

Oracle Health Insurance Back Office

HTTP Service Layer (HSL) Installation & Configuration Manual

Version 1.15

Part number: F13010-01

January 22, 2019

Copyright © 2016, 2019, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS

Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are “commercial computer software” or “commercial technical data” pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

This software and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Where an Oracle offering includes third party content or software, we may be required to include related notices. For information on third party notices and the software and related documentation in connection with which they need to be included, please contact the attorney from the Development and Strategic Initiatives Legal Group that supports the development team for the Oracle offering. Contact information can be found on the Attorney Contact Chart.

The information contained in this document is for informational sharing purposes only and should be considered in your capacity as a customer advisory board member or pursuant to your beta trial agreement only. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described in this document remains at the sole discretion of Oracle.

This document in any form, software or printed matter, contains proprietary information that is the exclusive property of Oracle. Your access to and use of this confidential material is subject to the terms and conditions of your Oracle Software License and Service Agreement, which has been executed and with which you agree to comply. This document and information contained herein may not be disclosed, copied, reproduced, or distributed to anyone outside Oracle without prior written consent of Oracle. This document is not part of your license agreement nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.

CHANGE HISTORY

Release	Version	Changes
10.16.2.2.0	1.0	<ul style="list-style-type: none"> • Creation
10.16.2.2.0	1.1	<ul style="list-style-type: none"> • Revision
10.17.1.0.0	1.2	<ul style="list-style-type: none"> • Changed grant instructions
10.17.2.0.0	1.3	<ul style="list-style-type: none"> • Documented hsl.<app>.developermode and hsl.developermode • Added reference to Doc[2] (Back Office HTTP Service Layer User Manual)
10.17.2.1.0	1.4	<ul style="list-style-type: none"> • Extended set of relevant properties.
10.17.2.2.0	1.5	<ul style="list-style-type: none"> • Minor revision of 'Creating a HSL database account' • Revised 'Security Configuration' • Removed 'Restricting access with custom roles' from Security Aspects • Renamed 'Security Aspects' to 'Additional Security Aspects' and revised contents. • Added 'Deployment Validation' • Added 'Appendix C - Testing with SoapUI' • Added 'Appendix D - Generating a WADL file' • Extended set of relevant properties for HSL_CLA.war deployment.
10.17.2.3.0	1.6	<ul style="list-style-type: none"> • Added paragraph 'Examining the Log File' • Added JDK version specific information regarding JSSE configuration.
10.18.1.0.0	1.7	<ul style="list-style-type: none"> • Added Appendix E - Authentication and Authorization • Added Appendix F - HSL_AUN and HSL_AUZ Services • Revised 2.1 including diagram • Revised introduction and document title
10.18.1.2.0	1.8	<ul style="list-style-type: none"> • Added Appendix G - PSL services • Use setUserOverrides.sh instead of modifying startManagedWebLogic.sh and Server Start arguments. Support for WLS 12.2.1.3.
10.18.1.3.0	1.9	<ul style="list-style-type: none"> • Added warning about patch 28278427
10.18.1.3.0	1.10	<ul style="list-style-type: none"> • Revised installation of PSL services
10.18.1.4.0	1.11	<ul style="list-style-type: none"> • Updated 'Additional Security Aspects' • Mentioned use of Basic Authentication for 'admin' requests. • Updated 'setting user context' in Appendix E • Added HTTP codes to 'Troubleshooting'
10.18.1.4.0	1.12	<ul style="list-style-type: none"> • Updated HTTP codes overview in 'Troubleshooting' • Revised Appendix G - PSL services
10.18.2.0.0	1.13	<ul style="list-style-type: none"> • Corrected typo: hsl.auz, not hsl.aur. • Rewrote explanation of setUserOverrides.sh
10.18.2.0.0	1.14	<ul style="list-style-type: none"> • Services Components: added HSL_AUN and HSL_AUZ • WLS Preparation: fixed typos • Additional Security Aspects: removed remarks about patch 28278427 (already covered by certification matrix)
10.18.2.1.0	1.15	<ul style="list-style-type: none"> • Added example properties for new PSL service with shortname zvp • Added description of allowedorigins property to protect against cross site scripting attack

RELATED DOCUMENTS

A reference in the text (**doc[x]**) is a reference to another document about a subject that is related to this document.

Below is a list of related documents:

Doc[1] Object Authorisation within OHI Back Office (CTA 13533)

Doc[2] Back Office HTTP Service Layer User Manual (CDO 15195)

Contents

1	Introduction.....	8
1.1	Licenses.....	8
2	Architectural overview	9
2.1	Services components	9
3	Installation of HSL services	11
3.1.1	Terminology	11
3.2	Sizing/load aspects	11
3.2.1	Deployment choices	12
3.3	Database installation	12
3.3.1	Creating a HSL database account	12
3.4	WLS Preparation.....	13
	Using setUserOverrides.sh.....	14
3.4.1	Requirements.....	15
3.4.2	Creating a domain	15
3.4.3	Creating Managed Server(s).....	18
3.4.4	Creating a machine definition.....	19
3.4.5	Creating a data source.....	21
3.5	Security Configuration.....	26
3.5.1	Set up security realm.....	26
3.5.2	Setup Weblogic user for accessing HSL application	27
3.5.3	Enable SSL	28
3.5.4	Configure JSSE	29
..3.5.4.1	JDK 1.8.0_162 and above	29
..3.5.4.2	JDK 1.8.0_151 .. 1.8.0_161.....	29
..3.5.4.3	Below JDK 1.8.0_151.....	30
3.5.5	Setting up a key store	30
3.5.6	Configure Managed Server logging level.....	31
3.5.7	Set user lockout	32
3.6	(Re)deployment of the HSL Application.....	33
3.6.1	Deploy to a single Managed Server	33
..3.6.1.1	Deploy WAR files	33
..3.6.1.2	Specify configuration file.....	35
3.6.2	Deploy to multiple Managed Servers	35
3.6.3	Deploy to cluster	36
3.6.4	Deploy for multiple environments (DTAP)	36
3.6.5	Validate deployment.....	36
3.7	Additional Security Aspects.....	37
3.7.1	Deploying HSL Application for use with any weblogic user	37
3.7.2	Using a custom security policy for a deployed application	38
3.7.3	Cross Site scripting protection	38
4	Deployment validation	39
4.1	Testing with Curl.....	39
4.2	Template Listing.....	40
4.3	getDatabaseInfo	40
4.4	Get Online Swagger definition	41
4.4.1	Saving the Swagger definition to a file	42
4.4.2	Viewing the Swagger definition	42

4.5	Troubleshooting.....	43
5	Configuration Files for HSL services.....	45
5.1	Back Office HSL properties file	45
5.1.1	hsl.jndiname	45
5.1.2	hsl.<app>.jndiname.....	45
5.1.3	hsl.usercontext.....	46
5.1.4	hsl.<app>.usercontext	46
5.1.5	hsl.developermode	46
5.1.6	hsl.<app>.developermode.....	47
5.1.7	hsl.<app>.logfile	47
5.1.8	hsl.<app>.loglevel	47
5.1.9	hsl.<app>.log.limit.....	47
5.1.10	hsl.<app>.log.count.....	48
5.1.11	hsl.<app>.log.append	48
5.1.12	Activating changes to hsl.properties	48
5.1.13	Troubleshooting hsl.properties.....	48
5.1.14	Example hsl.properties file	48
5.1.15	Keeping hsl.properties up to date	49
5.2	Examining the Log File	49
5.2.1	Changing the log format.....	50
6	Upgrading HSL services	53
7	Appendix A - Service Information.....	55
8	Appendix B - Removing a WLS domain.....	56
9	Appendix C - Testing with SoapUI.....	57
9.1	Create REST project and import Swagger definiton.....	57
9.2	Create a request	57
10	Appendix D - Generating a WADL file	59
10.1	Create a REST project in SoapUI for your HSL application	59
10.2	Open the Service Viewer for the REST Project	59
10.3	Export WADL from your REST project.....	59
11	Appendix E - Authentication and Authorization	61
11.1	HTTP OPTIONS method	61
11.2	OAuth 2.0 token authentication and validation	61
11.2.1	WAR File Deployment	61
11.2.2	Configuration.....	62
11.2.3	Which Authorization Method?	62
11.2.4	Access Token Validation.....	62
11.2.5	Place Holders.....	63
11.2.6	POST Example.....	63
11.2.7	GET example	63
11.2.8	Setting user context.....	64
11.2.9	Overriding User Context with Back Office Parameter	64
12	Appendix F - HSL_AUN and HSL_AUZ Services.....	65
12.1.1	Disclaimer	65
12.2	Use of JWT	65
12.2.1	Payload.....	65
12.2.2	Token Verification	66
12.3	HSL Properties for Signature Encryption.....	66
12.4	HSL_AUN Authentication Service.....	66
12.4.1	HSL properties	66
12.5	HSL_AUZ Authorization Service.....	66
12.5.1	postVerify operation.....	66
13	Appendix G - PSL services	67
13.1	Installation of PSL services.....	67
13.1.1	PSL database account.....	67

13.1.2	WLS Domain	68
13.1.3	Data source	68
13.1.4	WLS Managed Server Start arguments.....	68
13.2	Configuration of PSL.properties.....	68

1 Introduction

The OHI Back Office HTTP Service Layer is an optional component to provide so-called Use Case services.

Use Case services constitute a group of specific operations aiming to support use cases that are common for Dutch healthcare payers. Examples of typical use cases: requesting a new policy, adding an insured member, changing insured products, changing payment method etc.

OHI BO Use Case Services are implemented through the HTTP Service Layer (HSL).

The services in this service layer are based on RESTful Services technology which has the following advantages for current web application frameworks (like AngularJS and Oracle JET):

- accessible through HTTP (for example through Javascript)
- Supports (Javascript friendly) JSON as input and output formats
- standardized interface language through using HTTP verbs (GET, POST, PUT, PATCH, DELETE)
- standardized set of exceptions through HTTP error codes

This HTTP Service Layer is intended to ease integration in a Service Oriented environment.

This document describes the generic technical details regarding the HTTP Service Layer, how to install and update it and how to change configuration settings.

1.1 Licenses

Customers are required to have the appropriate license for using the HTTP Service Layer. Customers who have acquired a Connect to Back Office (C2B) license or an OHI SOAP ServiceLayer (SVL) license are currently permitted to install and use the web service component of the HTTP Service Layer. This is valid until further notice.

The corresponding PL/SQL services may not be used when no Connect to Back Office license, SOAP Service Layer license or HTTP Service Layer license has been obtained.

For further information please consult your OHI sales representative.

2 Architectural overview

This chapter gives a high level architectural overview of the current HTTP Service Layer implementation.

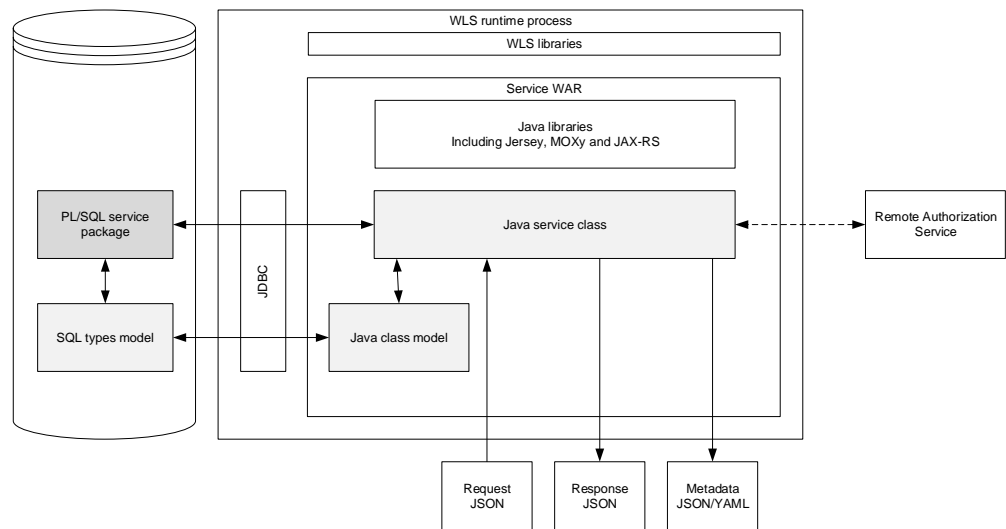
2.1 Services components

The functionality of each service is implemented through a PL/SQL service package. The service interface is provided through a Java layer.

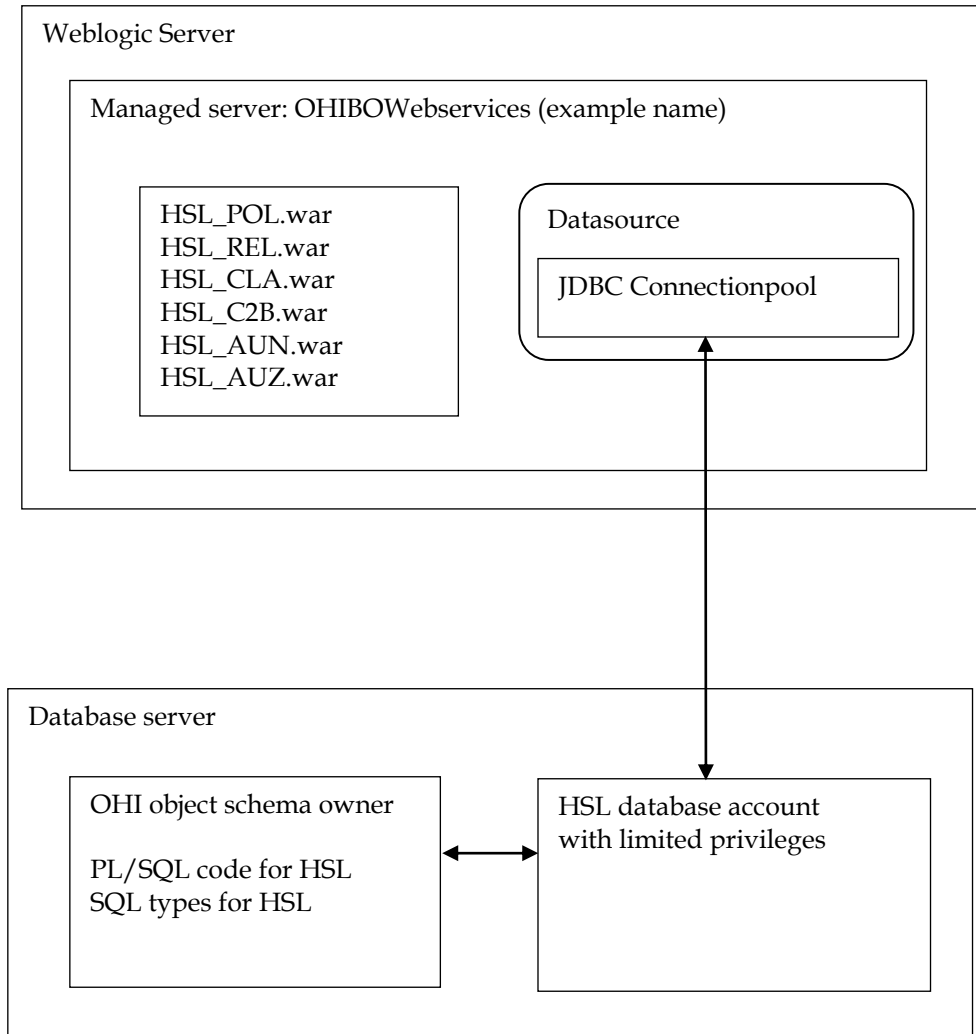
Jersey, JAX-RS and MOXy are used to serialize and deserialize JSON objects and for input validation.

JDBC is used to map Java objects to SQL objects and vice versa.

The PL/SQL service package performs the required operations, using operation parameters and inbound objects to communicate with the OHI Back Office database.



The high level schema below shows how the services are deployed. It also shows the database connection to OHI which uses a database account with restricted access to execute the HSL service implementation in PL/SQL.



3 Installation of HSL services

This chapter describes the steps to (re)install the HSL services.

