ORACLE
FUSION MIDDLEWARE

An Oracle White Paper
October 2014

Oracle HTTP Server 12cR1 - Technical Overview

ORACLE
FUSION MIDDLEWARE

Table of Contents

Introduction

Oracle HTTP Server 12c provides the key infrastructure for serving the Internet's HTTP(s) protocol and handles the *Web Tier* responsibilities with robust content serving and reverse proxy capabilities.

The key aspects of Oracle HTTP Server (OHS) are:

- **Technology:** OHS is based on the proven, open source technology - Apache HTTP Server. OHS 11g (11.1.1.x) and 12cR1 (12.1.2/12.1.3) release is based on the latest versions of the Apache 2.2 release. Oracle HTTP Server also includes additional capabilities that are very relevant to the Fusion Middleware deployment and are typically not available within the open source Apache HTTP Server.
- **Content Serving:** OHS provides the ability to serve static web content (like HTML5, images etc.) to the end-user and also to cache this static web content in memory for better throughput. In addition, OHS includes the ability to serve dynamic web content that are based on web scripting technologies like PHP, Python, Ruby etc. through its built-in CGI/FastCGI modules.
- **Reverse Proxy:** Act as a '*Reverse Proxy*' and transparently pass through the incoming HTTP(s) requests to the back-end content (application or simply '*origin*') server. This allows administrators to configure rules within OHS to authenticate and authorize the incoming HTTP(s) requests before efficiently routing to one or more content (application or simply '*origin*') servers.
- **Integration:** Oracle HTTP Server 12c includes end-to-end integration with the Fusion Middleware Applications allowing administrators to provision, configure, administer, and monitor the entire Fusion Middleware Application deployment. In addition, Oracle HTTP Server also supports stand-alone deployment scenario and supports front-ending non-Oracle HTTP servers through its built-in reverse proxy 'mod_proxy' module.

  Oracle HTTP Server 12c and above leverages the **WebLogic Management Framework** to deliver consistent administration within Oracle HTTP Server, Oracle Web Logic Server and the rest of the Fusion Middleware applications.

  This document provides a technical overview of Oracle HTTP Server and the new administration related capabilities available within the Oracle HTTP Server 12cR1 (12.1.x) release.

Oracle HTTP Server Capability Overview

Oracle HTTP Server 12.1.3 is the latest release within the Oracle HTTP Server 12cR1.

Here is the high level summary of all the capabilities included within the latest Oracle HTTP Server (OHS) 12.1.3 release.

- **Protocol:** OHS is capable of handling and responding to simultaneous HTTP 1.0/1.1 and HTTPS (SSL 3.0 and TLS 1.2) traffic from multiple users.
- **Mass Virtual Hosting:** Enables customers (large enterprises or ISPs) to host several web sites with different name within a single OHS instance. This allows administrators to consolidate multiple web

sites and its associated business applications within a single OHS instance. This capability is typically achieved by configuring multiple *Virtual Hosts* within Apache HTTP Server based OHS. For more information, please see Virtual Hosts section within Apache HTTP Server documentation.

- **URL Rewriting:** This capability allows administrators to quickly reorganize their web sites without any impact on externally visible URLs. For more information, please see URL Rewrite section within Apache HTTP Server documentation.
- **Content Serving:** OHS typically serves the client requested content through one of these ways – send the content from the disk, send the output generated from executing the CGI/FastCGI scripts, send the output generated by the back-end application server (reverse proxy).
- **Content Caching:** Administrators can leverage the built-in caching capabilities of the Apache HTTP Server to cache the contents either to the disk or to the memory thereby improving the web site throughput and response time. For more information, please see Content Caching section within Apache HTTP Server documentation.
- **Reverse Proxy:** OHS includes the Apache HTTP Server provided built-in reverse proxy modules like mod_proxy to front-end the incoming request to the back-end application server and send the response back to the client. This setup allows administrators to leverage OHS to implement their corporate policies (like authentication, authorization etc.) before the request is sent to the back-end HTTP compliant application server(s).
- **WebLogic Proxy Plug-In:** OHS bundles WebLogic Web Server Proxy Plug-In for Apache. This capability allows OHS to efficiently load balance the incoming traffic to the back-end WebLogic managed servers or clusters. In addition, management console accessible within OHS 12c also provides the administration capability for this module.
- **Authentication/Authorization:** OHS bundles Apache HTTP Server provided modules related to implementing the authentication and authorization policies. For more information on how to implement the various authentication / authorization policies, please refer to Authentication and Authorization section within Apache HTTP Server documentation.
- **Single Sign-On:** Employees and Corporate IT teams prefer that the employee authentication (username / password) information is shared and accessible between multiple web applications deployed within their corporate Intranet. Administrators can leverage the Web Gate module (mod_webgate) bundled within OHS to enable single sign-on with Oracle Access Management (OAM). This capability allows administrators to provision configure and administer the Oracle Single Sign-On solution.
- **SSL:** OHS is designed to handle and terminate SSL connections while processing the HTTPS traffic. Unlike the open source version of the Apache HTTP Server, OHS handles such SSL traffic through its own SSL libraries (mod_ossl module) to meet the strict corporate security requirements. These SSL libraries support the latest version of the SSL protocols including TLS 1.0/1.1 protocols. Accordingly, administrators can leverage OHS 12c to process HTTPS traffic on their web sites with either SSL v3 or TLS 1.0/1.1 protocols. For more information, please see SSL Cipher product documentation. OHS 12c includes the following additional capabilities:

  - **Hardware Offloading Support –** Offload AES crypto transactions to the hardware crypto on Intel x86 and SPARC T4/T5 processors and thereby significantly improve the overall HTTPs traffic throughput.
  - **FIPS 140-2 Level 1** - FIPS 140-1 certified runtime libraries. This capability allows administrators to comply with FIPS 140-1 (Level 1) certification by using Oracle HTTP Server based Web Tier solution. For more information how to enable this FIPS 140-1 certified runtime libraries within OHS, please refer to SSL configuration section within Oracle HTTP Server documentation.
  - **Variable Security per Resource** - This feature allows different web resources (like Directories that hosts web contents) to be protected by different strength encryption.
  - **SSL in Reverse proxy scenario** - OHS not only supports terminating incoming SSL connections but also supports initiating a SSL connection, when OHS is deployed in a Reverse Proxy mode (either using the built-in mod_proxy or using WebLogic Web Server Plug-In - mod_wl_ohs), with the back-end origin server.

