opmnctl restartproc, 4-9
opmnctl set, 4-22
opmnctl shutdown, 4-7
opmnctl start, 4-6
opmnctl startall, 4-6
opmnctl startproc, 4-9
opmnctl status syntax, 4-18
opmnctl stopproc, 4-7, 4-8, 4-9
opmnctl usage status, 4-19
opmnctl usage syntax, 4-26
opmnctl validate, B-2
opmnctl verbose syntax, 4-5
opmn.xml file, 2-3, 3-1
option
  -fmt, 4-20
  -fsep, 4-19
  -l, 4-19
  -noheaders, 4-19
  -rsep, 4-19
options, 4-2
  opmnctl, 4-19
Oracle Enterprise Manager Application Server Control, 2-5
Oracle Identity Management Installation, 2-1
Oracle instance, 2-2
Oracle Notification Server see ONS
Oracle Portal, Forms, Reports, and Discoverer Installation, 2-1
Oracle Process Manager Modules see PM Modules, 2-4
Oracle Process Manager see PM, 2-3
OSSO element, 6-22
path, 6-3
pid, 4-20
ping
  custom module, A-2
  opmnctl, 4-22
pong element, 6-30
ping-host id, A-7
ping-timeout id, A-7
ping-url id, A-7
PM, 2-3
PM component codes, 6-5
PM Modules, 2-4
PM modules, 3-1
por, 4-20
post element, 6-7, 6-30
post-crash element, 6-28
pre-stop element, 6-27
process control command
  opmnctl, 4-8
process crashed, B-3
process module
  custom, A-1
process unreachable, B-3
process-conversion, 6-14
process-manager element, 6-10
process-modules element, 6-11
process-set, 4-8, 4-9
process-set element, 6-31
process-set-id, 6-24, 6-31
process-set-id attribute, 6-24
process-type, 4-8, 4-9
process-type-id attribute, 6-24
process-type-id, 6-24
progressive request reports, 4-10
prs, 4-20
prt, 4-20
publish-subscribe model, 2-3

Q
query
  opmnctl, 4-24
quick reference
  opmnctl, 4-2

R
ready-parameters id, A-8
reload
  opmnctl, 4-7, 4-8
remote, 6-7
request, 6-7
restart element, 6-29
restart-args id, A-6
restart-executable id, A-6
restartproc
  opmnctl, 4-9
rotation-hour, 6-6
rotation-size, 6-5
  -rsep option, 4-19

S
script ping, A-2
script-args id, A-8
script-executable id, A-8
sequential requests, 4-10
set
  opmnctl, 4-22
shutdown
  opmnctl, 4-7
SSL, 3-3, 6-21
SSL element, 6-8, 6-22
sta, 4-20
start
  opmnctl, 4-6
start element, 6-28, B-3
startall
  opmnctl, 4-6
start-args id, A-4
start-executable id, A-4
starting OPMN, 5-1
startproc
  opmnctl, 4-9
statname, 4-20
status
  opmnctl, 4-20
stm, 4-20
stop element, 6-29
stopall
  opmnctl, 4-6
stop-args id, A-5
stop-executable id, A-5
stopproc
  opmnctl, 4-7, 4-8, 4-9
sync, 4-4, 4-9
Syntax
  opmnctl ping, 4-22
syntax, 4-3
  attribute, 4-3
  -fmtlist, 4-20
  opmnctl, 4-2
  opmnctl help, 4-25
  opmnctl options, 4-19
  opmnctl reload, 4-8
  opmnctl restartproc, 4-9
  opmnctl shutdown, 4-7
  opmnctl start, 4-6
  opmnctl startall, 4-6
  opmnctl startproc, 4-9
  opmnctl status, 4-18
  opmnctl stopall, 4-6
  opmnctl stopproc, 4-9
  opmnctl usage, 4-26
  opmnctl validate, 4-27
  opmnctl verbose, 4-5
syntax value, 4-3

tag-id, 6-11
timeout, 4-4, 4-9
timeout attribute, 6-22, B-3
timeout value, 4-6
timeout="depend-timeout", 6-22
tune, 6-9
typ, 4-20

U
duid, 4-19, 4-20
uniqueid, 4-4, 4-9, 4-19
usage
  opmnctl, 4-7, 4-8, 4-26
use-ping-for-ready id, A-8
utm, 4-20
V
validate
   opmnctl, 4-27
value, 4-3
value syntax, 4-3
variable element, 6-17
verbose, 4-2
   opmnctl, 4-5

W
wallet-file, 6-8, 6-21
wallet-password, 6-8, 6-21
Webtier Installation, 2-1
width, 4-20
working-dir, 6-32
working-dir attribute, 6-26