This chapter contains the following parts to separate the various work areas:

- Sizing/load aspects
- Database installation
- WLS preparation
- Security configuration
- (Re)deployment of the HSL application
- Additional Security aspects

3.1.1 Terminology

Note the following use of terminology:

- HSL stands for HTTP Service Layer. The underlying technology is based on RESTful service technology.
- A HSL service has one or more service operations.
- Each HSL service resides in its own HSL application.
- A HSL application is packaged as a WAR file, which is deployed to the WebLogic application server.

3.2 Sizing/load aspects

From the “Introduction” and the “Architectural overview” chapters it should be clear that the HSL services are implemented through PL/SQL in the database.

The Java layer providing the REST interface handles request and response messages. It validates an incoming request, calls the PL/SQL service implementation to perform the required operation and transforms the result into a response message.

This choice means that the larger part of the processing is carried out on the database server and only a small part is handled on the application server. Since the architecture for HSL is similar to the SVL services, the distribution of loads on the application and database server is expected to be comparable. Based on the SVL services it may be assumed that for heavy processing only 1 CPU thread will be busy processing HSL service requests if 10 CPU threads are needed for the database processing for these requests.

Based on SVL experience, most of the simpler service operations on a well-sized and well-performing production environment should not take more than 0.1 up to 0.5 second in total elapsed time when measured on the WebLogic Server. Of this elapsed time most of the time should be spent by the database server handling the call, as mentioned before.

More complicated calls and service calls that return large data sets may take more time but usually should not exceed response times of more than a few seconds. As an example, if it would be offered, a typical premium calculation call should be executed

within a second and a large set of claim lines (several hundreds) should usually be returned within 5 to 10 seconds.

3.2.1 Deployment choices

The overall load on the OHI application resulting from HSL service calls is customer specific and may change over time.

HSL services are likely to be used by customer-facing applications. Although it may technically be possible to deploy HSL services to the application server running the Forms GUI for internal users, you should be aware of the peak loads from HSL services during commercial campaigns. These loads may well exceed your normal capacity. You should devise your own strategy to cope with these extra loads. This strategy may include using separate application servers for internet users, using a separate database with cached data for information requests, throttling inbound requests, etc.

If you choose to install HSL services on the application server for the Forms GUI it is advisable to actively monitor the respective loads of Forms processes, SVL processes and HSL processes. This allows you to pick up trends to help you refine your infrastructure strategy.

Especially if you have multiple applications using the same HSL services, it may help to use a service bus to create a 'separation of concerns'. The service bus allows you to map the HSL interface specification to a customer-specific interface which means less maintenance on the client applications when deploying a new version of a HSL service. As long as the mapping on the service bus is synchronized with the HSL service interface, the code client applications can remain the same.

Stringent requirements for high availability and failover are also reasons to consider a service bus as a go-between.

3.3 Database installation

All database components of the HTTP Service Layer are owned by the OHI Back Office schema and are installed through the OHI Back Office release installation procedure.

To use the database components of the HTTP service layer, one or more database accounts must be created with HTTP Service Layer access privileges.

Before creating the account(s), check if you are licensed to use the HTTP Service Layer.

Please check if you have a database object (package) HSL_UTIL_PCK in the OHI Back Office schema. If not, something went wrong regarding the installation of the HTTP Service Layer code. If this is incorrect please contact the OHI Support department.

If the package is present in your database you can continue with the database part of the installation.

3.3.1 Creating a HSL database account

The OHI Back Office schema owns the PL/SQL code to implement the HTTP Service Layer but may not be used to execute the services.

The use of a separate database account to access the HTTP Service Layer components reduces the risk of accessing unauthorized OHI data and makes that account accountable for HSL actions. The HSL account(s) need a minimum of object privileges to the HSL database objects.

One or more HSL accounts can be created:

- Default HSL account for use with WebLogic application server
This account is configured in the HSL properties file as the default account for HSL service requests.
- Optional additional HSL accounts for use with WebLogic application server.
These may be configured in the HSL properties file for one or more specific services.
- Additional HSL account for use with bespoke PL/SQL code development by the customer. Please follow the directions in [Doc\[1\]](#).

The following steps are needed to setup a HTTP Service Layer database account:

1. Create a schema owner, for example HSL_USER. Determine the password policy, temporary tablespace, etc. according to your company standards but beware there is no interactive login which might show expiration messages for the password due to the enforced password policy.
2. Grant create session system privilege to this account.
3. Grant the HTTP Service Layer object privileges: logon as the OHI Back Office schema owner, enable server output, and run

```
alg_security_pck.HSL_grants
(pi_owner    => '<ohibo_owner>'
,pi_grantee => '<hsl_user_account>'
)
```

Example:

```
execute
alg_security_pck.HSL_grants
(pi_owner    => 'OZG_OWNER'
,pi_grantee => 'HSL_USER');
```

IMPORTANT: This command needs be run only once. While installing subsequent OHI BO updates, the privileges of the HSL user accounts are automatically updated.

However, if you run into ORA-01403 errors during an execution your first check should be to run this command in sqlplus, enabling server output before running, and see whether missing grant privileges were granted.

3.4 WLS Preparation

When the database account has been created and granted successfully, a WebLogic Server environment (software home) must be prepared for deploying the HSL application.

We expect that you are familiar with the WebLogic concepts like 'Domain', 'Managed Server', 'Cluster', etc.

These are your options:

- Use the same WebLogic environment which is used for servicing the OHI Back Office user interface and batches. In this case you should create a new WebLogic domain (with a new Admin Server) for the HSL applications to prevent interference with the GUI application.

- Deploy the HSL applications in a separate WebLogic environment (possibly on a separate server). This allows you to separately upgrade or patch the different WebLogic environments, or implement a workload distribution.

Deploy HSL applications to multiple environments for better scalability. Be sure to deploy each HSL application only once in a Managed Server or a cluster of Managed Servers.

- For testing purposes you may want to have multiple versions within the same domain. In that case you should have a separate Managed Server for each deployment.

Some remarks about installing in a separate WebLogic environment:

- The OHI Back Office GUI application (Forms) installation requires a WebLogic Server “Infrastructure” installation. That means the domain created for Forms needs to have its own database schemas with OPSS and Audit database tables (created by RCU). For the HTTP Service Layer domain these schemas are not required provided you do not select more components during the domain configuration than described.
- When installing in a separate WebLogic Server environment, use a different Installer: use the “Generic” installer instead of the “FMW Infrastructure” installer. When installing in a separate WebLogic environment make sure the correct components are installed when creating the Domain. You need at least:

- o Weblogic Advanced Web Services for JAX-WS Extension - 12.2.1.x.0 [oracle_common]
- o Weblogic JAX-WS SOAP/JMS Extension - 12.2.1.x.0 [oracle_common]

where x is 2 or 3, depending on your WebLogic version.

When you have not installed these components your web services will respond with ‘There are error messages.’ All info in the functionalFaultType will contain question marks (???)

The instructions in the following paragraphs cover the setup of a new domain including the setting up of Managed Servers, a machine definition, data sources, etc.

This will support the following scenarios:

- ✓ Creating a separate domain with a single Managed Server
- ✓ Creating a separate domain with a cluster of 2 Managed Servers
- ✓ Adding a Managed Server to an existing domain

Using setUserOverrides.sh

Traditionally, Server Start arguments for the WebLogic Managed Server had to be added in multiple locations (depending on the way the Managed Servers were started):

- Via the Admin console: in the tab “Server Start” in the field “Arguments” for Admin Servers and Managed Servers
- In the script \$DOMAIN_HOME/bin/startManagedWebLogic.sh for Managed Servers

- In script `$DOMAIN_HOME/bin/startWebLogic.sh` for Admin Servers

WebLogic 12.1.2 introduced a better way to pass Server Start arguments to the WebLogic servers. See document “How To Customize Env Parameters Via 'setUserOverrides.sh' File (In WLS 12.1.2.0.0 ~ 12.2.1.3.0) (Doc ID 2138183.1)” on My Oracle Support for details.

By using `setUserOverrides.sh` the Server Start arguments only need to be maintained in one place. This document will assume you use this new method and it will only give instructions for setting the Server Start arguments using this method. The file is located here:

```
$DOMAIN_HOME/bin/setUserOverrides.sh
```

3.4.1 Requirements

The following requirements/limitations must be taken into account:

- ✓ A certified WebLogic Server version including JAX-WS (SOAP/JMS) extensions. The HSL services must be deployed on a single Managed Server or a cluster of Managed Servers (the ‘target’).
- ✓ The HSL services may not be deployed on a Managed Server which is also used for hosting the OHI GUI application (Forms). The Managed Server may not belong to a cluster used for deploying the GUI application.
- ✓ One deployment can only service one single OHI Back Office environment (it connects to a specific connection pool which accesses a specific OHI Back Office ‘instance’).

If the HSL application must be deployed more than once (for servicing different OHI Back Office environments) each deployment should be on its own Managed Server or Cluster.

HSL can be deployed on the same Managed Servers as SVL.

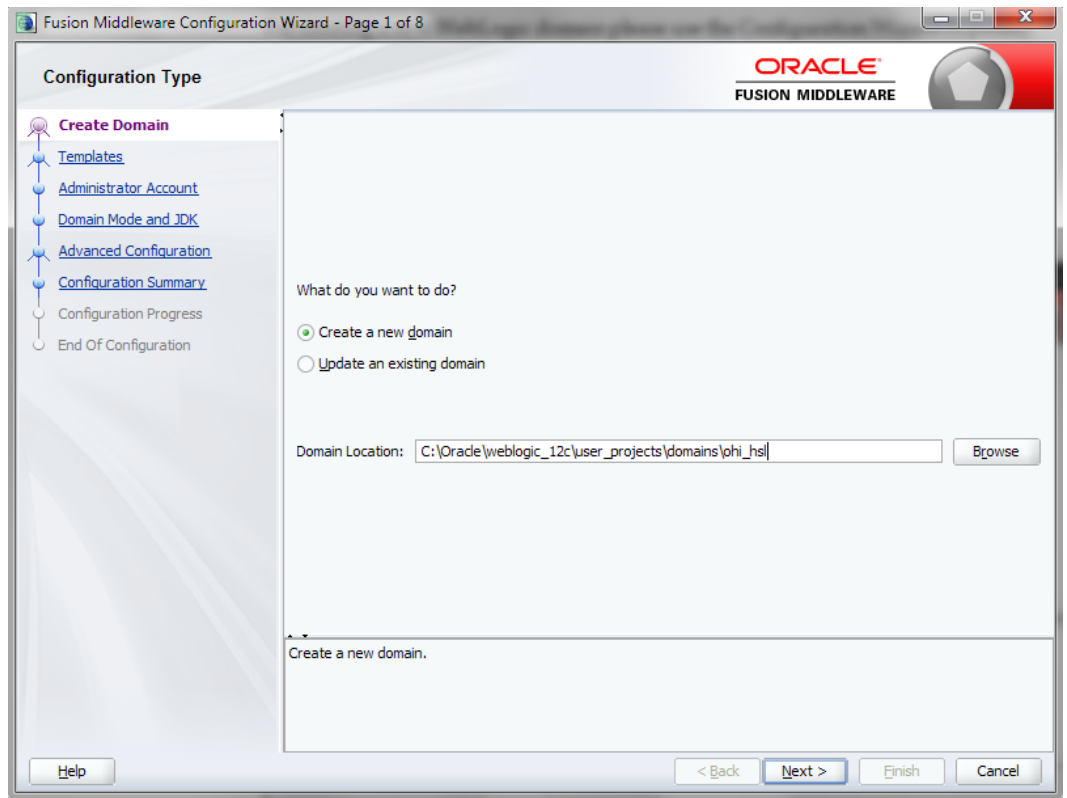
3.4.2 Creating a domain

Before creating a Domain, be sure to understand the difference between a “FMW Infrastructure” and a “Generic” WebLogic installation, and the consequences. Make sure the environment variable `DOMAIN_HOME` is not set.

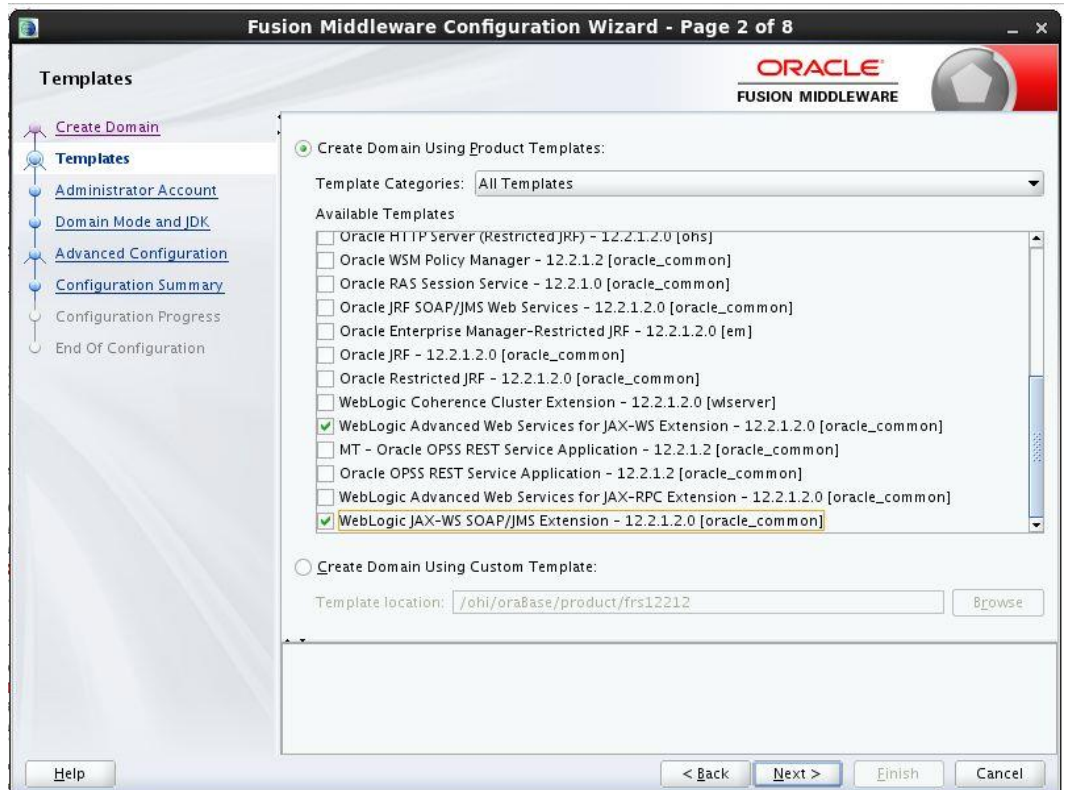
If you create the new Weblogic Domain from the same software home as the Forms Domain, you have to choose the same “Domain Mode” (Development or Production), to avoid errors during startup of the new Managed Server(s).

For creating a new WebLogic domain please use the Configuration Wizard (typically in the `common/bin` folder of the WebLogic Server home, so for example `$MW_HOME/oracle_common/common/bin/config.sh`)

Specify the domain location. This is inside the Weblogic Home by default, but you can specify a location outside the WebLogic Home. The last part of the location will be the Domain Name.



When creating a new domain select at least the options as shown below.