- **Management:** OHS 12c administration leverages the WebLogic Management Framework to provide a consistent administration experience with the rest of the Fusion Middleware applications. The OHS 12c administration now supports administering one or more OHS instances running on one or more hosts (machines). Users can administer OHS using the management console – commonly referred as – **Fusion Middleware Control**. To leverage this console, user will need to choose to install OHS in a WebLogic Server Domain (or simply 'Collocated') scenario. For more information, please refer to section - Administering OHS 12c with WebLogic Server Domain.

Oracle HTTP Server Architecture Overview

Oracle HTTP Server (OHS) 12cR1 (12.1.2/12.1.3) is based on the Apache HTTP Server 2.2 architecture and accordingly leverages the highly modular architecture within the Apache HTTP Server 2.2.

The Apache HTTP Server architecture can be simplified as an architecture that uses one of the Multi-Processing Modules (MPM) at its core and all other capabilities including the ability to serve HTTP traffic is designed as modules. Oracle HTTP Server (OHS), unlike Apache HTTP Server, uses only the Worker MPM as its core architecture for handling the HTTP traffic. This architecture allows OHS to seamlessly handle the content serving as well as the reverse proxy use cases.

When Apache HTTP Server is processing an incoming HTTP request, various modules (and its APIs) are appropriately invoked at the appropriate request / response lifecycle.

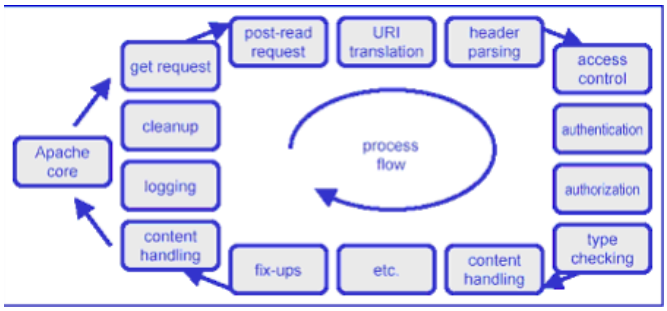

Figure – HTTP Request – Response Cycle

Customers can leverage this modular architecture and develop their own custom Apache HTTP Server 2.x compatible modules to supplement the functionality provided by OHS. However, Oracle Support may request that these custom modules are removed from the OHS configuration if there is a sufficient reason to believe that these modules can interfere, while troubleshooting, with the customer reported OHS related issues.

Oracle HTTP Server Modules

Apache HTTP Server modules are the plug-in modules to the core to allow the administrators to extend the basic functionality of Apache HTTP Server either as content server or as a (reverse) proxy server.

Oracle HTTP Server (OHS) includes most of the standard modules included within the standard Apache 2.2 release. In addition, OHS carries these additional modules to provide a seamless integration with the Fusion Middleware Architecture:

For more information, please visit Oracle HTTP Server 12c modules documentation.

Table- Additional Modules available within OHS 12cR1

Module Name	Module Version	Module Description
Mod_wl_ohs	12.1.x.0.0	WebLogic Web Server Proxy Plug-In for Oracle HTTP Server – This module allows OHS to effectively front-end the HTTP traffic to the back-end WebLogic managed server or clusters. This module replaces the functionality of the traditional 'mod_proxy' module while front-ending WebLogic Server applications.
Mod_ossl	12.1.x.0.0	Oracle SSL module that intercepts the SSL traffic coming to OHS to handle SSL handshake and then to appropriately terminate the SSL traffic.

		This module replaces the traditional 'mod_ssl' module while handling SSL traffic. This module leverages the RSA provided cypto libraries and accordingly offers FIPS 140-2 (Level 1) certification. For more information, please refer to product documentation.
Mod_Perl		This module allows customers to run their Perl scripts within OHS. **This module is now marked as 'Deprecated' and will be removed in future releases.**
Mod_plsql	2.1.22	This module allows administrators to connect to the Oracle Database from the OHS and thereby create web applications using the stored procedures capabilities available within the Oracle Database. **This module is now marked as 'Deprecated' and will be removed in future releases.**
Mod_fastcgi	2.4.6	This module supports the FastCGI protocol to allow users to efficiently run the traditional CGI applications thereby eliminating the startup and initialization overhead. **This module is now marked as 'Deprecated' and will be either removed or replaced with another module in future releases.**
Mod_security	2.6	Web Application Firewall module, based on open source Mod Security module. Administrators can leverage the Mod Security rule capabilities to develop and deploy appropriate rules to protect their web applications from known security vulnerabilities like SQL Injection, Cross Site Scripting Request Forgery (CSRF), Command Injection etc. Administrators can also leverage the open source OWASP Core Rule Set - a set of pre-built rules - as a foundation to protect their application from these well known security vulnerabilities.
Mod_dms	12.1.x.0.0	Provides Monitoring information from every OHS server instances to the Oracle Fusion Middleware Control Console or Oracle Enterprise Manager Cloud Control to provide a unified monitoring information across the entire farm.

ORACLE
FUSION MIDDLEWARE

Oracle HTTP Server 12c Administration Overview

This section provides an overview of Oracle HTTP Server 12c Administration and introduces the different scenarios and steps in administering *Oracle HTTP Server 12c* through the *WebLogic Management Framework*.

Oracle HTTP Server (OHS) 12c offers the ability to administer OHS and the rest of the Cloud Application Foundation 12c infrastructure using the WebLogic Management framework. This integrated administration approach allows administrators to manage their entire infrastructure through a consistent interface.

