

This FAQ addresses frequently asked questions relating to the Oracle Application Server 10g Release 3 (10.1.3) version of Oracle HTTP Server (OHS). This FAQ also covers mod_oc4j and is broken into the following two sections:

[Oracle HTTP Server \(OHS\)](#)
[Mod_oc4j](#)

1.0 Oracle HTTP Server (OHS)

1.1 What version of Apache is Oracle HTTP Server based on?

Two versions. Apache version 1.3.34 for OHS based on Apache 1.3 and Apache version 2.0.55 for OHS based on Apache 2.0.

1.2 Are Apache modules, not provided by Oracle, supported when integrated with OHS?

Oracle only provides support for modules included in the Oracle distribution. Oracle does not support modules obtained from any other source, including the Apache Software Foundation. However, OHS will still be supported when non-Oracle provided modules are included. If Oracle Support suspects that a non-Oracle provided module is contributing to a reported problem, customers may be requested to reproduce the problem without that module being included.

1.3 What is Oracle's policy on fixing security bugs found in OHS?

Oracle's policy and process for fixing security vulnerabilities can be found on [Oracle's SecAlert OTN page](#).

1.4 Is Oracle HTTP Server available as a standalone product too?

Yes. A new 'Web Server and Process Management' installation option let you install just the Oracle HTTP Server based on Apache 1.3 (with Oracle Process Manager and Notification Server). Also, Oracle HTTP Server based on Apache 2.0 is available to be installed as a standalone product off the Oracle Application Server 10g (10.1.3) Companion CD.

1.5 Is Apache v2.0 version of OHS supported with this release?

Yes. However, OHS based on Apache 2.0 is only supported in a standalone deployment version. It has the same functionality as OHS based on Apache 1.3 except for the following:

- IPv6 is supported in OHS based on Apache 2.0 but not in OHS based on Apache 1.3
- mod_oradav is not supported in OHS based on Apache 2.0
- mod_dms is not supported in OHS based on Apache 2.0
- mod_plsql is not supported in OHS based on Apache 2.0

1.6 Does OHS need to run as root?

No. OHS needs to run as root only when users want to use port less than 1024. If this will never be the case, then they can run OHS as the user that installed Oracle Application Server rather than root. In order to do this, perform the following steps:

- 1 Shutdown OHS
- 2 Become root
- 3 `cd $ORACLE_HOME/Apache/Apache/bin`
- 4 `chown root .apachectl`
- 5 `chmod 6750 .apachectl`
- 6 `cd $ORACLE_HOME/Apache/Apache/logs`
- 7 `rm -f *`
- 8 If you are using `mod_osso`, re-register `mod_osso`
- 9 Exit root
- 10 Restart OHS

1.7 Can I compress output from OHS (ex. gzip)?

In general, the recommendation is to use Web Cache for this purpose. There are other freeware modules (for example, `mod_gzip`) that may be plugged in for this purpose - but their use is not supported.

1.8 Why do I see a warning about the use of EAPI when starting OHS with a non-Oracle provided module?

Oracle HTTP Server is compiled with EAPI (Extended API) support. The EAPI is an extension to the Apache module API provided by `mod_ssl` (see <http://www.modssl.org>). If you see a message similar to the following example when starting OHS with a non-Oracle provided module it means that the module being loaded was not compiled with the EAPI.

```
[Mon Oct 31 12:11:37 2005] [warn] Loaded DSO
libexec/mod_python.so uses plain Apache 1.3 API, this module
might crash under EAPI! (please recompile it with -DEAPI)
```

In most cases, this warning message can safely be ignored. To eliminate the error message, the module can be recompiled using the `-DEAPI` compile time option and either the Apache header files provided with OHS in `$ORACLE_HOME/Apache/Apache/include` or the header files from a generic Apache instance that includes `modssl` and the corresponding EAPI additions.

1.9 Can a standalone OHS based on Apache 2.0 communicate with an existing Oracle Application Server 10g Release 3 (10.1.3) instance?

Yes. Standalone OHS based on Apache 2.0 can be configured to communicate with an existing Oracle Application Server 10g Release 3 (10.1.3) instance. Refer to chapter 2 'Configuring Standalone Oracle HTTP Server with Oracle Application Server' of *Oracle HTTP Server Standalone Administrator's Guide Based on Apache 2.0* for details on how to configure it.

1.10 Can an OHS of Oracle Application Server 10g Release 2 (10.1.2) be used to route requests to Oracle Application Server 10g Release 3 (10.1.3) instance?

Yes. You can configure an OHS of Oracle Application Server 10g Release 2 (10.1.2) middle-tier instance to communicate with Oracle Application Server 10g Release 3 (10.1.3) instance. Refer to chapter 6 'Reconfiguring Application Server Instances' of *Oracle Application Server Administrator's Guide 10g Release 3 (10.1.3)* for details on how to configure it.



2.0 Mod_oc4j

2.1 What is mod_oc4j?

Mod_oc4j is the load balancer for the requests going to the Oracle Application Server Container for J2EE (OC4J) Instances in Oracle Application Server. It is an OHS module that provides routing between OHS and OC4J. The Oracle Process Manager and Notification Server (OPMN) component of Oracle Application Server keeps mod_oc4j aware of the status of different OC4J processes - thus, mod_oc4j routes only to the processes that are up and running. Mod_oc4j also understands the concepts of Oracle Application Server Cluster and OC4J groups, and routes accordingly to provide as much transparent failover as possible.

2.2 Does mod_oc4j work with web servers other than OHS?

Yes. Mod_oc4j is available as a plug-in too, called OC4J Plug-in, to work with non-Oracle web servers too including IIS, iPlanet, and generic Apache.