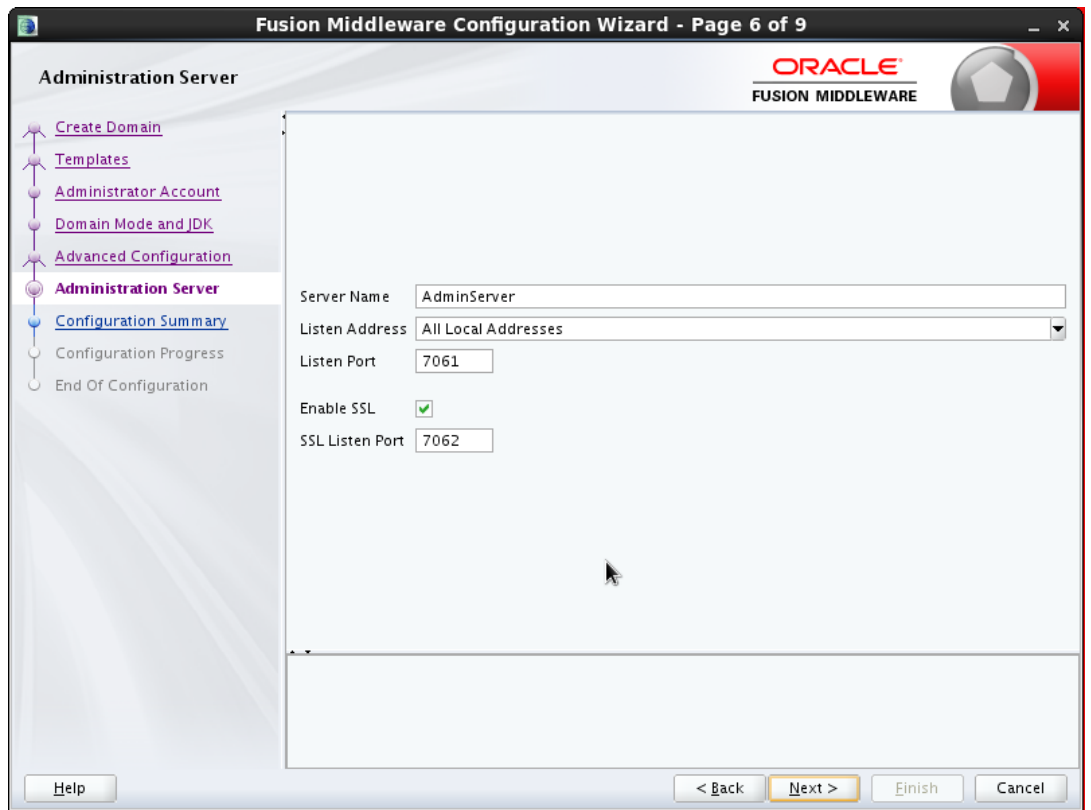


In the next screens, specify the *username* and *password* for the domain administrator account. When prompted for developer or production mode choose *production mode* and pick a JDK.

In this documentation we choose to configure only the Administration Server using the wizard. The Administration Server can be used as the starting point for additional configuration options you may want to choose later:



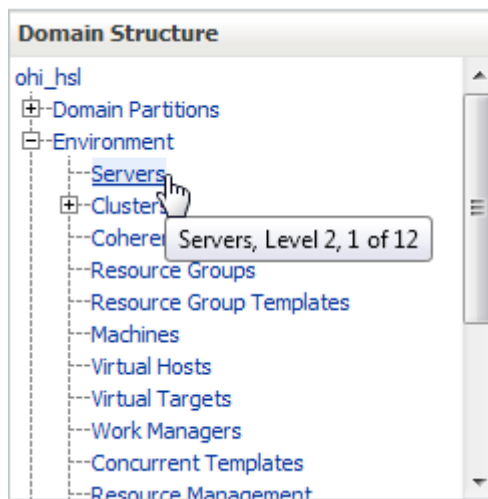
For the Administration Server a free port number must be specified. Enable SSL to support secure connections. An example using non default ports is shown below.



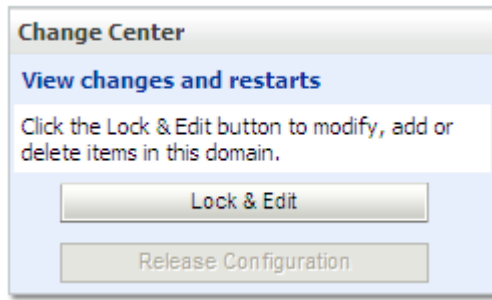
3.4.3 Creating Managed Server(s)

Start the Administration Server (of the existing or newly created domain) using the startWebLogic.sh script (this is present in the root folder of the domain folder, which you created through the Configuration Wizard).

After it has started, logon to the console and choose the Servers option in the left panel:



In the Change Center choose Lock & Edit to get into editing mode.



This enables the New option in the 'Summary of Servers' overview:



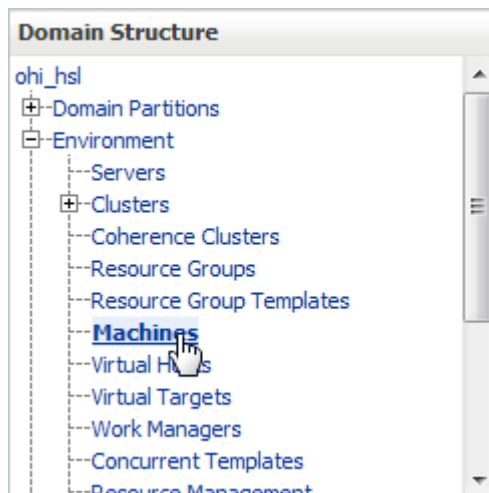
You need to provide a name and listening port for the Managed Server. For easy reference you may want to include the domain name in the name of the Managed Server, for example 'ms_ohi_hsl'.

At this point you should decide whether or not to make the Managed Server part of a Cluster.

If no Cluster exists you can create one; if there is an existing Cluster you can make the Managed Server a member of the Cluster.

3.4.4 Creating a machine definition

It is recommended to create a machine definition to make it easier to start up Managed Servers:



You can now assign Managed Servers to the new machine definition. In the example below Managed Server ms_ohi_hsl is assigned to Machine1.

<input type="checkbox"/>	ms_ohi_hsl	Configured		Machine1
--------------------------	------------	------------	--	----------

If you start a Node Manager you can use the console to start the Managed Servers.

You need to associate the machine with the Node Manager so that the Node Manager can start the Managed Server within the domain of the machine definition.

Do this in the Node Manager tab for the machine definition like in the example below:

Settings for Machine1


Configuration Monitoring Notes

General **Node Manager** Servers

Save


This page allows you to define the Node Manager configuration for this machine. To control a Managed Server from Node Manager must be configured and running on the machine where the Managed Servers are installed.

The settings defined on this page are used to configure communication between the current domain and Node Manager that control Managed Servers. This page does not control the configuration of the Node Manager instances.

 **Type:** Returns the node manager type.

Listen Address: The host name or IP address where Node Manager listens for connection requests. [Info...](#)

Listen Port: The port number where Node Manager listens for connection requests. [More Info...](#)

 **Node Manager Home:** Returns the node manager home directory that will be used to substitute for %HOME% in the command template. [More Info...](#)

Make sure the listen address is the actual listen address that is used by the Node Manager. This is passed as first parameter to the `$WL_HOME/server/bin/startNodeManager.sh` shell script. The correct value can be found as ListenAddress in the file `nodemanager.properties`.

This address can be changed in the file `nodemanager.properties` which is located in the `<domain home>/nodemanager` folder. This is necessary when you have a node manager per domain.

You need to create a `boot.properties` file for the new Managed Server for the domain in the domain home Managed Server `../data/nodemanager`.

This is done automatically when you start the Managed Server in the console (after you have started the AdminServer for the domain).

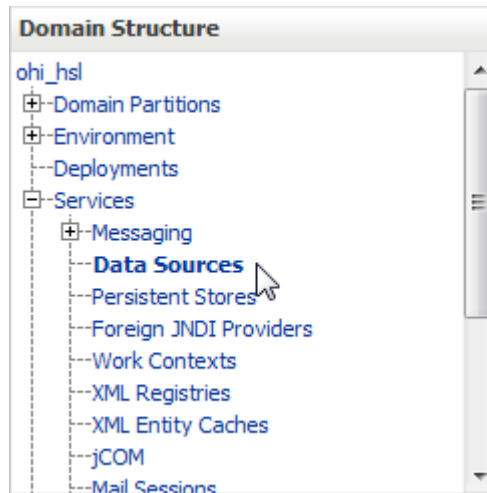
When you are running in Development Mode, a `boot.properties` file is automatically created for the AdminServer.

Because you are running in Production Mode, you need to create the file yourself, in the `$DOMAIN_HOME/servers/AdminServer/security` folder. This file is used when the AdminServer is started by the script `startWebLogic.sh`. If the file is not present, the script prompts for the username/password. The same goes for the Managed Servers when you start them through a script.

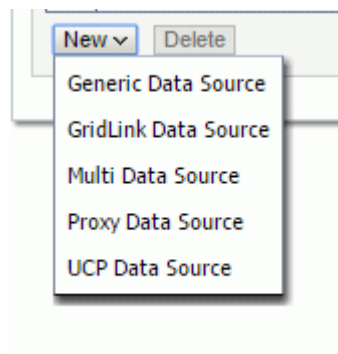
3.4.5 Creating a data source

The HSL application needs a data source to connect with the OHI Back Office database.

To create a data source, navigate in the Domain Structure panel on the left to the data sources option. Choose 'Lock & Edit' so you are able to create a new data source.



Create a new 'Generic data source':



Choose a name for the data source to reflect its purpose. For example, you may want to reference the database name: DS_OHI_prd.

Next specify a JNDI name. The JNDI name will be used in the properties file for starting the HSL application.

Specify 'Oracle' as the database type.

An example:

Home Log Out Preferences Record Help

Home > Summary of JDBC Data Sources

Create a New JDBC Data Source

Back Next Finish Cancel

JDBC Data Source Properties

The following properties will be used to identify your new JDBC data source.
* Indicates required fields

What would you like to name your new JDBC data source?

Name: DS_vohi_dev

What scope do you want to create your data source in ?

Scope: Global ▼

What JNDI name would you like to assign to your new JDBC Data Source?

JNDI Name:
jdbc/DSvohi

What database type would you like to select?

Database Type: Oracle ▼

Back Next Finish Cancel

Next you need to specify a database driver. Use "Oracle's Driver (Thin) for Service connections; Versions: Any". If you are using RAC (or considering to use RAC) choose the thin RAC driver. Do not use the XA driver.

Home Log Out Preferences Record Help

Home > Summary of Machines > ol6ohi.ohi.oracle.com > Summary of JDBC Data Sources

Create a New JDBC Data Source

Back Next Finish Cancel

JDBC Data Source Properties

The following properties will be used to identify your new JDBC data source.

Database Type: Oracle

What database driver would you like to use to create database connections? Note: * indicates that the driver is explicitly supported by Oracle WebLogic Server.

Database Driver:

Back Next Finish Cancel

- *Oracle's Driver (Thin) for Service connections; Versions:Any
- *Oracle's Driver (Thin XA) for Application Continuity; Versions:Any
- *Oracle's Driver (Thin XA) for Instance connections; Versions:Any
- *Oracle's Driver (Thin XA) for RAC Service-Instance connections; Versions:Any
- *Oracle's Driver (Thin XA) for Service connections; Versions:Any
- *Oracle's Driver (Thin) for Application Continuity; Versions:Any
- *Oracle's Driver (Thin) for Instance connections; Versions:Any
- *Oracle's Driver (Thin) for RAC Service-Instance connections; Versions:Any
- *Oracle's Driver (Thin) for Service connections; Versions:Any
- *Oracle's Driver (Thin) for pooled instance connections; Versions:Any
- DataDirect's Oracle Driver (Type 4 XA) Versions:Any
- DataDirect's Oracle Driver (Type 4) Versions:Any
- Other

Choose the following Transaction Options:

- 'Supports Global Transactions';
- 'One-Phase Commit' (this is why you don't need the XA driver)

Example:

Home Log Out Preferences Record Help Welcome, weblogic Connected to: ohi_svl

Home > Summary of Machines > o6ohi.ohi.oracle.com > Summary of JDBC Data Sources

Create a New JDBC Data Source

Back Next Finish Cancel

Transaction Options

You have selected non-XA JDBC driver to create database connection in your new data source.

Does this data source support global transactions? If yes, please choose the transaction protocol for this data source.

Supports Global Transactions

Select this option if you want to enable non-XA JDBC connections from the data source to participate in global transactions using the *Logging Last Resource* (LLR) transaction optimization. Recommended in place of Emulate Two-Phase Commit.

Logging Last Resource

Select this option if you want to enable non-XA JDBC connections from the data source to emulate participation in global transactions using JTA. Select this option only if your application can tolerate heuristic conditions.

Emulate Two-Phase Commit

Select this option if you want to enable non-XA JDBC connections from the data source to participate in global transactions using the one-phase commit transaction processing. With this option, no other resources can participate in the global transaction.

One-Phase Commit

Back Next Finish Cancel

Next specify the connection details like the example on the page below. Be sure to use values which are valid for your environment.

Create a New JDBC Data Source

Back Next Finish Cancel

Connection Properties
Define Connection Properties.

What is the name of the database you would like to connect to?

Database Name:

What is the name or IP address of the database server?

Host Name:

What is the port on the database server used to connect to the database?

Port:

What database account user name do you want to use to create database connections?

Database User Name:

What is the database account password to use to create database connections?

Password:

Confirm Password:

Additional Connection Properties:

oracle.jdbc.DRCPConnectionClass:

Back Next Finish Cancel

On the next page the result of your answers will be shown. You can test the connection with the data shown (the table name is not relevant).

When you navigate to the next page you can select the targets where the data source should be deployed to. In the example below only the Managed Server shown will be used for deploying the data source to.

Servers

AdminServer

ms_ohi_hsl

Back Next Finish Cancel

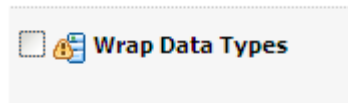
Press Activate Changes to conclude your configuration.

At this point, go back to your data source and re-open the connection pool tab.

Navigate to the 'Advanced' part.

Ensure that the option 'Wrap Data Types' is unchecked. This setting is needed for passing CLOB objects to and from the database and when activated slows down execution. Press Lock & Edit and uncheck this option and Save and Activate the change.

Example:



3.5 Security Configuration

All HSL applications are preconfigured to use basic authentication and SSL encryption.

The following steps are needed to set up minimal security for the HSL application:

- Set up security realm
- Setup Weblogic user for accessing HSL application
- Enable SSL
- Configure JSSE
- Configure key store
- Configure logging level
- Configure user lockout

Before you can install HSL applications, you need to decide on the security model you want to use. The options are:

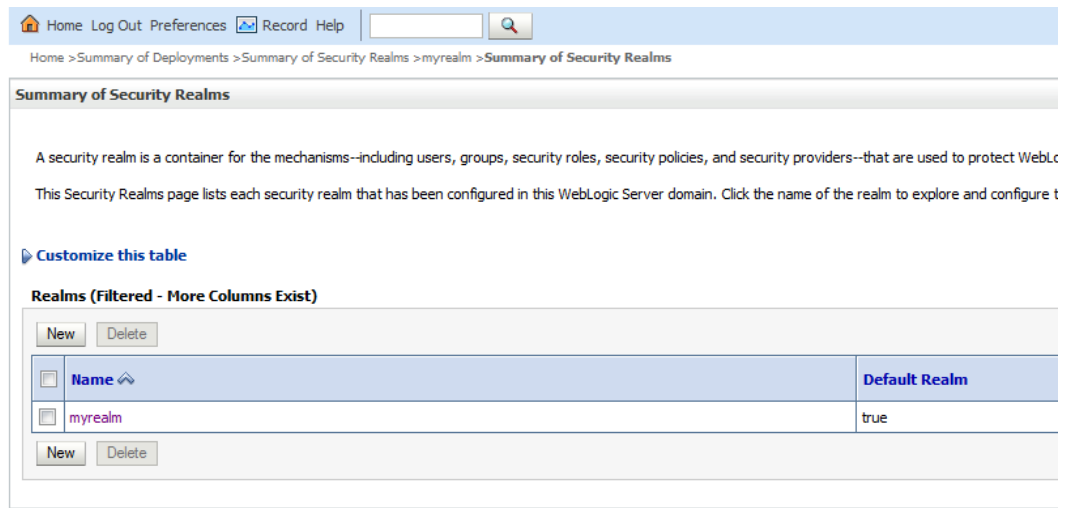
1. Use the predefined default user `restuser` and the predefined policy. During deployment, choose the security model indicated with 'DD Only'.
2. Use the default policy and create your own roles to restrict access to the web services. During deployment, choose the security model indicated with 'Custom Roles'
3. Use a custom security model to overrule the default of each web service. See paragraph 3.7 "Additional Security Aspects". During deployment, choose the security model indicated with 'Custom Roles and Policies'.