WebLogic Management Framework

WebLogic Management Framework 12c is the framework designed to administer system components like Oracle HTTP Server and the rest of the Fusion Middleware applications including WebLogic Server and provides two ways of administering Oracle HTTP Server:

- **WebLogic Server Domain** – Contains a WebLogic Administration Server, zero or more WebLogic Managed Servers, and zero or more System Component Instances (for example, an Oracle HTTP Server instance). A WebLogic Server Domain can span multiple physical machines, and the administration server centrally manages these instances and machines. Because of these properties, a WebLogic Server Domain provides the best integration between your System Components (like OHS) and Java EE Components (like FMW applications). The primary use cases for choosing to administer OHS using Standalone Domain are:
    - You do want OHS to front-end Fusion Middleware domain (like SOA applications)
    - You do need the management and monitoring functionality provided by the browser based Fusion Middleware Control console.
    - You are comfortable with installing WebLogic / JRF libraries and OHS in the same installation location (Collocated).
- **Standalone Domain** - Container for one or more system components, such as Oracle HTTP Server. It has a directory structure similar to an Oracle WebLogic Server Domain, but it does NOT contain an Administration Server or Managed Servers. For standalone domains, the WebLogic Management Framework supports these WebLogic administration tools – Node Manager, WebLogic Scripting Tool (WLST), Config Wizard, and Pack/Unpack. The primary use cases for choosing to administer OHS using Standalone Domain are:
    - You do NOT want OHS to front-end Fusion Middleware domain. However, you can still front-end WebLogic Managed Server(s)/Cluster(s) by manually configuring the WebLogic Plug-In configuration file.
    - You do NOT need the management and monitoring functionality provided by Fusion Middleware Control. However, administrators can still leverage WebLogic Scripting

Tool (WLST) to administer OHS and monitor the OHS performance using the Apache HTTP Server provided tools like 'mod_status' modules.

- You want to provision OHS in a "demilitarized zone" (DMZ; that is, the zone between the internal and external firewalls). Accordingly, you want to minimize the installation footprints and the dependencies and do NOT want to open additional management ports (typically used by the WebLogic Node Manager within the Management Framework).

Managing OHS 12c with WebLogic Server Domain

In this scenario, users can administer the OHS configuration and server lifecycle through either the Fusion Middleware Control console or the WLST.

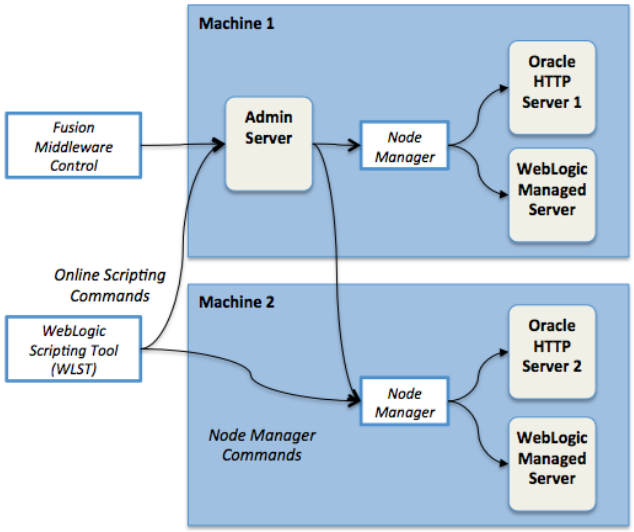

Figure - OHS Administration with WebLogic Server Domain

This scenario is applicable when users choose to install OHS, WebLogic and JRF in the same installation location to be able to manage their Oracle HTTP Server through a browser based administration console (Fusion Middleware Control Console) or through a command line based WebLogic Scripting Tool (WLST).

To implement this scenario, users need to implement the following sequence of steps to administer OHS:

- If your system does NOT have JDK7, then users will need to download and Install Java SE 7 Development Kit in your machine. You can download Java SE 7 from here.
- If your installation location (say Oracle Middleware Home) does NOT include JRF enabled WebLogic 12.1.3, then you can download JRF enabled WebLogic – Fusion Middleware Infrastructure 12.1.3 from here. For more information, please refer to section – Configuring Fusion Middleware Infrastructure domain.
- Install Oracle HTTP Server 12c (12.1.3) in the same location as above WLS + JRF enabled installation location (the same Oracle Middleware Home). For more information on OHS installation, please refer to section – Installing Oracle HTTP Server 12cR1.
- Configure Oracle HTTP Server 12c domain as mentioned within Configuring Fusion Middleware Infrastructure domain. For more information, please refer to section – Installing Oracle HTTP Server 12cR1 (12.1.x) (Collocated)
- Start WebLogic Administration Server and launch Fusion middleware console through http://localhost:<wls-admin-server-port>/em . Users should be now able to administer OHS using the Fusion Middleware Control console. For more information, please see section –.

Managing OHS 12c with Standalone Domain

In this scenario, users can provision Oracle HTTP Server with minimal dependency and administer the server lifecycle using the integrated WebLogic Node Manager interface.

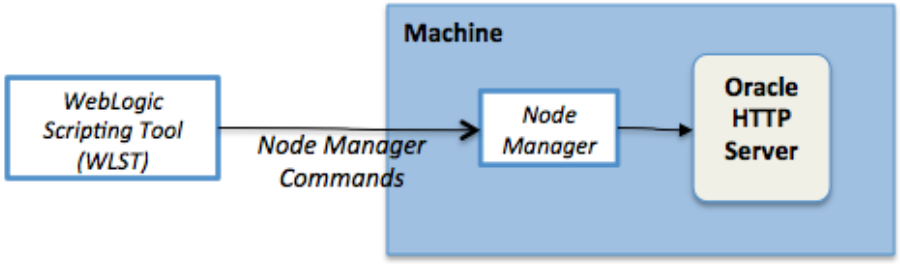

Figure - OHS Administration with Standalone Domain

This scenario is applicable when users (developers and system administrators) like to install OHS with a minimum footprint and dependencies and does not really need to administer OHS (locally or remotely) through a browser based administration console. In addition, system administrators will prefer this scenario, when OHS is installed and provisioned as a proxy in a De-Militarized zone (DMZ).

To implement this scenario, users need to implement the following sequence of steps to administer OHS:

- If your system does NOT have JDK7, then users will need to download and Install Java SE 7 Development Kit in your machine. You can download Java SE 7 from here.
- You can download Oracle HTTP Server 12cR1 binaries from here and install Oracle HTTP Server 12.1.3 (using Standalone as the option) in a given location (say Oracle Middleware Home). For more information on OHS installation, please refer to section – nstalling Oracle HTTP Server 12cR1 (12.1.x) (Standalone)
- Configure Oracle HTTP Server using Fusion Middleware 12.1.3 Config Wizard. For more information, please see section – Configuring OHS 12c with Standalone.
- Start WebLogic Node Manager and leverage WebLogic Administration Framework command line interface (WLST) to provision one or more OHS instances and start / stop these OHS server instances. For more information, please refer to section –.