2.3 What are the different routing/load balancing algorithms?

Mod_oc4j provides three distinct kinds of routing: (a) round robin, (b) random and (c) metric based. The effective performance of round robin and random algorithms is the same. The latter, metric based routing, is based on OC4J process informing mod_oc4j of a metric based on its internal resource availability (ex. connection pools). Mod_oc4j then uses this metric to make routing decisions.

These load balancing/routing algorithms also have a flavor - affinity based. In this mode (it is the default mode), these algorithms will always route to the local node, except in cases when no process is available on the local node. The random and round robin algorithms have an extra flavor - weight based. In case of weight based, mod_oc4j distributes requests according to the routing weight configured for each host. Refer to *Oracle HTTP Server Administrator's Guide* for more details on load balancing algorithms.

2.4 Can mod_oc4j talk to OC4J using SSL?

Yes, the AJP communication between mod_oc4j and OC4J processes can now be over AJP/SSL. Previously, this was in the clear text. Also, the SSL negotiation does not happen each time the two need to talk - resulting in less performance impact.

2.5 There are no Oc4jMount directives in my mod_oc4j.conf file, how does mod_oc4j know where to route the requests?

In previous releases of Oracle Application Server (version 10.1.2 and earlier), OC4J mount points were statically configured in mod_oc4j.conf file. Thus, when a user deployed or un-deployed an application, mod_oc4j.conf file was updated and OHS restarted.

In Oracle Application Server 10g Release 3 (10.1.3), OC4Js announce their mount-point(s) in the notifications they send out and mod_oc4j dynamically adjusts its routing table using this information. This eliminates the need for static mount point configuration and enables mod_oc4j to update its mount point configuration dynamically (without restarting OHS).

2.6 Can I still use the old static mount point configuration?

Yes. Although dynamic mount point creation is enabled by default, you do have the option of continuing to use statically configured mount points. You can configure this by setting a new directive called Oc4jRoutingMode to 'Static' in mod_oc4j.conf file. Directive Oc4jRoutingMode specifies the routing behavior and can take one of the following values:

- Dynamic – This specifies that the new dynamic routing functionality is used and any old style routing configuration is ignored.
- Static – This informs mod_OC4J to use 9.0.4/10.1.2 style routing configuration (where mount points are explicitly listed). Dynamic routing is not used.
- DynamicOverride – Both dynamic and 9.0.4/10.1.2 style routing are used by mod_oc4j. If there are conflicts, OHS routes to the dynamically specified mount points.
- StaticOverride – Both dynamic and 9.0.4/10.1.2 style routing are used by mod_oc4j. If there are conflicts, OHS routes to the statically specified mount points.

2.7 Is it possible to dump/view the list of current mount points mod_oc4j is using?

Yes, you can dump/view the in-memory mod_oc4j routing table contents by going to the following URL:

```
http://localhost:<dms_port>/oc4j-status
```

The information you will find here includes configured load balancing algorithm, routing mode, routing id, application name, context and the OC4J process(s) the application requests are routing to.

The URI for this pages is configurable, and by default is located in the dms.conf file in \$ORACLE_HOME/Apache/Apache/conf directory. The default configuration only allows it to be accessible from the localhost (127.0.0.1) Virtual Host. The configuration snippet looks like this:

```
<IfModule mod_oc4j.c>
    Oc4jSet StatusUri /oc4j-status
</IfModule>
```

You can place this snippet anywhere in the httpd.conf if you want to see that status page under the default server, instead of only accessing it from the localhost Virtual Host. If you do that, the URL to dump/view routing table will be:

```
http://<host.domain>:<http_port>/oc4j-status
```

2.8 What is routing ID?

The routing ID specifies a routing relationship between OC4Js and OHSs. In other words, an OHS routes to every OC4J that it shares a routing ID with. Every OC4J is assigned a routing ID, similarly each OHS is assigned one or more routing IDs to route to.

OPMN passes the routing ID to OC4J as a system property and to OHS as an environment variable when it is started. OC4J adds this routing ID to the ONS notifications it publishes. OHS listens for notifications from OC4J. When an OHS sees the first notification from an OC4J containing a routing ID on its list, it begins routing to it.

The addition of routing IDs and mount point discovery in Oracle Application Server 10g Release 3 (10.1.3) version of OHS allows mod_OC4J to dynamically discover all aspects of OC4J routing.

2.9 Should I configure routing ID for OHS in both mod_oc4j.conf and opmn.xml files?

No. Out of the box, OHS is configured to pick up its routing ID from opmn.xml file. Though it is possible to configure routing IDs for OHS in both opmn.xml and directly in mod_oc4j.conf file, but if OHS is configured with routing-id in both places, it considers it an error and fails to start. So routing IDs for OHS should either be configured in opmn.xml (specified as module data under <ias-instance> element or under <ias-component> element of OHS) or mod_oc4j.conf but not at both places.



ORACLE FUSION MIDDLEWARE

Oracle Application Server 10g R3:

Oracle HTTP Server FAQ

November 2005

Author: Shail Goel

Contributing Author:

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:

+1.650.506.7000

Fax +1.650.506.7200

<http://www.oracle.com/>

Copyright © Oracle Corporation 2005

All Rights Reserved

This document is provided for informational purposes only, and the information herein is subject to change without notice. Please report any errors herein to Oracle Corporation. Oracle Corporation does not provide any warranties covering and specifically disclaims any liability in connection with this document.

Oracle is a registered trademark of Oracle Corporation.

All other company and product names mentioned are used for identification purposes only and may be trademarks of their respective owners.