If you want to use OAUTH 2.0 token authentication and validation (as an alternative to Basic Authentication) you need to choose Custom Roles and Policies'. See *Appendix E - Authentication and Authorization* for details.

3.5.1 Set up security realm

Create a security realm if this has not already been done (normally realm 'myrealm' will already be present).

The security realm 'myrealm' as shown below will be used to configure the security at application level.



If there are no other security realms, this will be the default security realm.

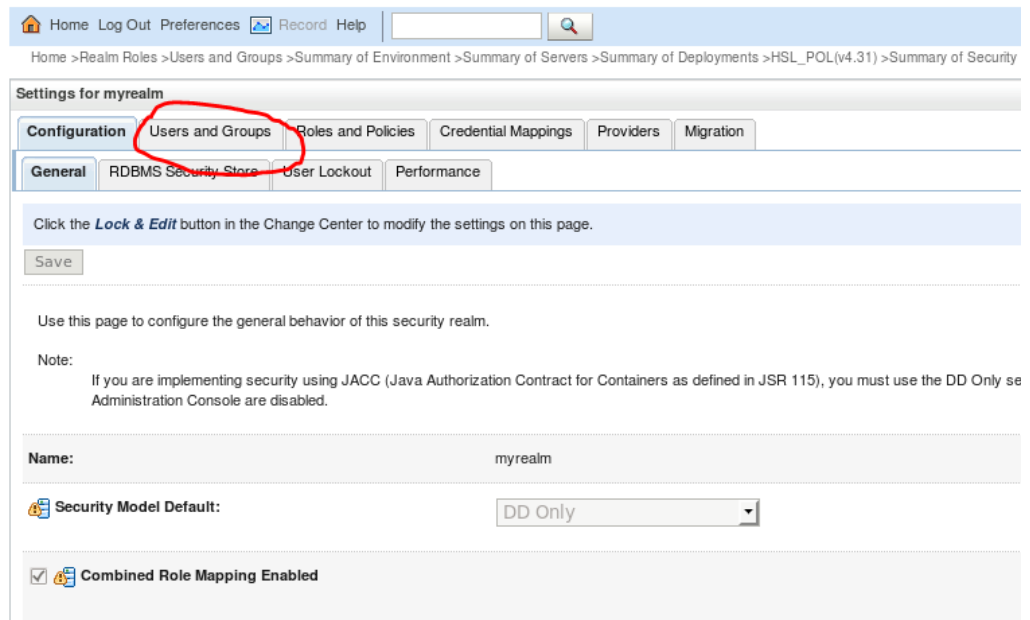
3.5.2 Setup Weblogic user for accessing HSL application

Each operation of an HSL application requires basic authentication. This means that each call must be made as an authenticated Weblogic user.

The name of the default Weblogic user to access the HSL application is defined in a preconfigured deployment descriptor in the WAR file. This default name is 'restuser'.

So if you deploy the HSL application with the default security model (DD Only) you will need to create the Weblogic user 'restuser' and authenticate as 'restuser' when invoking the HSL application.

To set up the Weblogic user 'restuser', select 'Users and Groups' for your security realm:



Add 'restuser' to the pool of Weblogic users:

Home >Users and Groups >Summary of Environment >Summary of Servers >Summary of Deployments >HSL_POL(v4.31) >Summary of Security Realms >my

Messages
 ✓ User created successfully

Settings for myrealm

Configuration **Users and Groups** Roles and Policies Credential Mappings Providers Migration

Users Groups

This page displays information about each user that has been configured in this security realm.

Customize this table

Users (Filtered - More Columns Exist)

New Delete

<input type="checkbox"/>	Name ↕	Description
<input type="checkbox"/>	LCMUser	This is the default service account for WebLogic Server Lifecycle Manager configuration updates.
<input type="checkbox"/>	OracleSystemUser	Oracle application software system user.
<input type="checkbox"/>	restuser	weblogic user for accessing HSL applications
<input type="checkbox"/>	weblogic	This user is the default administrator.

New Delete

Note that if you do not want to authenticate an HSL application using the predefined weblogic user 'restuser', you can choose to deploy using 'Custom Roles and Policies'. See 'Additional Security Aspects' below.

3.5.3 Enable SSL

The HSL services are preconfigured to use a default policy which uses SSL. Therefore you need to enable SSL for every Managed Server to which you deploy the HSL application.

Go to the Managed Server configuration and enable SSL in the 'Configuration > General' tab:

Settings for ms_ohi_hsl

Configuration Protocols Logging Debug Monitoring Control Deployments Services Security Notes

General Cluster Services Keystores SSL Federation Services Deployment Migration Tuning Overload Concurrency Health Monitoring

Save

Use this page to configure general features of this server such as default network communications.

Name: ms_ohi_hsl

Template: (No value specified) Change

Machine: Machine1

Cluster: (Stand-Alone)

Listen Address: localhost

Listen Port Enabled

Listen Port: 7063

SSL Listen Port Enabled

SSL Listen Port: 7064

Client Cert Proxy Enabled

Java Compiler: javac

Diagnostic Volume: Low

Default Datasource:

Advanced

Save

3.5.4 Configure JSSE

To use SSL with WebLogic you need to configure the use of Java Secure Socket Extension (JSSE) as this is the only supported SSL implementation. The RSA JSSE provider is not installed as part of Weblogic Server since WLS 12.1.1 but needs to be provided by the JVM.

It depends on the JDK version whether additional configuration action is required.

For more generic information about Oracle's JDK and JRE cryptographic algorithms please visit: https://www.java.com/en/configure_crypto.html

For more information regarding the changes in the specific JDK 8 releases as mentioned below:

<http://www.oracle.com/technetwork/java/javase/8all-relnotes-2226344.html>

..3.5.4.1 JDK 1.8.0_162 and above

No action is needed.

..3.5.4.2 JDK 1.8.0_151 .. 1.8.0_161

Only a small configuration change in your JDK is required.

Uncomment the following line in <JDK_HOME>/jre/lib/security/java.security:

```
#crypto.policy=unlimited
```

Remove the hash (#) from this line to enable the RSA JSSE provider.

..3.5.4.3 Below JDK 1.8.0_151

To configure the use of RSA JSSE, follow the instruction at [Using the RSA JSSE Provider in WebLogic Server](#) in paragraph “Using the RSA JSSE Provider in WebLogic Server”.

The installation means that you have to replace two jar files within the JDK installation that is used by WebLogic. These files are JDK version specific and contain the stronger encryption methods that are needed.

As summarized during an OHI presentation:

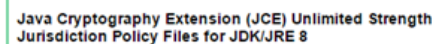
SVL domain creation – JSSE continued

- Lot of text: what is meant?

- Download zip with JSSE implementation

- <http://www.oracle.com/technetwork/java/javase/downloads/index.html>

- Unzip download (/tmp ?)



Java Cryptography Extension (JCE) Unlimited Strength
Jurisdiction Policy Files for JDK/JRE 8

DOWNLOAD

- replace 2 files in the JDK 8 install (as root!), i.e.:

- ```
cd /usr/java/jdk1.8.0_102/jre/lib/security
cp /tmp/UnlimitedJCEPolicyJDK8/local_policy.jar .
cp /tmp/UnlimitedJCEPolicyJDK8/US_export_policy.jar .
```
- Adapt file in same folder:  

```
vi java.security
```
- Add new first line: 

```
security.provider.1=com.rsa.jsse.JsseProvider
```
- Resequence existing lines from 2..10

Typically the name of the downloaded file will be jce\_policy-8.zip.

### 3.5.5 Setting up a key store

---

For testing purposes you may want to use the built-in keystore as shown below in the ‘Configuration > Keystores’ tab for the Managed Server:

Settings for ms\_ohi\_hsl

Configuration Protocols Logging Debug Monitoring Control Deployments Services Security Notes

General Cluster Services **Keystores** SSL Federation Services Deployment Migration Tuning Overload Concurrency Health Monitoring

Click the **Lock & Edit** button in the Change Center to modify the settings on this page.

Save

Keystores ensure the secure storage and management of private keys and trusted certificate authorities (CAs). This page lets you view and define various keystore configurations.

**Keystores:** Demo Identity and Demo Trust

— Identity —

**Demo Identity Keystore:** C:\Oracle\weblogic\_12c\user\_projects\domains\ohi\_hsl\security\DemoIdentity.jks

**Demo Identity Keystore Type:** jks

**Demo Identity Keystore Passphrase:**

— Trust —

**Demo Trust Keystore:** C:\Oracle\WEBLOG~1\server\server\lib\DemoTrust.jks

**Demo Trust Keystore Type:** jks

**Demo Trust Keystore Passphrase:**

**Java Standard Trust Keystore:** C:\PROGRA~1\Java\JDK18~1.0\_7\jre\lib\security\cacerts

**Java Standard Trust Keystore Type:** jks

**Java Standard Trust Keystore Passphrase:**

**Confirm Java Standard Trust Keystore Passphrase:**

Save

Click the **Lock & Edit** button in the Change Center to modify the settings on this page.

**Note that in a production environment it is not safe to use the demo keystore.**

For more information about configuring keystores please read the WebLogic documentation. As a starter you can use this address: [Oracle® Fusion Middleware Administering Security for Oracle WebLogic Server 12.2.1 - 29 Configuring Keystores](#)

It contains references to pages which describe in more detail how to obtain private keys, digital certificates, etc.

You should take action and not rely on the demo keystore!

### 3.5.6 Configure Managed Server logging level

The standard logging level for a Managed Server regarding security issues is intentionally non-informative to discourage fraudulent users.

A typical security-related error message looks like:

*Got 'Unknown exception, internal system processing error.'*

If you are trying to setup the HSL application to work with SSL and basic authentication in a non-production environment you can configure verbose logging with the following Server Start argument for the Managed Server:

```
-Dweblogic.wsee.security.debug=true
```

Create a new file `$DOMAIN_HOME/bin/setUserOverrides.sh` and add the following text:

```
#!/bin/bash
echo Adding Settings from UserOverrides.sh

global settings (for all servers)
this will decrease start up times
JAVA_OPTIONS="-Djava.security.egd=file:/dev/./urandom" "${JAVA_OPTIONS}"
export JAVA_OPTIONS
CONFIG_JVM_ARGS="-Djava.security.egd=file:/dev/./urandom ${CONFIG_JVM_ARGS}"
export CONFIG_JVM_ARGS

specify additional java command line options for the Admin Server
#if ["${SERVER_NAME}" = "${AS_NAME}"]
#then
#
#fi
#export JAVA_OPTIONS

specify additional java command line options for specific servers
if ["${SERVER_NAME}" = "ms_ohi_hsl"]
then
 # add settings for HSL
 # Custom Setting for SSL
 JAVA_OPTIONS="-Dweblogic.wsee.security.debug=true" "${JAVA_OPTIONS}"
fi
export JAVA_OPTIONS
```

Replace the server name `ms_ohi_hsl` with your server name.

When startup times of your service calls are important and the security of the connection is less important you may consider to specify an alternative for retrieving cryptographically strong random numbers (included above):

```
JAVA_OPTIONS="-Djava.security.egd="file:/dev/./urandom" "${JAVA_OPTIONS}"
```

Restart the Managed Server to get the new verbose messages later on.

### 3.5.7 Set user lockout

While setting up HSL services for testing you may want to disable user lockout. In a production environment you should enable user lockout to discourage fraudulent use. Navigate to the Security Realm and use the 'Configuration > User Lockout' tab.

Home > Summary of Deployments > Summary of Security Realms > myrealm > Summary of Security Realms > myrealm

Settings for myrealm

Configuration Users and Groups Roles and Policies Credential Mappings Providers Migration

General RDBMS Security Store **User Lockout** Performance

Save

Password guessing is a common type of security attack. In this type of attack, a hacker attempts to log in to a computer using various security realm.

Lockout Enabled

Lockout Threshold: 5

Lockout Duration: 30

Lockout Reset Duration: 5

Lockout Cache Size: 5

Lockout GC Threshold: 400

Save



## 3.6 (Re)deployment of the HSL Application

HSL applications are deployed through WAR (Web Application Archive) files. Each HSL service application has its own WAR file, for example HSL\_POL.war or HSL\_REL.war.

The WAR file of a HSL application resides in the \$OZG\_BASE/java directory on the application server containing the OHI Back Office software release. You can copy this to another location if required.

Ensure that the .war file is located on the WLS Admin Server host (this is the server running the WLS Administration Console).

Note that you cannot use an older WAR file with a newer OHI Back Office release and vice versa.

The following scenarios are discussed:

- Deploy to a single Managed Server
- Deploy to multiple Managed Servers
- Deploy to a cluster
- Deploy for DTAP (development, test, acceptance, production)

### 3.6.1 Deploy to a single Managed Server

---

#### ..3.6.1.1 Deploy WAR files

Repeat the following tasks for each HSL WAR file.

In the Domain Structure pane, select the Deployments branch. This will show the applications that have already been deployed

If you want to shorten this list, use 'Customize this table' to exclude the libraries.

Select 'Lock & Edit' to enter editing mode, this will enable the 'Install' button which you need to use next.

In the new window, locate the .war files on the WLS server, select one and press 'Next':



Note: See Appendix E to decide if you need to install HSL\_AUN and HSL\_AUZ too.

Select 'Install this deployment as an application', press 'Next' and select the target(s) for deployment. In the example below only Managed Server ms\_ohi\_hsl is chosen.

| Servers                             |             |
|-------------------------------------|-------------|
| <input type="checkbox"/>            | AdminServer |
| <input checked="" type="checkbox"/> | ms_ohi_hsl  |

Press 'Next' and decide about a deployment name and security model. At this moment the version of the .war file is also shown (can contain up to 4 digits like any application source).

**Install Application Assistant**

Back Next Finish Cancel

---

**Optional Settings**

You can modify these settings or accept the defaults.

\* Indicates required fields

---

**General**

What do you want to name this deployment?

\* Name:

Archive Version: v4.5

Deployment Plan Version:

---

**Security**

What security model do you want to use with this application?

DD Only: Use only roles and policies that are defined in the deployment descriptors.

Custom Roles: Use roles that are defined in the Administration Console; use policies that are defined in the deployment descriptor.

Custom Roles and Policies: Use only roles and policies that are defined in the Administration Console.

Advanced: Use a custom model that you have configured on the realm's configuration page.

---

**Source Accessibility**

How should the source files be made accessible?

Use the defaults defined by the deployment's targets

Recommended selection.

Copy this application onto every target for me

During deployment, the files will be copied automatically to the Managed Servers to which the application is targeted.

I will make the deployment accessible from the following location

Select 'Custom Roles' if you want to use the default policy and create your own roles to restrict access to the web services. Select 'Custom Roles and Policies' if you want to overrule the default of each web service.

Regarding source accessibility, select 'Copy this application....' if you want to remove the WAR file from its current location.

Finish the configuration.

Beware that – in Production mode - you need to Activate your changes in order to enable the web services. At that moment the deployment will show status 'Prepared'.

By selecting the deployment in the Control tab and pressing Start → Servicing all requests the State will change to 'Active' (assuming your Managed Server is in 'Running' state, the hsl.properties file has been specified and can be found).

### ..3.6.1.2 Specify configuration file

Before using the web services, implement the following actions as described below. These actions have to be executed only once. There is no need to repeat them when you update a deployment or delete and install it again.

Add a Server Start argument by adding a line to the file `$DOMAIN_HOME/bin/setUserOverrides.sh` you created earlier. Add the line to the part for the HSL server, as indicated below:

```
specify additional java command line options for specific servers
if ["${SERVER_NAME}" = "ms_ohi_hsl"]
then
 # add settings for HSL
 # Custom Setting for ms_ohi_hsl to set debug level for SSL
 JAVA_OPTIONS="-Dweblogic.wsee.security.debug=true" ${JAVA_OPTIONS}"
 # Set location for HSL properties file
 JAVA_OPTIONS="-Dhsl.properties="/u01/app/oracle/product/OHI/vohi/hsl.properties" ${JAVA_OPTIONS}"
fi
export JAVA_OPTIONS
```

- Make sure to keep the parts with `{JAVA_OPTIONS}` on the same line

This example uses a properties file with the name `hsl.properties` which is located in the `$OZG_BASE` folder of your OHI Back Office application server environment, but you can specify any name and location.