Configuring OHS 12c with WebLogic Server Domain

This section walks through the steps involved in configuring Oracle HTTP Server 12c within WebLogic Server domain for both development and production scenario so that administrators can leverage the *WebLogic Management Framework* to administer OHS and the rest of the *Fusion Middleware applications*. Users can also visit the Configuring OHS 12cR1 in a WebLogic Server Domain product documentation for more information.

This section assumes that users have successfully completed these steps:

- Installing JRF enabled WebLogic 12.1.3 (aka Fusion Middleware Infrastructure 12.1.3)
- Installing OHS 12.1.3 (as Collocated with JRF enabled WLS 12.1.3)

Once WLS 12.1.3 and OHS 12.1.3 is installed successfully, then users can leverage the WebLogic configurator to setup a domain and a WLS Node Manager to manage one or more OHS instances.

Now, users can configure OHS within *WebLogic Server Domain* through one of these methods:

- Invoke Fusion Middleware Config Wizard 12c and provision WebLogic with 'Expanded Domain' and select Oracle HTTP Server template. For more information, please see Configuring OHS in WebLogic Server Domain section. This option is supported for production deployments and requires a database.
- Use the WebLogic Scripting Tool (WLST) to create a 'OHS Test Domain'. This option does not have any dependency on a database. This option is designed for development and staging environment only. Accordingly, this option is not production certified deployment configuration.

This section will walk through the step-by-step process of configuring OHS in both these scenarios.

Development and Testing Scenario

At the end of installing WebLogic (JRF enabled) Server and Oracle HTTP Server 12.1.3, users will need to provision a WebLogic domain to create and administer a Oracle HTTP Server instance.

For development and testing scenarios, users can leverage the OHS provided command line interface for WebLogic Scripting Tool (WLST) and provision a OHS domain and the corresponding server instance within WebLogic Administration Server. This is done in two steps.

Provision OHS Domain

The first step is to provision the OHS domain within WebLogic.

Users can run the following command through the command line terminal window to provision a OHS domain (as 'Base Domain' within WebLogic Administration Server)

(On Linux)

<ORACLE_MIDDLEWARE_HOME>/ohs/common/bin/wlst.sh

(On Windows)

<ORACLE_MIDDLEWARE_HOME>\ohs\common\bin\wlst.cmd

And run the following commands:

wls:/offline> createOHSTestDomain(adminAccountPass='welcome1', ,nmAccountName='nm', nmAccountPass='welcome1')

wls:/offline> exit()

where –

adminAccountPass -> contains the password to use to login to WebLogic Administration Server

nmAccountName -> contains the user name to use within WebLogic Node Manager

nmAccountPass -> contains the password to use for this WebLogic Node Manager.

This above command configures a WebLogic Administration Server with a Base Domain.

As a next step, we will need to create the OHS instance that can be managed using this WebLogic Administration Server. This step requires the WebLogic Node Manager and Administration Server services, associated within this 'Base Domain', enabled and running.

To do so, users will need to first the start the WebLogic Node Manager and WebLogic Administration Server by running the following commands:

Start Web Logic Node Manager

(On Linux)

<ORACLE_MIDDLEWARE_HOME>/user_projects/domains/base_domain/bin/startNodeManager.sh &

(On Windows)

<ORACLE_MIDDLEWARE_HOME>\user_projects\domains\base_domain\bin\startNodeManager.cmd &

Start Web Logic Administration Server

(On Linux)

<ORACLE_MIDDLEWARE_HOME>/user_projects/domains/base_domain/bin/startWebLogic.sh &

(On Windows)

<ORACLE_MIDDLEWARE_HOME>\user_projects\domains\base_domain\bin\startWebLogic.cmd &

Provision OHS Instance

The second step is to provision the OHS instance. This step requires WebLogic Administration Server and WebLogic Node Manager to be running. For more information on how to start WebLogic Administration Server and Node Manager, refer to the above section.

Once WebLogic Administration Server and the Node Manager are running successfully, users can leverage the OHS provided command line interface to WebLogic Scripting Tool (WLST) to create an OHS instance by running the following command:

(On Linux)

<ORACLE_MIDDLEWARE_HOME>/ohs/common/bin/wlst.sh

(On Windows)

<ORACLE_MIDDLEWARE_HOME>\ohs\common\bin\wlst.cmd

And run the following commands:

wls:/offline> connect()

wls:/base_domain/serverConfig> createOHSInstance(instanceName='ohs1', machine='localmachine', listenPort='7777', sslPort='4443', adminPort='9999')

where

instanceName ->'<ohs-instance-name>' (this creates the OHS instance with this name)

machine -> '<configured-machine-name> (this uses the machine name that was used to create the OHS Test Domain in the previous command)

This command does a basic error checking of instance name, machine name and the port information before configuring the OHS instance.

Here-forth, the OHS instance home location is referred as ***OHS_INSTANCE_HOME*** where this location translates to <ORACLE_MIDDLEWARE_HOME>/user_projects/domains/base_domain

In this above command, '*instanceName*' and '*machine*' are the only required options and the rest are optional. If the other options like '*listenPort*', '*sslPort*' or '*adminPort*' are not provided, then the '*createOHSInstance*' will make a best effort to create the OHS instance using the default ports for these.

If in case, users do NOT remember the associated 'machine' information with this 'Base Domain', then they can find the 'machine name' information used within this domain configuration by running the following commands:

- Look for <machine> element within the <OHS_INSTANCE_HOME>/config/config.xml OR
- Within WLST, run the following:
    - connect()
    - ServerConfig()
    - cd('Machines')
    - ls()
- This command will create OHS instance configurations under
- **<ORACLE_INSTANCE_HOME>/config/fmwconfig/components/OHS/ohs1** -> this acts as a "configuration repository" for WLS Admin Server running on this domain. This "configuration repository" is used to replicate the OHS configuration on more than one machine (clustered deployment).
- **<OHS_INSTANCE_HOME>/config/fmwconfig/components/OHS/instances/ohs1** -> this is where WLS Node Manager manages the configuration on the current machine and is typically created on every machine in a clustered deployment. The OHS HTTPd server processes also use this config directory.

Note: If the WLS Node Manager is unavailable, then the create operation will complete only partially. In this case, only the files under '*<OHS_INSTANCE_HOME>/config/fmwconfig/components/OHS/ohs1'* will be created.