The contents of this file are discussed in a Chapter 5 "Configuration files for HSL services").

When completed, (re)start the Managed Server. This can be done from the WebLogic Admin console, or from the command line with the following commands;

```
cd $DOMAIN_HOME/bin
./startManagedWebLogic.sh ms_ohi_hsl http://localhost:7061
```

The example above contains the Managed Server's name as first parameter and the listen address of the Admin Server of the domain as second parameter

Check in the `<ManagedServer>.out` file in the logs folder of your Managed Server whether the command line contains the arguments as specified above.

If the file specified by `hsl.properties` cannot be read , messages as below will show up:

```
ERROR: logfile could not be set because of: null
```

## 3.6.2 Deploy to multiple Managed Servers

---

You may deploy the application to more than one target.

Example: if you choose to target the application to Managed Servers MS1 and MS2, the application will be available on separate end points. The URLs of these end points will only differ in port number.

If you choose this scenario, be aware that each Managed Server should have different Server Start argument values (`hsl.properties`).

### 3.6.3 Deploy to cluster

You may deploy the application on all the Managed Servers of a cluster. This may be needed for better scalability. Be aware to use some form of load balancing to allow the use of a single end point.

The best way to implement this type of deployment depends on your specific situation.

If you are planning a load balanced environment with multiple Managed Servers in a cluster it is vital that the configuration of every Managed Server is aligned with the others.

If you deploy to the cluster, it is recommended to redirect the logging of all Managed Servers to a single file.

### 3.6.4 Deploy for multiple environments (DTAP)

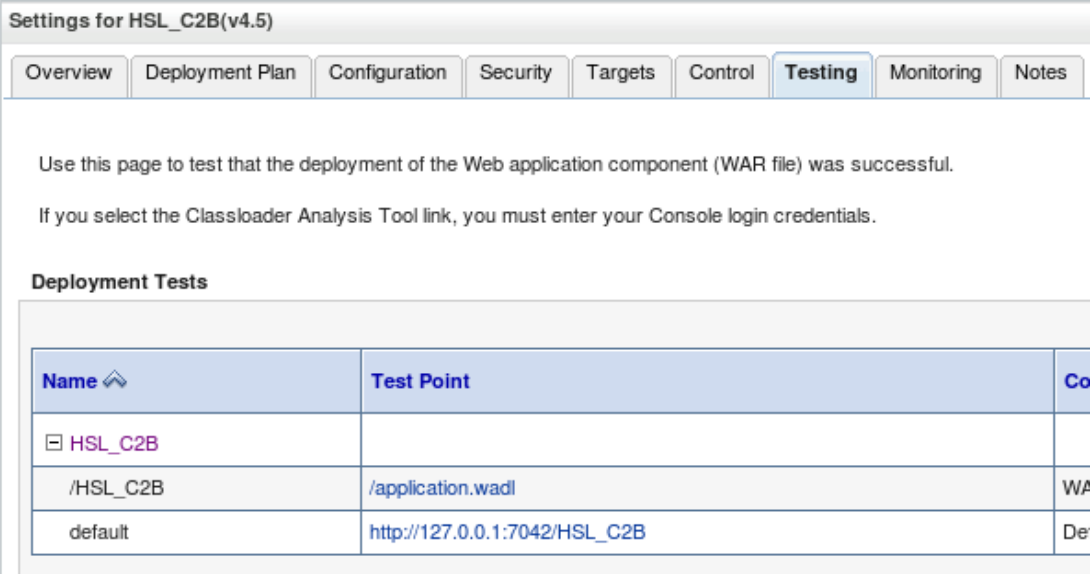
If you use several OHI-related environments to support the various DTAP (Develop-Test-Accept-Production) stages you may want to have different versions of the HSL application running at the same time.

To implement this you need to:

- Create a Managed Server for each of the DTAP stages.
- Create a data source for each OHI Back Office database and deploy that data source only to the corresponding Managed Server.
- Create an hsl.properties file for each Managed Server.
- Configure each Managed Server to start up with the appropriate hsl.properties.
- Deploy the appropriate version of the HSL application to its corresponding Managed Server and give it a unique deployment name to identify its deployment.

### 3.6.5 Validate deployment

Be aware that the URLs displayed in the Admin Console cannot be used to test or validate the deployment.



Settings for HSL\_C2B(v4.5)

Overview Deployment Plan Configuration Security Targets Control **Testing** Monitoring Notes

Use this page to test that the deployment of the Web application component (WAR file) was successful.

If you select the Classloader Analysis Tool link, you must enter your Console login credentials.

**Deployment Tests**

| Name     | Test Point                    | Co |
|----------|-------------------------------|----|
| HSL_C2B  |                               |    |
| /HSL_C2B | /application.wadl             | WA |
| default  | http://127.0.0.1:7042/HSL_C2B | De |

Also note that, even with the correct URLs, you cannot use a browser to test, because the request needs to send a Request Header "Accept:application/json".

You may get no response, or a reply like this:

```
<exceptionResponse>
<internalStatus>Not Acceptable</internalStatus>
<message>Wrong value for Accept</message>
</exceptionResponse>
```

Instead, use curl, as described in chapter 4 "Deployment validation" or SoapUI, as described in Appendix C "Testing with SoapUI".

## 3.7 Additional Security Aspects

Since HSL services provide an additional channel to access OHI Back Office data, you must prevent unauthorized use of the HSL applications.

Please consult the 'Oracle Health Insurance Security Aspects' guide for more information about OHI Back Office security aspects.

In order to prevent the exposure of sensitive data or unauthorized changes to the OHI Back Office data, access of the HSL applications should be limited to trusted systems and interfaces. Otherwise people in your organization might be tempted to try to misuse the functionality provided by the HSL services.

By default, HSL service operations require basic authentication as a minimal policy to reduce the risk of unauthorized access and network sniffing. Basic authentication requires HTTPS communication and providing a Weblogic username/password with each call.

The preconfigured deployment descriptor in the HSL applications requires authentication by a Weblogic user 'restuser'. See the instructions for creating this user in 'Security Configuration' above.

If you deploy using 'Custom Roles and Policies' you may:

- Deploying HSL Application for use with any weblogic user
- Using a custom security policy for a deployed application

Note that all 'admin' requests require Basic Authentication and SSL (/swagger, /swagger.json, /swagger.yaml, /dbinfo and /templates)

Finally, note that it is your responsibility as an administrator to secure the HSL services within your organization.

### 3.7.1 Deploying HSL Application for use with any weblogic user

---

If you want to deploy a HSL application for use with another weblogic user than the default 'restuser', you should deploy with the security model 'Custom Roles and Policies':

**Install Application Assistant**

---

**Optional Settings**

You can modify these settings or accept the defaults.

\* Indicates required fields

**— General —**

What do you want to name this deployment?

\* Name:

Archive Version: v4.5

Deployment Plan Version:

**— Security —**

What security model do you want to use with this application?

**DD Only:** Use only roles and policies that are defined in the deployment descriptors.

**Custom Roles:** Use roles that are defined in the Administration Console; use policies that are defined in the deployment descriptor.

**Custom Roles and Policies:** Use only roles and policies that are defined in the Administration Console.

**Advanced:** Use a custom model that you have configured on the realm's configuration page.

**— Source Accessibility —**

How should the source files be made accessible?

**Use the defaults defined by the deployment's targets**

Recommended selection.

**Copy this application onto every target for me**

You can now use any weblogic user to access the HSL application.

### 3.7.2 Using a custom security policy for a deployed application

The Weblogic console allows the administrator to specify a custom security policy for HSL applications deployed using 'Custom Roles and Policies'. For example, a custom security policy can be used:

- to limit access to a specified list of named Weblogic users; or
- to limit access to a group of Weblogic users; or
- to limit access to Weblogic users with a specific role;
- or a combination of the above.

### 3.7.3 Cross Site scripting protection

As a measure to prevent against potential cross site scripting attacks you can limit the callers of the REST services by specifying a set of trusted origins. For this the property 'allowedorigins' can specify one or more originating (trusted) server URL's (property files are discussed later on in this document).

An example:

```
hsl.rel.allowedorigins=https://server.domain.com:7430, https://localhost:8000
```

## 4 Deployment validation

Especially when deploying a HSL application for the first time, it makes sense to validate that the HSL application is in working order.

Before you begin, check in the WLS Admin Console that the deployment status of the HSL application is active.

The following validation tests can be performed by the administrator:

- Template Listing
- getDatabaseInfo operation
- Get Online Swagger definition

Apart from the template listing, each of the validations requires a JDBC connection between the HSL application and the OHI BO database, so you are not only testing the deployment itself but also the integration between the HSL application and the OHI database.

If a validation test fails see the paragraph '*Troubleshooting*' below to find and resolve the problem.

The validation tests described below assume that you test with 'curl'.

### 4.1 Testing with Curl

An operation of the HSL application can be invoked with many HTTP client tools. One of these tools is curl, which is present on any Linux/Unix server. Assuming that you have terminal access to the Linux server running WLS, curl is a good tool to run the deployment validation tests.

Use 'curl --version' to check the curl version. Ensure that you are running curl 7.35.0 or higher as that supports the required SSL implementation.

A typical invocation of a HSL operation using curl would look like this

```
curl -D - -X <verb> -k -H Accept:application/json --user <user> <url>
```

Explanation of the used options and placeholders:

- -D -  
Dump response headers to stdout
- -X <verb>  
add HTTP verb (GET/PUT/POST/PATCH/DELETE)
- -k  
Allow curl to run HTTP requests without checking SSL certificates.
- -H Accept:application/json  
Add request header to require a response in application/json format. This is required for every HSL operation.
- --user <user>  
The username of the WLS user used for Basic Authentication.

The <user> must refer to an existing WLS user.  
Note that curl will prompt for a password if it is not given at the command line.

- <url>  
The path to the HSL operation.

The url format is <https://server:port/application/path> where

<b>server</b>	<b>This must be one of the managed servers listed in WLS console as an active target for the HSL application.</b>
<b>port</b>	The SSL port of the managed server running the HSL application.  Every HSL operation requires SSL and Basic Authentication.
<b>application</b>	This is the name of the HSL application as listed on the WLS deployment page.  For example, HSL_POL, HSL_REL, HSL_CLA or HSL_C2B.
<b>path</b>	The path to this operation. Each operation is uniquely identified by a <path> + <verb> combination.  Path examples: '/dbinfo' or 'templates' or 'api/swagger.json'

In the following example, a template listing is requested from the HSL\_POL application on the local WLS host running a managed server at SSL port 7094:

```
curl -D - -k --user restuser -H Accept:application/json
https://localhost:7094/HSL_POL/templates
```

## 4.2 Template Listing

This operation lists the templates which were used to generate the Java code for the HSL application. The listing itself is irrelevant, but since the operation does not require a JDBC connection with the OHI BO database, it is the simplest of all deployment tests. If it fails the remaining deployment tests will also fail.

The template listing is invoked through

```
https://server:port/application/templates
```

In the following example we retrieve the template listing through curl for the HSL\_POL application. It is assumed that we run curl locally on the WLS server and that the managed server running the HSL application can be accessed through SSL port 7094.

```
curl -k -H Accept:application/json --user restuser
https://localhost:7094/HSL_POL/templates
```

The output is a JSON object listing template files and template versions used to generate the Java code for the HSL application.

## 4.3 getDatabaseInfo

This operation provides information about the database connection between the HSL application and the OHI BO database.

If you are familiar with OHI BO's SVL services, note that the getDatabaseInfo



operation is comparable with the 'isAlive' operation implemented in every SVL service.

This operation requires a working database connection and invokes the PL/SQL implementation package specific to the HSL application.

The `getDatabaseInfo` operation is invoked through

```
https://server:port/application/dbinfo
```

In the following example the `getDatabaseInfo` of the HSL\_POL application running on SSL port 7094 on our local WLS host is invoked through curl.

```
curl -k -H Accept:application/json --user restuser
https://localhost:7094/HSL_POL/dbinfo
```

The output is a JSON object with information about the database connection and the PL/SQL package implementing the HSL application in the OHI BO database:

```
{
 "basePath": "https://localhost:7094/HSL_POL/pol",
 "database": "BDDEV1722",
 "instance": "CDB02",
 "jndiName": "HSL_BDDEV1722",
 "plsqlPackage": "hsl_pol_sp_pck $Revision: 4.21 $",
 "user": "HSL_USER",
 "userContext": "MANAGER"
}
```

## 4.4 Get Online Swagger definition

Each HSL application has an operation to generate a Swagger definition which documents the operations and the objects of the HSL service.

This documentation is not only useful to client application developers, but can also be used as the basis for code generation.

The Swagger 2.0 standard is supported by many leading software vendors including Oracle. It is documented on <http://swagger.io>.

The Swagger definition can be retrieved as follows:

- `https://server:port/application/api/swagger.json`  
Returns the Swagger definition in JSON format
- `https://server:port/application/api/swagger`  
Returns the Swagger definition in JSON format
- `https://server:port/application/api/swagger.yaml`  
Returns the Swagger definition in YAML format

In the following example we retrieve the online Swagger definition of the POL service running on localhost at SSL port 7094:

```
curl -k -H Accept:application/json --user restuser
https://localhost:7094/HSL_POL/api/swagger.json
```

The output is a JSON object containing the Swagger definition of the deployed HSL application.

For retrieving the YAML format beware that you specify `x-yaml` in the `-H` argument:

```
curl -k -H Accept:application/x-yaml --user restuser
https://localhost:7094/HSL_POL/api/swagger.yaml
```

#### 4.4.1 Saving the Swagger definition to a file

---

By redirecting the output of the curl command to a file you can use the contents for other purposes like viewing the Swagger definition in an editor.

In the example below we save the online Swagger definition of the POL service running on localhost at SSL port 7094 to a file called 'saved\_swagger.json':

```
curl -k --user restuser -H Accept:application/json
https://localhost:7094/HSL_POL/api/swagger.json >
saved_swagger.json
```

#### 4.4.2 Viewing the Swagger definition

---

The online Swagger editor (<http://editor.swagger.io>) provides a user friendly overview of the Swagger definition.

In the following example we use curl to retrieve the Swagger definition of the HSL\_POL service and save it to a file. Having opened the saved Swagger definition we then copy its contents into the online Swagger Editor.

Assuming that the HSL\_POL application is running on localhost at port 7094 the following command can be used to save the Swagger definition to 'saved\_swagger.json':

```
curl -k --user restuser -H Accept:application/json
https://localhost:7094/HSL_POL/api/swagger.json >
saved_swagger.json
```

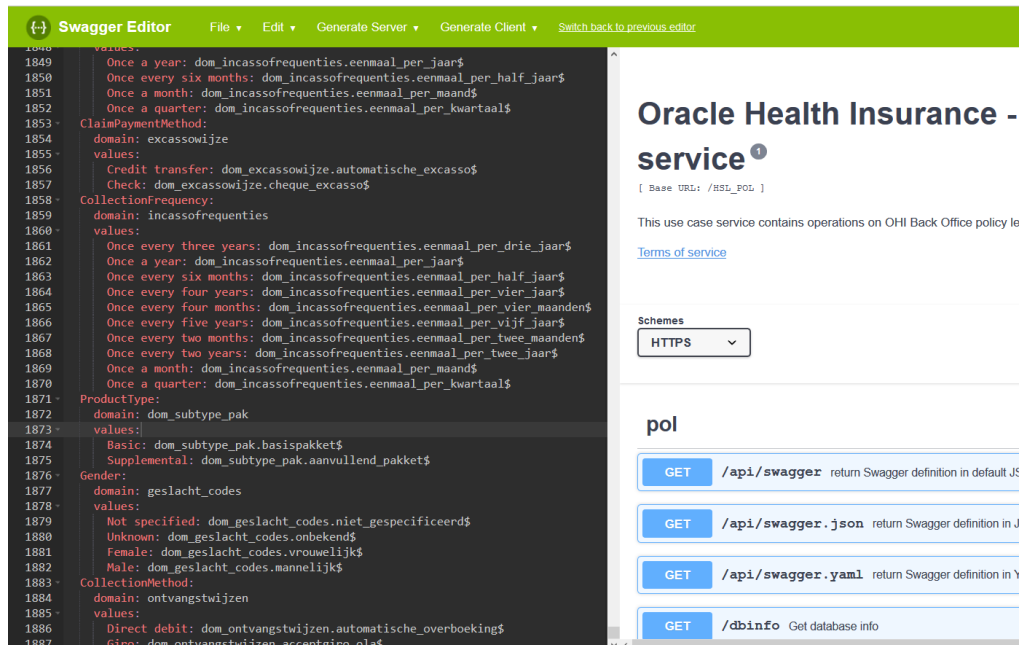
Open 'saved\_swagger.json' in a text editor (or browser) and copy the contents of the entire file.

Open the online Swagger editor by browsing <http://editor.swagger.io>

Use 'File > Clear Editor' to clear the contents of the online editor (if any).

Right-click and paste the contents of the saved Swagger definition into the editor.

The screen will look like this:



You can now navigate through the paths, operations and type definitions of the HSL service. More information about Swagger can be found on <http://swagger.io>

## 4.5 Troubleshooting

If the deployment validation fails, first check that the following items have been configured correctly:

- **hsl.properties configuration file**  
This sets the data source for your HSL application.
- **hsl.properties Server Start argument**  
This parameter tells the HSL application where to find the hsl.properties file. If the hsl.properties parameter refers to a non-existing file, the HSL application cannot be started by WLS and its state will be 'Failed'.
- **Data source configured in hsl.properties configuration file**  
This data source is used to create the JDBC connection between the HSL application and the OHI BO database
- **HSL database account**  
This account in the OHI BO database has access to the PL/SQL components used by the HSL application.

Troubleshooting tips:

- Edit the hsl.properties file and set 'developermode=true' for your HSL application.
- Restart the managed server for your application.  
Error messages will now be included in the output (normally they are suppressed from the output).
- Reproduce the request with curl. Be sure to use the 'dumpheader' option (-D) to dump the response headers.

The table below may help you to pinpoint the problem:

HTTP	Message	Problem	Action
	WLS Console: java.lang.RuntimeException: Property file could not be loaded.	The configuration file could not be loaded when starting the HSL application.	Restart application after ensuring that the file referred to by the 'hsl.properties' Server Start argument exists and is readable.
500	Unable to resolve 'xyz'	The jndiname property for this application does not refer to a valid data source.	Examine hsl.properties and ensure that the application's jndiname points to a valid datasource.
500	ORA-06550:line 1..	The required HSL objects cannot be accessed by the database user related to the data source.	Verify that the data source points to a HSL database account.  Verify that the HSL database account has access to the HSL objects (see 'Creating a HSL database account' above).
401	Missing Authentication Scheme	No WLS user credentials supplied for this request	Add WLS user credentials to the request. The preconfigured WLS user is 'restuser'.
401	Unauthorized	Wrong WLS user credentials. Wrong username and/or password.	Add correct WLS user credentials to the request.
422	Functional Error	A syntactically correct request could not be completed due to a functional error or business rule violation.	Adapt the request or the database situation so the cause of the failure is prevented.
412	Precondition Failed	Another user already updated these data.	Refresh data and try again.

## 5 Configuration Files for HSL services

In the previous chapter a properties file was referenced in the web service application server deployment description. This chapter provides more information about that file.

### 5.1 Back Office HSL properties file

The location of the Back Office properties file for the HSL services is specified as a Server Start argument for a Managed Server with:

```
-Dhsl.properties=<filename>
```

This file contains properties to configure the various deployed HSL applications:

- Datasource to connect the HSL application to the OHI database
- Default OHI officer on whose behalf a request is executed
- Logging configuration.  
Note that HSL services use Java Util Logging (JUL). You may find more information about the configuration options of JUL on the internet.

#### 5.1.1 hsl.jndiname

---

The JNDI name of the default data source to connect the HSL application to the OHI database.

Default value: none

Example:

```
hsl.jndiname=HSL_BDDEV1622
```

Note that you must use // for each forward slash in the JNDI name.

Example:

```
hsl.jndiname=jdbc//DSVOHI
```

#### 5.1.2 hsl.<app>.jndiname

---

The JNDI name of the data source to connect this HSL application to the OHI database.

If not set, this value defaults to the value of the `hsl.jndiname` property

Setting `hsl.<app>.jndiname` allows you to use different datasources for different HSL applications. A different datasource may connect to the same database using a different account, or to a different database altogether.

As an example, you may want to use `HSL_PRD` for the REL service and `HSL_RO` ('read only') for the POL service to avoid changes to the policies in the production database.

Note that you must use // for each forward slash in the JNDI name.

Example:

```
hsl.rel.jndiname=HSL_BDDEV1622
```

### 5.1.3 hsl.usercontext

---

The OHI officer (Dutch: functionaris) on whose behalf a request is executed.

The user context is inserted in the call context which is included in the call to the PL/SQL implementation procedure. Note that the PL/SQL implementation may set a different OHI officer based on the request data.

This user context determines the user identity that is used for logging changes to the data, and which language is used for messages. The value must be the Oracle username of a registered BackOffice user (in Dutch: "Functionaris").

NOTE: This value does not have to match the technical account (HSL\_USER) used for the DataSource. If you do want to use HSL\_USER, make sure you register a BackOffice user with that Oracle username.

NOTE: Do not use the value "MANAGER". Records created and update by HSL functionality should be recognizable as such. Using MANAGER will make it impossible to distinguish those records from records created or updated by batch procedures and conversion scripts.

The examples in this document use HSL\_FUNC\_USER.

### 5.1.4 hsl.<app>.usercontext

---

The OHI officer on whose behalf a request is executed for this application.

If not set, this value defaults to the value of the `hsl.usercontext` property

Setting `hsl.<app>.usercontext` allows you to set an OHI officer per HSL application.

Example:

```
hsl.rel.usercontext=HSL_FUNC_USER
```

Notes:

- the usercontext must always be set, even if `hsl.<app>.usercontext.control` is set to TOKEN
- the user context from the `hsl.properties` file may be overwritten in the SQL at the HSL application level. This should be documented in the functional specification(s) which apply to the given HSL application.

### 5.1.5 hsl.developermode

---

For security reasons, a response for a failed request contains minimal information so that potential hackers cannot use this information to misuse the HSL services. The original error message is written to the log file and replaced with 'Non-disclosed'.

If `hsl.developermode` is set to 'true', the response for a failed request contains the original error message.

Note that in production mode it is strongly advised to delete the `hsl.developermode` property from the `hsl.properties` file.

See **Doc[2]** ('Error Handling') for the differences in error handling between developer mode and non-developer mode.

## 5.1.6 hsl.<app>.developermode

---

The developer mode setting for this HSL application.

If not set, this value defaults to the value of the `hsl.developermode` property

## 5.1.7 hsl.<app>.logfile

---

The logfile for this HSL application.

Default value: `hsl.<app>.log` in the WLS domain directory.

Note:

- the directory referenced in `hsl.<app>.logfile` must exist
- the directory referenced in `hsl.<app>.logfile` must be writable to the OS user running the WLS application server.

Example:

```
hsl.rel.logfile=/home/oracle/hsl.rel.log
```

## 5.1.8 hsl.<app>.loglevel

---

The severity level of which logging should be written.

Default value: SEVERE

Logging levels: SEVERE, WARNING, INFO, CONFIG, FINE, FINER or FINEST.

The following logging levels are currently used: SEVERE, FINE, FINER and FINEST. The logging levels FINE, FINER and FINEST should only be used for debugging.

Example:

```
hsl.rel.loglevel=SEVERE
```



**WARNING:** When setting the loglevel to FINE, FINER or FINEST this may lead to extensive log messages being recorded which can slow down the processing of service requests considerably. Response times measured while using such detailed log levels are clearly affected and should not be considered as representative for regular use.

## 5.1.9 hsl.<app>.log.limit

---

The maximum size of the log file in bytes.

Default value: 1000000 (1Mb)

When the size of the log file reaches this limit, the log is rolled over to the next log file.

Note that a value of 0 means 'unlimited'.

Example:

```
hsl.rel.limit=5000000
```

### 5.1.10 hsl.<app>log.count

---

The number of log files to use in the log file rotation.

Default value: 1

A value of 1 means that only 1 log file is created and no log rotation takes place. When the log.limit is reached, the log file is overwritten and its previous contents are lost.

Set the log.count to 2 or higher to avoid overwriting the log file once it is full.

Example

```
hsl.rel.count=2
```

### 5.1.11 hsl.<app>.log.append

---

Configure if logging can be appended to existing log files.

Default value: true

If false, a new log file will be created when rotating log files.

Example:

```
hsl.rel.append=false
```

### 5.1.12 Activating changes to hsl.properties

---

To activate changes to hsl.properties you must restart the managed server.

### 5.1.13 Troubleshooting hsl.properties

---

Note the following if you have trouble starting up with a new hsl.properties file:

- an empty value for ANY property will block any HSL application from starting up.  
Example:  
`hsl.rel.jndiname=`
- lines starting with '#' are ignored.
- empty lines are ignored
- do not use whitespace characters in property lines. Whitespace characters are tabs and spaces. Inserting whitespace characters may result in a malfunction in the operation of HSL services.

### 5.1.14 Example hsl.properties file

---

The following properties file is for documentation purposes only, it needs to be adjusted to your situation and requirements.

With the release of version 10.18.1.2.0 of OHI Back Office the six services below may be used in your properties file.

```
hsl.cla.jndiname=jdbc//DSVOHI
hsl.cla.usercontext=HSL_FUNC_USER
The name of the logfile for logging messages
hsl.cla.logfile=/u01/app/oracle/product/OHI/vohi/HSL_CLA.log
SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST, OFF, ALL
```



```

hsl.cla.loglevel=SEVERE
hsl.cla.log.limit=1000000
hsl.cla.log.count=2
hsl.cla.log.append=true

hsl.pol.jndiname=jdbc//DSVOHI
hsl.pol.usercontext= HSL_FUNC_USER
The name of the logfile for logging messages
hsl.pol.logfile=/u01/app/oracle/product/OHI/vohi/HSL_POL.log
SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST, OFF, ALL
hsl.pol.loglevel=SEVERE
hsl.pol.log.limit=1000000
hsl.pol.log.count=2
hsl.pol.log.append=true

hsl.rel.jndiname=jdbc//DSVOHI
hsl.rel.usercontext= HSL_FUNC_USER
The name of the logfile for logging messages
hsl.rel.logfile=/u01/app/oracle/product/OHI/vohi/HSL_REL.log
SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST, OFF, ALL
hsl.rel.loglevel=FINEST
hsl.rel.log.limit=1000000
hsl.rel.log.count=2
hsl.rel.log.append=true

hsl.c2b.jndiname=jdbc//DSVOHI
hsl.c2b.usercontext= HSL_FUNC_USER
The name of the logfile for logging messages
hsl.c2b.logfile=/u01/app/oracle/product/OHI/vohi/HSL_C2B.log
SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST, OFF, ALL
hsl.c2b.loglevel=FINEST
hsl.c2b.log.limit=1000000
hsl.c2b.log.count=2
hsl.c2b.log.append=true

hsl.aun.jndiname=jdbc//DSVOHI
hsl.aun.usercontext= HSL_FUNC_USER
The name of the logfile for logging messages
hsl.aun.logfile=/u01/app/oracle/product/OHI/vohi/HSL_AUN.log
SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST, OFF, ALL
hsl.aun.loglevel=FINEST
hsl.aun.log.limit=1000000
hsl.aun.log.count=2
hsl.aun.log.append=true

hsl.auz.jndiname=jdbc//DSVOHI
hsl.auz.usercontext= HSL_FUNC_USER
The name of the logfile for logging messages
hsl.auz.logfile=/u01/app/oracle/product/OHI/vohi/HSL_AUR.log
SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST, OFF, ALL
hsl.auz.loglevel=FINEST
hsl.auz.log.limit=1000000
hsl.auz.log.count=2
hsl.auz.log.append=true

```

See Appendix F “*HSL\_AUN and HSL\_AUZ Services*” to determine if you need to add the last two services.

### 5.1.15 Keeping hsl.properties up to date

---

As new HSL services are released through (patch) releases of OHI Back Office, the installation instructions will tell you to change the hsl.properties file if required. When a new version of this manual is released the example properties file of the previous paragraph will be adjusted.

## 5.2 Examining the Log File

When encountering long-running HSL operations, examining the log file allows you to break down the roundtrip into different components.

Ensure that the log level for the HSL application is set to FINE.

If the log level is set to FINEST, writing log messages alone may require significant time and may account for much of the time spent in the HSL operation.

If you changed the log level you must restart the Managed Server to activate the new log properties.

Next, look up the long-running operation in the log file. The example shows log messages of a fictitious operation:

```
Mar 08, 2018 5:09:39 PM com.oracle.insurance.ohibo.hpo.HpoService
getDossierRegels
FINE: begin getDossierRegels
Mar 08, 2018 5:09:39 PM com.oracle.insurance.ohibo.hpo.HpoService getLanguage
FINE: getLanguage() returns: nl-NL
Mar 08, 2018 5:09:39 PM com.oracle.insurance.ohibo.hpo.HpoService
getDossierRegels
FINE: expand=all
Mar 08, 2018 5:09:39 PM com.oracle.insurance.ohibo.hpo.HpoService
getDossierRegels
FINE: limit=10000
Mar 08, 2018 5:09:39 PM com.oracle.insurance.ohibo.hpo.HpoService
getDossierRegels
FINE: number=35
Mar 08, 2018 5:09:39 PM com.oracle.insurance.ohibo.hpo.HpoService
getDossierRegels
FINE: offset=0
Mar 08, 2018 5:09:39 PM com.oracle.insurance.ohibo.hpo.CCallContext
toJDBCObject
FINE: enter toJDBCObject
Mar 08, 2018 5:09:40 PM com.oracle.insurance.ohibo.hpo.CCallContext
toJDBCObject
FINE: leave toJDBCObject
Mar 08, 2018 5:09:41 PM com.oracle.insurance.ohibo.hpo.HpoService
getDossierRegels
FINE: Before calling PL/SQL operation
Mar 08, 2018 5:09:59 PM com.oracle.insurance.ohibo.hpo.HpoService
getDossierRegels
FINE: After calling PL/SQL operation
Mar 08, 2018 5:10:00 PM com.oracle.insurance.ohibo.exception.ExceptionUtil
handleReturnContext
FINE: start handleReturnContext
Mar 08, 2018 5:10:00 PM com.oracle.insurance.ohibo.exception.ExceptionUtil
handleReturnContext
FINE: end handleReturnContext
Mar 08, 2018 5:10:00 PM com.oracle.insurance.ohibo.hpo.HpoService
getDossierRegels
FINE: Before mapping SQL object to Java object
Mar 08, 2018 5:10:21 PM com.oracle.insurance.ohibo.hpo.HpoService
getDossierRegels
FINE: After mapping SQL object to Java object
Mar 08, 2018 5:10:21 PM com.oracle.insurance.ohibo.hpo.HpoService
getDossierRegels
FINE: http code=200
Mar 08, 2018 5:10:21 PM com.oracle.insurance.ohibo.hpo.HpoService
getDossierRegels
FINE: Before creating response
Mar 08, 2018 5:10:22 PM com.oracle.insurance.ohibo.hpo.HpoService
getDossierRegels
FINE: After creating response
Mar 08, 2018 5:10:22 PM com.oracle.insurance.ohibo.hpo.HpoService
getDossierRegels
FINE: end getDossierRegels
```

From this fragment we may derive the following information:

- Total roundtrip is about 43s (5:09:39 - 5:10:22)
- PL/SQL execution: 18s (5:09:41 - 5:09:59)
- Mapping SQL object to Java object:<1s
- Creating response with JSON string:<1s

## 5.2.1 Changing the log format

---

The default format for logging timestamps is not suitable for sub-second operations. Logging timestamps in milliseconds since 01-01-1970 is needed if you want to analyse sub-second operations.