When the WLS Node Manager comes back up online again, then the rest of the files under *<OHS_INSTANCE_HOME>/config/fmwconfig/components/OHS/instances/ohs1* will be created appropriately.

Also, users can now access the Fusion Middleware Control console – a browser based administration console to administer Oracle HTTP Server (OHS) server lifecycle and configurations – by accessing the following URL – http://localhost:7777/em/console , where this URI can be translated as http://<hostname>:<OHS-HTTP-LISTENER-PORT>/em/console , where

<hostname> -> refers to the hostname of the machine where Oracle HTTP Server is installed
<OHS-HTTP-LISTENER-PORT> -> refers to the http listener port. By default, this is set to 7777

For next steps, users can refer to OHS product documentation - Next Steps After Configuring OHS in WebLogic Domain and specifically refer to section - Next Steps for OHS in WebLogic Domain.

Production Scenario

At the end of installing WebLogic (JRF enabled) Server and Oracle HTTP Server 12.1.3, users will need to provision a WebLogic domain to create and administer Oracle HTTP Server instance.

For production scenarios, users will need to leverage the Fusion Middleware Config 12.1.3 Wizard to configure a OHS domain within WebLogic and the corresponding OHS instance within this domain. This step has a dependency on a database.

The step-by-step installation steps are covered within our product documentation - Configuring OHS in a WebLogic Server Domain.

Once OHS domain and the instance has been provisioned, users can now administer the server instance using the WebLogic Management framework.

Also, users can now access the Fusion Middleware Control console – a browser based administration console to administer Oracle HTTP Server (OHS) server lifecycle and configurations – by accessing the following URL – http://localhost:7777/em/console , where this URI can be translated as http://<hostname>:<OHS-HTTP-LISTENER-PORT>/em/console , where
<hostname> -> refers to the hostname of the machine where Oracle HTTP Server is installed
<OHS-HTTP-LISTENER-PORT> -> refers to the http listener port. By default, this is set to 7777

For next steps, users can refer to OHS product documentation - Next Steps After Configuring OHS in WebLogic Domain and specifically refer to section - Next Steps for OHS in WebLogic Domain.

Configuring OHS 12c with Standalone Domain

This section walks through the steps involved in administering Oracle HTTP Server with minimal dependency on the WebLogic Management framework and the accompany database.

This use case is useful for administering OHS when deployed within the DMZ environment.

To administer OHS in this environment, users need to successfully install Oracle HTTP Server 12.1.3 (say under *Oracle Middleware Home*). For more information on installing OHS, refer to section - nstalling Oracle HTTP Server 12cR1 (12.1.x) (.

After installing OHS 12.1.3, users can now configure OHS using the Oracle Fusion Middleware Config Wizard and choose the ‘Standalone’ option.

Users can invoke the Fusion Middleware Config Wizard 12.1.3 as follows

(On Linux)

<ORACLE_MIDDLEWARE_HOME>/oracle_common/common/bin/config.sh

(On Windows)

<ORACLE_MIDDLEWARE_HOME>\oracle_common\common\bin\config.cmd

Note: On Linux machines, the configurator wizard will require a valid X11 display to be available for the GUI to appear.

This wizard provisions Web Logic Node Manager and Oracle HTTP Server (OHS) instance. Web Logic Node Manager will be used to administer the OHS server life cycle.

For more information on how to configure OHS using this Fusion Middleware Config wizard, please refer to documentation - Configuring Oracle HTTP Server 12c in a Standalone Domain

For next steps, users can refer to OHS product documentation - Next Steps for OHS 12c in Standalone Domain.

Administering OHS 12c with WebLogic Server Domain

To administer OHS instances using the WebLogic Management framework, users will need to start the WebLogic Administration Server and the WebLogic Node Manager and leverage either Fusion Middleware Control console or WebLogic Scripting Tool (WLST) to manage the OHS instance lifecycle.

As a first step, users will need to the start the WebLogic Node Manager and Administration Server.

Starting WebLogic Admin Server

This is typically accomplished as follows:

- Set up JAVA_HOME environment variable to reflect the location of JDK/JRE6 or JDK/JRE7 on your machine.
    - JAVA_HOME=<jdk-install>; export JAVA_HOME
- Set up ORACLE_ MIDDLEWARE_HOME environment variable to reflect the installation location where OHS and WLS are collocated together.
    - ORACLE_MIDDLEWARE_HOME=<ohs12.1.3-installation>; export ORACLE_HOME
- Set up JAVA_OPTIONS environment variable to allow ignoring the SSL host name verification
    - JAVA_OPTIONS="-Dweblogic.security.SSL.ignoreHostnameVerification=true"; export JAVA_OPTIONS

Note: On Windows, users will need to appropriately set this environment variable from the command line window 'using the set command' and then start the WebLogic server from the same command line terminal window.

Finally, users can now start the WLS Admin Server as follows:

<OHS_INSTANCE_HOME>/bin/startWebLogic.sh (On Linux)

<OHS_INSTANCE_HOME>\bin\startWebLogic.cmd (On Windows)

Where:

<OHS_INSTANCE_HOME> -> <ORACLE_MIDDLEWARE_HOME>/user_projects/domains/<ohs-domain>

The WebLogic Admin server logs are typically located at <OHS_INSTANCE_HOME>/servers/AdminServer/logs directory.

Note: WLS configurator wizard defaults to '7001' as the default port for running the WLS Admin Server. However, if the configurator wizard determines that this port is being used by another application, then a random port will be allocated and this information should have been available in the configuration summary at the end of the WLS configurator wizard.

Starting WebLogic Node Manager

This is typically accomplished as follows:

- Set up JAVA_HOME environment variable to reflect the location of JDK/JRE6 or JDK/JRE7 on your system.
    - JAVA_HOME=<jdk-install>; export JAVA_HOME
- Set up ORACLE_MIDDLEWARE_HOME environment variable to reflect the installation location where OHS and WLS are collocated together.
    - ORACLE__MIDDLEWARE_HOME=<ohs12.1.3-installation> ; export ORACLE_HOME
- Set up JAVA_OPTIONS environment variable to allow ignoring the SSL host name verification
    - JAVA_OPTIONS="-Dweblogic.security.SSL.ignoreHostnameVerification=true"; export JAVA_OPTIONS

Note: On Windows, users will need to appropriately set this environment variable from the command line window '*set JAVA_HOME etc*' and then start the WebLogic Node Manager from the same command line terminal window.