To override the default format, create a configuration file with the following contents:

```
override default format for timestamps in milliseconds since 01-01-1970.
java.util.logging.SimpleFormatter.format=%1$tQ %2$s%n%4$s: %5$s%6$s%n
```

You now need to activate this configuration for the managed server to which the HSL application is deployed:

- Start WebLogic Console
- Choose Environments > Servers > *managed\_server*
- Add `-Djava.util.logging.config.file=your_config_file` to the Server Start parameters. . Add a line to the file `$DOMAIN_HOME/bin/setUserOverrides.sh` you created earlier. Add the line to the part for the HSL server:

```
JAVA_OPTIONS="-Djava.util.logging.config.file="your_config_file" {JAVA_OPTIONS}"
```

- Restart the managed server.
- Call the HSL operation and check that the subsequent log messages show log messages in milliseconds since 01-01-1970

The output should now look like:

```
1520867075960 com.oracle.insurance.ohibo.hpo.HpoService getDossierRegels
FINE: begin getDossierRegels
1520867075971 com.oracle.insurance.ohibo.hpo.HpoService getLanguage
FINE: getLanguage() returns: nl-NL
1520867075974 com.oracle.insurance.ohibo.hpo.HpoService getDossierRegels
FINE: expand=all
1520867075974 com.oracle.insurance.ohibo.hpo.HpoService getDossierRegels
FINE: limit=10000
1520867075975 com.oracle.insurance.ohibo.hpo.HpoService getDossierRegels
FINE: number=11
1520867075975 com.oracle.insurance.ohibo.hpo.HpoService getDossierRegels
FINE: offset=0
1520867075976 com.oracle.insurance.ohibo.hpo.CCallContext toJDBCObject
FINE: enter toJDBCObject
1520867075977 com.oracle.insurance.ohibo.hpo.CCallContext toJDBCObject
FINE: leave toJDBCObject
1520867075978 com.oracle.insurance.ohibo.hpo.HpoService getDossierRegels
FINE: Before calling PL/SQL operation
1520867092723 com.oracle.insurance.ohibo.hpo.HpoService getDossierRegels
FINE: After calling PL/SQL operation
1520867092728 com.oracle.insurance.ohibo.exception.ExceptionUtil
handleReturnContext
FINE: start handleReturnContext
1520867092729 com.oracle.insurance.ohibo.exception.ExceptionUtil
handleReturnContext
FINE: end handleReturnContext
1520867092730 com.oracle.insurance.ohibo.hpo.HpoService getDossierRegels
FINE: Before mapping SQL object to Java object
1520867093066 com.oracle.insurance.ohibo.hpo.HpoService getDossierRegels
FINE: After mapping SQL object to Java object
1520867093068 com.oracle.insurance.ohibo.hpo.HpoService getDossierRegels
FINE: http code=200
1520867093072 com.oracle.insurance.ohibo.hpo.HpoService getDossierRegels
FINE: Before creating response
1520867093073 com.oracle.insurance.ohibo.hpo.HpoService getDossierRegels
FINE: After creating response
1520867093074 com.oracle.insurance.ohibo.hpo.HpoService getDossierRegels
FINE: end getDossierRegels
```

This log output of a fictitious operation gives us the following information:

- Total roundtrip is 17114 ms (1520867093074 - 1520867075960)
- PL/SQL execution: 16745 ms (1520867092723 - 1520867075978)
- Mapping SQL object to Java object: 336 ms (1520867093066 - 1520867092730)

- Creating response with JSON string: 1 ms

## 6 Upgrading HSL services

Future OHI releases may include new WAR files for HSL services.

To deploy a new version of an existing HSL application, follow the steps below:

- ✓ Check your web service properties file (typically `hsl.properties`) and implement necessary changes for your release. For information about the contents please see the previous chapter.
- ✓ Logon to the Admin Server console of the domain where the web services are deployed.
- ✓ Navigate to the deployments pane.
- ✓ Choose the 'Lock & Edit' option.
- ✓ If you already have a Retired version of the deployment, mark the check box in front of the retired deployment and delete it.
- ✓ Navigate to the deployment that must be updated and mark the check box in front of it.
- ✓ Press the Update button.
- ✓ Determine whether the same source path still applies (typically a new version is delivered in the `$OZG_BASE/java` folder of your environment but your organisation may have additional distribution methods implemented). When the correct `.war` file is selected press Next.
- ✓ You now have two options for 'retiring' the previous version. Because normally the Back Office application is not available during patching, you can retire the previous version 'immediately', meaning using a timeout of 1 second:

How would you like to retire the previous version of this application?

Allow the application to finish its current sessions and then retire.

Retire the previous version after retire timeout.

Retire timeout (seconds):

Press 'Finish' to retire the previous version and continue.

- ✓ Choose 'Activate Changes'.
- ✓ Refresh the screen a few seconds after having activated the changes.
- ✓ Inform the communities which use the web services of the availability and publish the latest URI's to the swagger definitions to them.

If the old deployment cannot be deleted when updating, stop the deployment with the 'Force' option and deploy it again completely (using the 'Install' option for deployments). In some cases (depending on the changes) you may need to repeat the Deployment delete/install when the install results in errors. If the deployment keeps failing, you may have to restart the Managed Server(s) as a last resort.

After this the deployment state of the web services should be Active again (be sure the Managed Server(s) is/are running, otherwise start it/them to get this result).

If not, check whether your OHI database environment and deployed version are correct, meaning that their version levels correspond with each other.

## 7 Appendix A – Service Information

The following URI provides version information about a running HSL application:

```
https://server:port/application/dbinfo
```

For example:

```
https://localhost:7002/HSL_POL/dbinfo
```

This will return a JSON object like below:

```
{
 "basePath": "https://localhost:7002/HSL_POL/pol",
 "database": "BTSPC12",
 "instance": "CDB02",
 "jndiName": "HSL_BTSPC12",
 "plsqlPackage": "hsl_pol_sp_pck $Revision: 4.39 $",
 "user": "HSL_USER",
 "userContext": "MANAGER"
}
```

Information:

- **basePath**  
Format: `https://server:port/application/context`  
This is the base URI for all operations in this service.
- **database**  
The name of the database associated with the current database connection
- **instance**  
Instance name of the database associated with the current database connection.
- **jndiName**  
The JNDI name of the database connection (specified in the `hsl.properties` file)
- **plsqlPackage**  
The PL/SQL package which implements the operations of the HSL service.  
In this release, the revision number refers to the revision number of the code template used to generate the PL/SQL package. In a future release this will point to the minimum revision number of the compiled PL/SQL package.
- **user**  
The database account used to log on to the database.
- **user context**  
The default OHI officer on whose behalf service operations are performed, as specified in the `hsl.properties` file.

---

## 8 Appendix B – Removing a WLS domain

In case you want to restructure your environment or recreate a domain you can remove an existing domain.

In order to do this make sure all servers for the domain are stopped and make sure there is no Node Manager process running which 'guards' this domain.

Next perform the following actions:

- ✓ Completely remove your domain directory including all contents.
- ✓ Remove any reference in start and stop scripts to this domain.
- ✓ Remove, if present, the domain from the <WebLogic home>\oracle\_common\common\nodemanager\nodemanager.domains.
- ✓ Remove the domain from the domain-registry.xml file which is located in the Middleware home folder (\$MW\_HOME).

For more information please use the standard WebLogic documentation.



## 9 Appendix C – Testing with SoapUI

SoapUI is a tool for testing web services which can be downloaded from <http://www.soapui.org>.

It is especially useful for functional testing of the HSL application.

WLS 12c has removed the support for the security protocols SSLv3 and TLS 1.1, because they are now considered insecure.

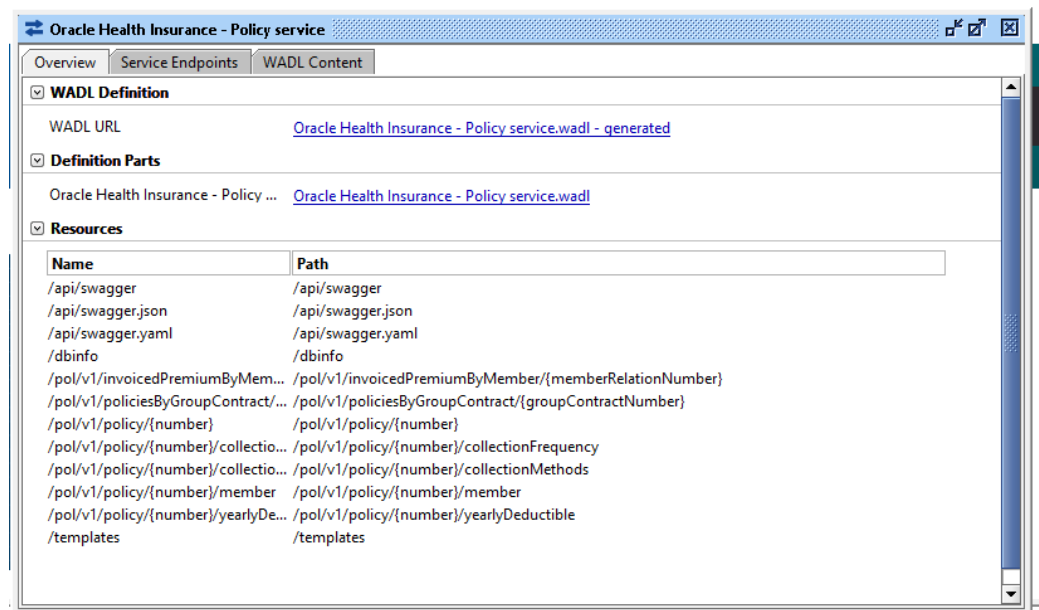
This means you must access the HSL application with a client that uses TLS 1.2.

Unlike earlier versions, SoapUI 5.3 and 5.4 enable TLS 1.2 by default. The examples below assume SoapUI 5.4 or higher.

### 9.1 Create REST project and import Swagger definition

- Follow the instructions in ‘Get Online Swagger Definition (curl)’ to retrieve the online Swagger definition from the HSL application and save the output to a file (for example ‘saved\_swagger.json’)
- Create a new REST project (empty value for URL)
- Choose ‘Project > Import Swagger’ and select the saved Swagger definition.

The operations of the HSL application are now discovered:



You may now create requests for the operations provided by this HSL application.

### 9.2 Create a request

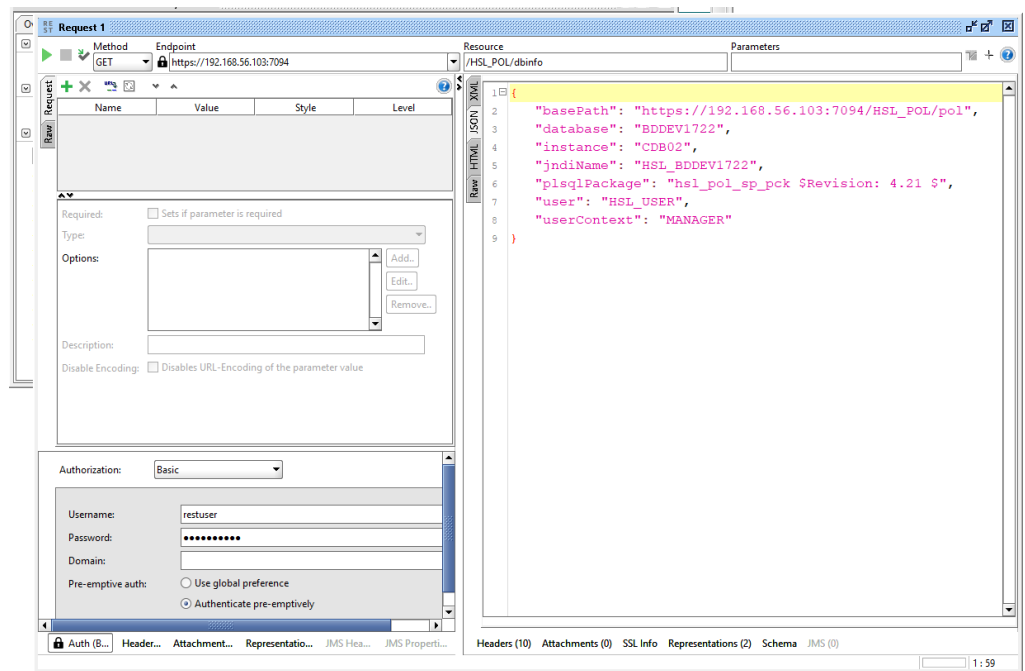
Once you have imported the Swagger definition you may create a request for each of the operations.

In the example below we create a request for the getDatabaseInfo operation:

- Double-click on ‘Request 1’ of the requested operation (in our case /dbinfo > getDatabaseInfo).

- Set the endpoint for the request to `https://<server>:<port>`  
For example <https://127.0.0.1:7094>.
- Select 'Headers' and add a HTTP request header with Header value 'Accept' and with Value value 'application/json'.
- Add other HTTP request headers as required (not needed for this example)
- Select 'Auth' to add Basic Authentication for the WLS user.  
If you deployed with the 'DD Only' deployment model the WLS user should be 'restuser'.
- Set 'preemptive authentication'.
- Run the request.

The request window should now look like this:



## 10 Appendix D - Generating a WADL file

A WADL (Web Application Description Language) file may be required by Oracle Service Bus or other middleware to describe your HSL application.

The current HSL applications cannot be used to generate WADL files directly.

However, a WADL file can be easily generated from the online Swagger definition using SoapUI.

This involves the following steps:

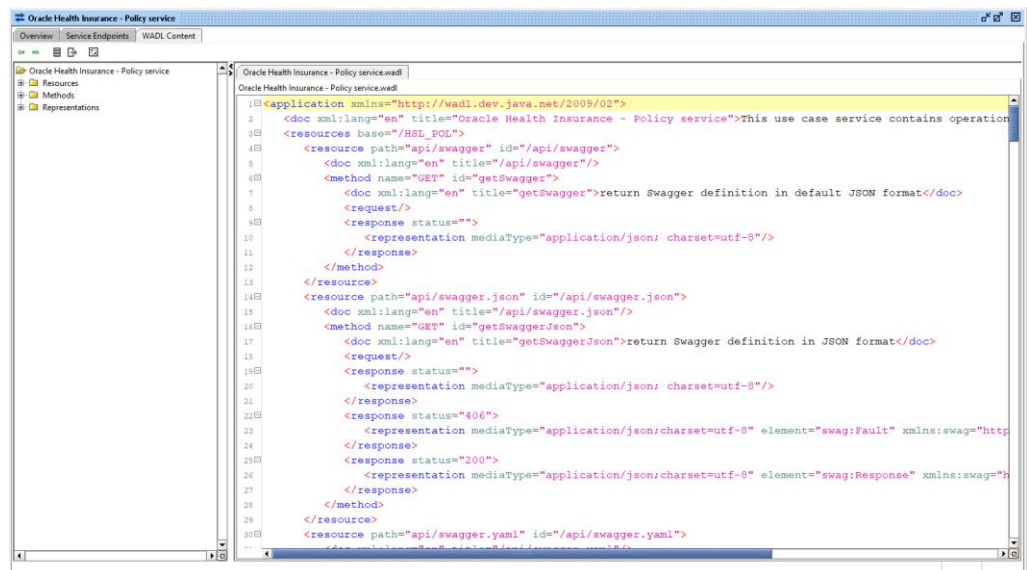
- Create a REST project in SoapUI for your HSL application
- Open the Service Viewer for the REST project
- Export WADL from your REST project

### 10.1 Create a REST project in SoapUI for your HSL application

Follow the instructions in 'Testing with SoapUI' to set up SoapUI for testing with your HSL application.