Finally, users can now start the WLS Node Manager as follows

<OHS_INSTANCE_HOME>/bin/startNodeManager.sh (On Linux)

<OHS_INSTANCE_HOME>\bin\startNodeManager.cmd (On Windows)

Where:

<OHS_INSTANCE_HOME> refers to <ORACLE_MIDDLEWARE_HOME>/user_projects/domains/<ohs-domain>

The WLS Node Manager is typically located at <OHS_INSTANCE_HOME>/nodemanager/nodemanager.log.

Users are hereby advised to refer to these log files to ensure that WLS Admin Server and WLS Node Manager is running successfully.

Note: WLS configurator wizard defaults to 5556 as the default port for running the WLS Node Manager. Users can choose an appropriate port within the WLS configurator wizard.

Starting Oracle HTTP Server Instance 12c

The last step in administering the OHS instance is to start the OHS instance. This is typically accomplished by starting the Oracle HTTP Server directly from a command line—that is, without launching WLST—by entering the following command:

Linux: <ORACLE_MIDDLEWARE_HOME>/user_projects/domains/base_domain/bin/startComponent.sh <componentName>

Windows: <ORACLE_MIDDLEWARE_HOME>\user_projects\domains\base_domain\bin\startComponent.cmd <componentName>

Where
componentName is the name of the OHS instance like 'ohs1' etc.

For more information, please refer to OHS documentation on Performing Basic OHS Tasks.

Administering OHS 12c through WebLogic Scripting Tool

Oracle HTTP Server 12c leverages WebLogic Scripting tool (WLST) – a scriptable command line administration interface – for its administration tasks. This allows users to administer FMW infrastructure components like Oracle HTTP Server and WebLogic Server in a consistent and seamless manner. For more information, please refer to WLST syntax reference. And specifically the custom OHS 12c administration interface within WLST.

The administration capabilities of WLST depend on how Oracle HTTP Server (OHS) is provisioned. This section will focus on the limitations and the steps to accomplish common administration tasks within Oracle HTTP Server 12.1.3 using the WLST command interface.

Oracle HTTP Server 12c (Standalone Domain)

When Oracle HTTP Server is provisioned in standalone domain, WebLogic Administration Server is not available in this configuration. Hence, WebLogic Scripting Tool (WLST) can be used to only monitor and administer OHS server lifecycle operations (like start, stop and soft restart).

Oracle HTTP Server 12c (WebLogic Server Domain)

When Oracle HTTP Server is provisioned in WebLogic Server domain, WebLogic Administration Server is available in this configuration. Hence, WebLogic Scripting Tool (WLST) can be used to perform all the Oracle HTTP Server (OHS) administration tasks including server lifecycle operations (like start, stop and soft restart), managing configurations and monitoring the server instances.

To launch the WLST command for administering OHS server life cycle, users will need to run as follows:

$ORACLE_HOME/oracle_common/common/bin/wlst.sh (On Linux)

$ORACLE_HOME\oracle_common\common\bin\wlst.cmd (On Windows)

And then subsequently connect to 'OHS domain':

connect('<weblogic-admin-username>', '<weblogic-admin-password>','<host:port>')

For example, connect('weblogic', 'welcome1', 'adc6260179:7001')

To successfully manage OHS server instances, users will need to connect to either WebLogic Node Manager (in the case of OHS Standalone domain) or to WebLogic Administration Server (in the case of OHS WebLogic Server domain). This should be the first step in any of the below WLST commands.

Creating OHS 12c Instance

Users can leverage WLST to perform this task only when OHS is provisioned in WebLogic Server Domain.

Within WLST interface, users will need to run the below command to create a new OHS instance:

createOHSInstance(instanceName='ohs1', machine='new_Machine_1', listenPort='7777', sslPort='4443', adminPort='9999')

or

createOHSInstance(instanceName='<ohs-instance-name>',machine='<configured-machine-name>') -> where this command will make the best effort to configure the OHS instance with the default HTTP, HTTPS and Admin port (though not guaranteed).

For example:

<ORACLE_HOME>/oracle_common/common/bin/wlst.sh

wls:/offline> connect('weblogic','welcome1','<host>:7001') -> where connect('<user-name>', '<password>', '<host>:<port>')

wls:/ohs_domain> createOHSInstance(instanceName='ohs1',machine='new_Machine_1') -> where

'ohs1' is the appropriate '<ohs-instance-name>' and

new_Machine_1 is the appropriate name used within the 'Machines' during WLS configuration wizard.

Here, in this above command, instanceName and machine are the required options and the rest are optional.

If the other options like 'listenPort', 'sslPort' or 'adminPort' is not provided, then the 'createOHSInstance' will make a best effort to create the OHS instance using the default ports for these.

If in case, users do not remember the 'machine' information that you used during the WebLogic configuration wizard, then they can find 'machine name' information used within the OHS Domain configuration as follows:

- Look for <machine> element within the <OHS_INSTANCE_HOME>/config/config.xml OR
- Within WLST, run the following:
    - connect()
    - ServerConfig()
    - cd('Machines')
    - ls()

Note: This command does provide basic error checking of instance name, machine name and the port information.

This command will also create OHS configurations under -

- **<OHS_INSTANCE_HOME>/config/fmwconfig/components/OHS/ohs1** -> this acts as a "configuration repository" for WLS Admin Server running on this domain. This "configuration repository" is used to replicate the OHS configuration on more than one machine (clustered deployment).
- **<OHS_INSTANCE_HOME>/config/fmwconfig/components/OHS/instances/ohs1** -> this is where WLS Node Manager manages the configuration on the current machine and is typically created on every machine in a clustered deployment. The OHS HTTPd server process also uses this config directory.

  Note: If the WLS Node Manager is unavailable, then the create operation will complete only partially.

  Accordingly, only the files under '*<OHS_INSTANCE_HOME>/config/fmwconfig/components/OHS/ohs1'* will be created.

  When the WLS Node Manager comes back up online again, then the rest of the files under *<OHS_INSTANCE_HOME>/config/fmwconfig/components/OHS/instances/ohs1* will be created appropriately.

Deleting OHS 12c Instance

Users can leverage WLST to perform this task only when OHS is provisioned in WebLogic Server Domain.

To delete the OHS instance, users will need to run the below command within the WLST:

deleteOHSInstance(instanceName='<ohs-instance-name>')

For example:

On Linux:

<ORACLE_HOME>/oracle_common/common/bin/wlst.sh

On Windows:

<ORACLE_HOME>/oracle_common/common/bin/wlst.cmd

wls:/offline> connect('weblogic','welcome1','<host>:7001') -> where connect('<user-name>', '<password>', '<host>:<port>')

wls:/ohs_domain/ServerConfig> deleteOHSInstance(instanceName='ohs1') -> where 'ohs1' is appropriate '<ohs-instance-name>'

Note: This command requires WLS Node Manager to be running and OHS application to be already stopped.

Starting OHS 12c Instance

Users can leverage WLST to perform this task when OHS is provisioned in WebLogic Server Domain as well as in Standalone domain.

Users can also simply use the 'startComponent.sh' (On Linux) and 'startComponent.cmd' (On Windows) commands to start the OHS instance from the command line. For more information, please refer to section - OHS 12c WLST Administration Interface.

To start a give the OHS instance through WLST, users will need to run the below command within the WLST:

On Linux:

<ORACLE_HOME>/oracle_common/common/bin/wlst.sh

On Windows:

<ORACLE_HOME>/oracle_common/common/bin/wlst.cmd

wls:/offline> connect('weblogic','welcome1','<host>:7001') -> where connect('<user-name>', '<password>', '<host>:<port>')

wls:/ohs_domain/ServerConfig> start('ohs1') -> where start('<ohs-instance-name>')

Also, users can run the 'state(ohs1)' command within the WLST to retrieve the current status of OHS application.

Once OHS instance started successfully,

- OHS log files for the instance will be available at <OHS_INSTANCE_HOME>/servers/ohs1/logs.
- OHS Audit logs are at <OHS_INSTANCE_HOME>/auditlogs/OHS/ohs1
- OHS instance state information will be logged by Node Manager at <OHS_INSTANCE_HOME>/system_components/OHS/ohs1/

Here 'ohs1' is the name of OHS instance used while creating this OHS instance.

Now, users should be able to access this OHS instance through the browser by accessing http://<host>:<http-port>/ or https://<host>:<ssl-port>

Note: If the WLS Node Manager for OHS Domain is not available, then this OHS instance cannot be started until the Node Manager comes online again.

Stopping OHS 12c Instance

Users can leverage WLST to perform this task when OHS is provisioned in WebLogic Server Domain as well as Standalone domain.

Users can simply use the 'startComponent.sh' (On Linux) and 'startComponent.cmd' (On Windows) commands to start the OHS instance from the command line. For more information, please refer to product documentation Stopping OHS Instances.

To stop a give OHS instance within WLST, users will need to run the following command within WLST:

On Linux:

<ORACLE_HOME>/oracle_common/common/bin/wlst.sh

On Windows:

<ORACLE_HOME>/oracle_common/common/bin/wlst.cmd

wls:/offline> connect('weblogic','welcome1','<host>:7001') -> where connect('<user-name>', '<password>', '<host>:<port>')

wls:/ohs_domain/ServerConfig> shutdown('ohs1') -> where shutdown('<ohs-instance-name>')

WARNING: Invoking '*shutdown()*' without any parameter shuts down WLS Admin Server within OHS Domain and also quits WLST shell. However, this does NOT shut down the OHS 'httpd' process.

Note: If the WLS Node Manager for OHS Domain is not available, then this OHS instance cannot be started until the Node Manager comes online again.

Soft Restart of OHS 12c Instance

Users can leverage WLST to perform this task when OHS is provisioned in WebLogic Server Domain as well as Standalone domain.

To perform a soft bounce of OHS instance, users will need to run the following command within WLST:

On Linux:

<ORACLE_HOME>/oracle_common/common/bin/wlst.sh

On Windows:

<ORACLE_HOME>/oracle_common/common/bin/wlst.cmd

wls:/offline> connect('weblogic','welcome1','<host>:7001') -> where connect('<user-name>', '<password>', '<host>:<port>')

wls:/ohs_domain/ServerConfig> softRestart('ohs1') -> where softRestart('<ohs-instance-name>')

Note: If the WLS Node Manager for OHS Domain is not available, then this OHS instance cannot be started until the Node Manager comes online again.

Retrieve OHS 12c Instance Status

Users can leverage WLST to perform this task when OHS is provisioned in WebLogic Server Domain as well as Standalone domain.

To get the status of the OHS instance, users will need to run the below command within the WLST:

On Linux:

<ORACLE_HOME>/oracle_common/common/bin/wlst.sh

On Windows:

<ORACLE_HOME>/oracle_common/common/bin/wlst.cmd

wls:/offline> connect('weblogic','welcome1','<host>:7001') -> where connect('<user-name>', '<password>', '<host>:<port>')

wls:/ohs_domain/ServerConfig> state('ohs1') -> where state('<ohs-instance-name>')

If the HTTPd process is running, then the above command will report as 'RUNNING'.

Retrieve OHS 12c Monitoring Statistics

Users can leverage WLST to perform this task when OHS is provisioned in WebLogic Server Domain as well as Standalone domain.

To retrieve the performance monitoring statistics, administrators can leverage the WLST as follows:

On Linux:

<ORACLE_HOME>/oracle_common/common/bin/wlst.sh

On Windows:

<ORACLE_HOME>/oracle_common/common/bin/wlst.cmd

wls:/offline> connect('weblogic','welcome1','<host>:7001') -> where connect('<user-name>', '<password>', '<host>:<port>')

wls:/base_domain/ServerConfig> displayMetricTables('ohs_*')

Accessing the OHS configuration properties

Users can leverage WLST to perform this task with OHS provisioned in WebLogic Server Domain only.

To retrieve the list of editable properties within the OHS configuration, users can leverage WLST command and run the following commands:

On Linux:

<ORACLE_HOME>/oracle_common/common/bin/wlst.sh

On Windows:

<ORACLE_HOME>/oracle_common/common/bin/wlst.cmd

wls:/offline> connect('weblogic','welcome1','<host>:7001') -> where connect('<user-name>', '<password>', '<host>:<port>')

wls:/ohs_domain/ServerConfig> editCustom()

wls:/ohs_domain/ServerConfig> cd('oracle.ohs')

wls:/ohs_domain/ServerConfig> cd('oracle.ohs:type=OHSInstance,name=ohs1')

wls:/ohs_domain/ServerConfig> ls()

Appendix

Installing WebLogic JRF 12cR1

To manage OHS 12c through WebLogic Management Framework, users will need to install JRF enabled WebLogic and OHS in the same installation location (say Oracle Middleware Home). This section will walk through the steps involved in installing JRF enabled WebLogic 12.1.x

- You can download JRF enabled WebLogic 12.1.3 from here.

- Users are advised to set JDK/JRE7 in their JAVA_HOME environment and begin installing WLS 12.1.3 by running the following command:
    - $JAVA_HOME/bin/java –jar wls_jrf_generic.jar
- Choose the appropriate options to specify the inventory directory. The default option will work fine as well.
- In the Next screen, select 'Skip Software Updates' and proceed to finish the pre-requisite checks.
- In the "Oracle Home" installation location, **choose a new installation location that will host both WebLogic 12.1.3 server and OHS 12.1.3 server(s) and click on 'Next'.**
- Users can safely choose to not get the security updates and allow the installer to start laying out the binaries

  For more information, refer to product documentation - Installing WebLogic 12.1.3

Installing Oracle HTTP Server 12cR1 (12.1.x) (Collocated)

This section will walk through the steps involved in installing Oracle HTTP Server (OHS) 12cR1 in a collocated (same installation location) with JRF enabled WebLogic.

- Users will need to complete WebLogic JRF enabled (Fusion Middleware Infrastructure) in a specific Oracle Home as captured in section - Installing WebLogic JRF 12cR1
- Users can download Oracle HTTP Server 12cR1 either from E-Delivery (recommended for production) or from Oracle Technology Network (for evaluation and development) and install in the same Oracle Home where you installed WebLogic JRF (Fusion Middleware Infrastructure).
- Users can install OHS 12.1.3 by running the following command:
    - ohs_linux64.bin (from a terminal on Linux 64-bit)
    - setup_ohs_win64.exe (from a command line window on Windows 64-bit)
- Choose the default options for the inventory directory.
- In the Next screen, select 'Skip Software Updates' and proceed to finish the pre-requisite checks.
- In the 'Next' screen, choose for the '**OHS managed through a pre-installed WebLogic server**' option.

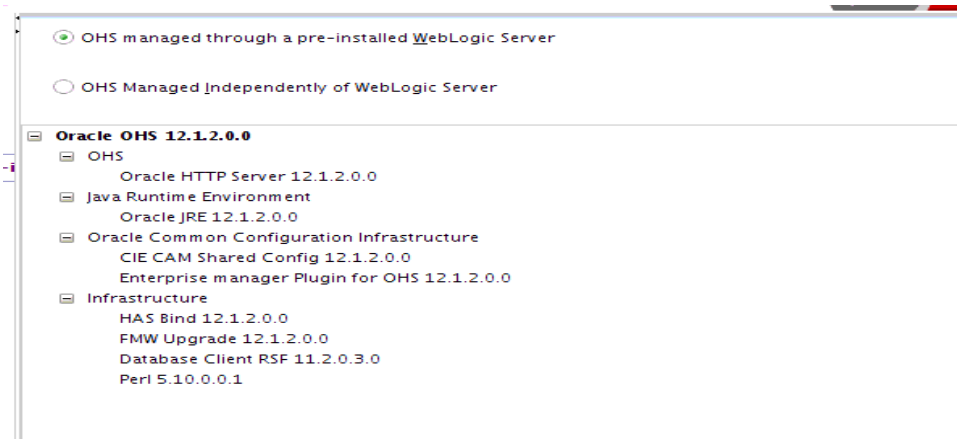

Figure - Installing OHS Collocated

- In the "Oracle Home" installation location, choose the same installation location that was used to install WebLogic 12.1.3 earlier and click on 'Next'.
- Now, you can safely choose to not get the security updates and allow the installer to start laying out the binaries.

  This installation location will here forth be simply referred as the 'ORACLE_MIDDLEWARE_HOME'.

  For more information, please refer to product documentation Installing Oracle HTTP Server 12cR1

Installing Oracle HTTP Server 12cR1 (12.1.x) (Standalone)

This section will walk through the steps involved in installing Oracle HTTP Server (OHS) 12.1.3 in a standalone scenario.

- Users can download Oracle HTTP Server 12cR1 either from E-Delivery (recommended for production) or from Oracle Technology Network (for evaluation and development)
- Users can install OHS 12.1.3 by running the following command:
    - ohs_linux64.bin (from a terminal on Linux 64-bit)
    - setup_ohs_win64.exe (from a command line window on Windows 64-bit)
- Choose the default options for the inventory directory.
- In the Next screen, select 'Skip Software Updates' and proceed to finish the pre-requisite checks.
- In the 'Next' screen, choose for the 'OHS standalone' option.

  For more information, please refer to product documentation Installing Oracle HTTP Server 12cR1

Conclusion

Oracle HTTP Server 12cR1 - 12.1.x - is based on the popular Apache HTTP Server infrastructure and can reliably serve the '*Web Tier*' responsibilities by reliably and securely front-ending the HTTP(s) traffic to the applications hosted within the Fusion Middleware. The following capabilities distinguish Oracle HTTP Server 12.1.3 from the rest of the non-Oracle Web Tier offerings:

- Integrated management and monitoring capabilities consistent with the rest of the Fusion Middleware Applications.
- Multiple provisioning models – *WebLogic Server Domain* for integrated administration and *Standalone Domain* for lean and simplified provisioning with remote management capability.
- Ability to administer a farm of Oracle HTTP Server running on multiple machines through the WebLogic Domain

These capabilities allow Oracle HTTP Server to serve the static content and reverse proxy the HTTP(s) requests to the WebLogic Managed Server(s)/Cluster(s). In addition, customers can also administer the '*Web Tier*' and the '*Application Tier*' within the WebLogic domain thereby simplifying the common administration tasks like provisioning and server lifecycle administration.

White Paper Title
July 2013
Author: Sriram Natarajan
Contributing Authors: Frances Zhao

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

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200

oracle.com

Oracle is committed to developing practices and products that help protect the environment