### 10.2 Open the Service Viewer for the REST Project

Click on the 'WADL Content' to see the WADL description.  
Your screen may look like this:

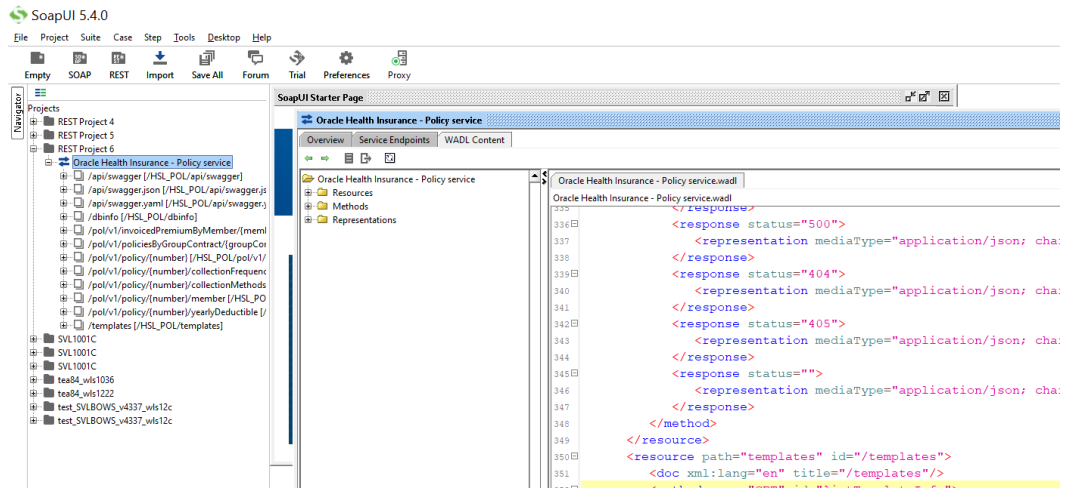


```
Oracle Health Insurance - Policy service.wadl
1 <!DOCTYPE wadl [
2 <doc xml:lang="en" title="Oracle Health Insurance - Policy service">This use case service contains operation
3 </doc>
4 <resources base="/HSL_POL">
5 <resource path="/api/swagger" id="/api/swagger">
6 <doc xml:lang="en" title="/api/swagger"/>
7 <method name="GET" id="getSwagger">
8 <doc xml:lang="en" title="getSwagger">return Swagger definition in default JSON format</doc>
9 </method>
10 <request/>
11 <response status="">
12 <representation mediaType="application/json; charset=utf-8"/>
13 </response>
14 </resource>
15 <resource path="/api/swagger.json" id="/api/swagger.json">
16 <doc xml:lang="en" title="/api/swagger.json"/>
17 <method name="GET" id="getSwaggerJson">
18 <doc xml:lang="en" title="getSwaggerJson">return Swagger definition in JSON format</doc>
19 </method>
20 <request/>
21 <response status="">
22 <representation mediaType="application/json; charset=utf-8"/>
23 </response>
24 <response status="406">
25 <representation mediaType="application/json; charset=utf-8" element="swag:Fault" xmlns:swag="http://www.swagger.io/2015-09-08/swagger">
26 </representation>
27 </response>
28 <response status="200">
29 <representation mediaType="application/json; charset=utf-8" element="swag:Response" xmlns:swag="http://www.swagger.io/2015-09-08/swagger">
30 </representation>
31 </response>
32 </method>
33 </resource>
34 <resource path="/api/swagger.yaml" id="/api/swagger.yaml">
35 <doc xml:lang="en" title="/api/swagger.yaml"/>
36 <method name="GET" id="getSwaggerYaml">
37 <doc xml:lang="en" title="getSwaggerYaml">return Swagger definition in YAML format</doc>
38 </method>
39 <request/>
40 <response status="">
41 <representation mediaType="application/yaml; charset=utf-8"/>
42 </response>
43 </resource>
44 </resources>
45 </wadl>
```

### 10.3 Export WADL from your REST project

Save the buffer to a WADL file.

Alternatively you may right-click on the service within the REST project (highlighted in the screen shot below):



And select 'Export WADL' to create the WADL for this application.

## 11 Appendix E – Authentication and Authorization

In 10.18.1.0.0 the following changes were made to the security of the HSL services:

- No authentication needed for OPTIONS method
- OAUTH 2.0 token authentication and validation as an alternative to Basic Authentication

### 11.1 HTTP OPTIONS method

The HTTP OPTIONS method is used by browsers as a pre-flight check to retrieve the allowable methods for a given URL. This check, for which no authentication is needed, is part of the CORS mechanism.

### 11.2 OAUTH 2.0 token authentication and validation

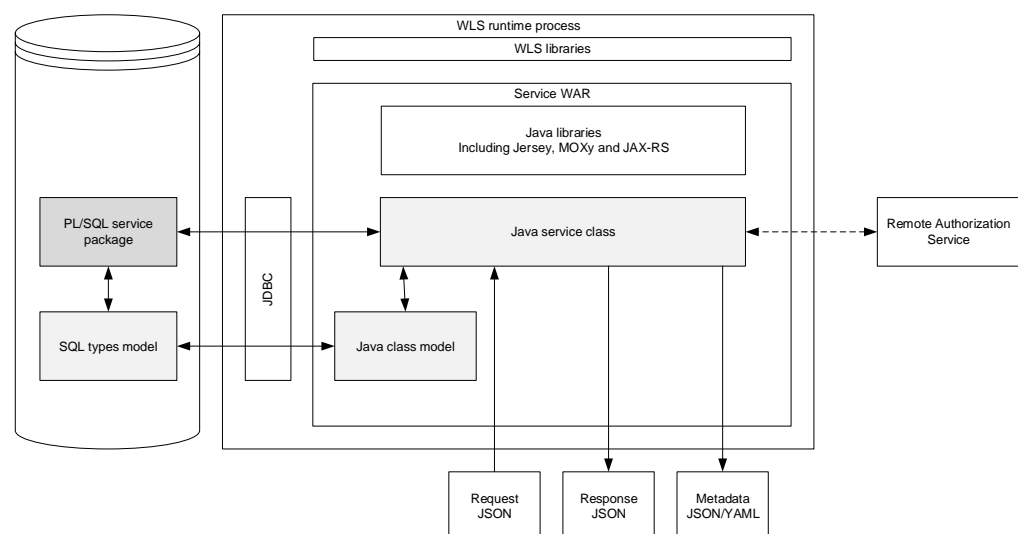
If Basic Authentication is used, the 'Authorization' header starts with 'Basic' followed by a base64-encoded username/password combination for a valid Weblogic account. Basic authentication is still the default mechanism for HSL services.

As an alternative for Basic Authentication, support for OAUTH 2.0 has been developed for the HSL services:

- Authentication based on access-token
- Token validation using an Authorization Service
- Set OHI officer (optional)

If access-token authorization is used, the 'Authorization' header starts with 'Bearer' followed by an encoded string which must be validated by a remote authorization service.

The token validation using a remote authorization service has been indicated with a dotted line in the HSL application architecture diagram:



#### 11.2.1 WAR File Deployment

In order to use access token validation (OAUTH2), the HSL application must be deployed with 'Custom Roles and Policies'.

## 11.2.2 Configuration

---

Authentication and authorization are configured through the `hsl.properties` file. To implement the changes in the `hsl.properties` file, the managed server(s) associated with the WAR file must be restarted.

## 11.2.3 Which Authorization Method?

---

The `hsl.app.authorization` property selects which authorization methods are allowed for a HSL service. This value defaults to the value of `hsl.authorization`.

If `hsl.authorization` is not set, the default value of 'BASIC' is used to enforce Basic Authentication

Allowed values:

- BASIC - use 'Basic' HTTP Authorization header with WLS credentials
- TOKEN - use 'Bearer' HTTP Authorization header with JWT token
- Or a combination of these.

## 11.2.4 Access Token Validation

---

To support the OAUTH2 access token validation, the HSL service must be configured to call a remote authorization service, which validates:

- that the caller is a legitimate user of the system
- that the caller is authorized for the requested HSL service operation (expressed through path+method)

The HSL service which is called by the client application now dynamically creates the call to the remote authorization service.

The remote authorization should meet the following criteria:

- The authorization service can be invoked using the HTTP protocol.
- POST or GET is used as the default method to invoke remote authorization
- Basic authentication may be used to call the remote authorization service
- JSON is used to format the body parameter

To test this functionality, the HSL\_AUN and HSL\_AUZ services were developed. See 'Appendix F' for a description.

The following `hsl.properties` parameters are used to configure the call:

- `hsl.app.tokenvalidation.url` - defaults to the value of `hsl.tokenvalidation.url`  
The URL of the authorization service.
- `hsl.app.tokenvalidation.method` - defaults to the value of `hsl.tokenvalidation.method`  
Method to access the authorization service operation  
Default value: POST
- `hsl.app.tokenvalidation.headerparams` - defaults to the value of `hsl.tokenvalidation.headerparams`

A template with place holders which is used to add HTTP request headers when calling the remote authorization service.

- `hsl.app.tokenvalidation.queryparams` - defaults to the value of `hsl.tokenvalidation.queryparameters`  
A template with placeholders which is used to construct a query string.
- `hsl.app.tokenvalidation.bodyparam` - defaults to the value of `hsl.tokenvalidation.bodyparam`  
A template with placeholders which is used to create a JSON string which is used as a body parameter.
- `hsl.app.tokenvalidation.authentication` - defaults to the vaule of `hsl.tokenvalidation.authentication`  
Contents for the Authorization header which is used to authenticate the request to the remote authorization service.

Note that the dynamically constructed request may be created from

- `hsl.app.tokenvalidation.headerparams`
- `hsl.app.tokenvalidation.bodyparm`
- `hsl.app.tokenvalidation.queryparams`
- or a combination of these.

### 11.2.5 Place Holders

---

The following placeholders are expanded before calling the remote authorization service:

- `#path#`  
Path to the service operation for which the token validation is desired. The path value is derived from the HTTP request for the service operation.
- `#method#`  
Method (GET,POST,PUT,PATCH,DELETE) of the service operation. The method value is derived from the HTTP request for the service operation.
- `#jwt#`  
Contents of the access token which must be passed to the remote authorization service.  
This is the contents of the 'Authorization' header of the original request after removing the 'Bearer\s' prefix.

### 11.2.6 POST Example

---

```
hsl.tokenvalidation.authentication=Basic cmVzdHVzZXI6b3Blbnpvcmc50Q==
hsl.tokenvalidation.bodyparam={ "method" : "#method#", "token" : "#jwt#" ,
"resource" : "#path#" }
hsl.tokenvalidation.method=post
hsl.tokenvalidation.url=https://ol6ohi.us.oracle.com:7110/HSL_AUZ/auz/v1/authorizati
on/verify
```

### 11.2.7 GET example

---

Fictitious example to configure a verify operation using the GET method:

```
hsl.hba.tokenvalidation.url=https://ol6ohi.ohi.oracle.com:7094/
HSL_HBA/hba/v1/authorization/verify
hsl.hba.tokenvalidation.method=get
```

```
hsl.hba.tokenvalidation.queryparams=?id=123&HTTPverb=#method#&t
oken=#jwt#&resource=#path#
hsl.hba.tokenvalidation.authentication=Basic
cmVzdHVzZXI6b3Blbnpvcmc5OQ==
hsl.hba.tokenvalidation.headerparams=hdr1:value1 hdr2:value2
```

## 11.2.8 Setting user context

---

Every operation of a REST service must be executed by a OHI officer account (Dutch: functionaris). This is a registered user of the OHI BackOffice application).

The default account is set through `hsl.app.usercontext` (defaults to `hsl.usercontext`).

If token validation is used, the `usercontext` is retrieved from the JWT access token.

The following configuration parameters are used:

- `hsl.app.usercontext` - defaults to the value of `hsl.usercontext`  
The user context which must be used for executing an operation.
- `hsl.app.usercontext.control` - defaults to the value of `hsl.usercontext.control`  
Allowed values:
  - PROPERTY  
Use the value of `hsl.app.usercontext` to set the user context.
  - TOKEN  
Retrieve the user context from the access token.  
Note that `hsl.app.usercontext.claim` must be set to indicate which field contains the `usercontext`.

If `hsl.app.usercontext.control` is set to `TOKEN`, the following configuration parameters control how the `usercontext` is retrieved:

- `hsl.app.usercontext.token.type` - defaults to the value of `hsl.usercontext.token.type`  
Set the type of token.  
Allowed values: `JWT`
- `hsl.app.usercontext.claim` - defaults to the value of `hsl.usercontext.claim`  
Determines which field in the JWT token contains the `usercontext`.  
Example: `hsl.app.usercontext.claim=prn`

Note that `hsl.app.usercontext` must always be set to a valid default value in order to retrieve the Swagger definition.

## 11.2.9 Overriding User Context with Back Office Parameter

---

For several HSL services, a Back Office parameter (Dutch: functionaris) has been created to override the user context.

If the Back Office parameter for setting the user context for a specific service has been set it will overrule the user context as set at the start of the service operation!

If you want ensure that the user context is set by the `hsl.app.usercontext` parameter or through the access token, you should remove the value of the 'functionaris' parameter for the given service through the OHI BO application.



---

## 12 Appendix F - HSL\_AUN and HSL\_AUZ Services

Two new HSL services were developed for testing the OAUTH2 support of HSL services:

- **HSL\_AUN**  
This service authenticates the username/password account of a database account and, if successful, returns an access token in JWT format.
- **HSL\_AUZ**  
After verifying that the JWT-formatted access token was not compromised, this service should verify that the OHI officer referenced in the token (acquired from HSL\_AUN) is authorized for the requested service operation.

The HSL\_AUN and HSL\_AUZ services may be used by customers to test the OAUTH2 support in the HSL services.

---

### 12.1.1 Disclaimer

Note that HSL\_AUN and HSL\_AUZ are only for testing! You can use them as mock-up services to imitate a real OAUTH2 implementation, but should NOT use them to authenticate or authorize requests in a production environment.

Also note that at the time of writing the implementation of the 'postVerify' operation in HSL\_AUZ is incomplete.

## 12.2 Use of JWT

JWT (JSON Web Token) is emerging as a standard format for access tokens. A JWT is a base64-encoded string consisting of three parts separated by '.' characters:

- **header**  
Contains encryption method and token type (JWT)
- **payload**  
Contains principal and claims (privileges)  
The principal for HSL\_AUN and HSL\_AUZ is the 'OHI officer' (aka 'functionaris').
- **signature**  
Checksum based on encrypted header + payload

---

### 12.2.1 Payload

The payload may contain the following attributes:

- **exp** - expirydate in number of seconds since 01-01-1970  
When the token is issued this is the system date + one year.
- **iss** - token issuer  
Hardcoded: [www.oracle.com](http://www.oracle.com)
- **prn** - the username of the OHI officer making the request.  
Example: HSL\_FUNC\_USER
- **name** - The name of the OHI officer.  
Example: "HSL Web Services"

- claims - a list of modules which can be started by the OHI officer.

## 12.2.2 Token Verification

---

The JWT signature contains the encrypted concatenation of the header and payload when the token was issued by HSL\_AUN.

When verifying the access token, HSL\_AUZ.postVerify recalculates the signature using the header and payload. For this it uses the same algorithm as HSL\_AUN.postToken. If the new signature is different from the original signature, the token verification will fail.

## 12.3 HSL Properties for Signature Encryption

The encryption algorithm used by HSL\_AUN and HSL\_AUZ is driven by the HSL property `hsl.token.validation.rotor=your_secret_key`

Keep the value of `hsl.token.validation.rotor` secret. Although HSL\_AUN and HSL\_AUZ are not currently in production mode, this may change in the future.

## 12.4 HSL\_AUN Authentication Service

The HSL\_AUN service has a single operation 'postToken' to

- log in to the (OHI Back Office) database using the username and password passed through the 'Credentials' resource.
- Issue a access token in JWT format  
The 'claims' attribute will contain a list of modules for which the principal (OHI officer) is authorized.  
Note: this list is currently empty!

### 12.4.1 HSL properties

---

In addition to the usual HSL properties (see 'Back Office HSL properties file' above), set the following HSL properties before activating HSL\_AUN:

- `hsl.aun.authorization=NONE`  
Rationale: the postToken operation uses the username and password supplied in the Credentials parameter to log into the OHI Back Office database. The operation fails if the credentials are incorrect.

## 12.5 HSL\_AUZ Authorization Service

This service has a single operation: postVerify.

### 12.5.1 postVerify operation

---

The postVerify operation verifies that:

- the access token has not expired.
- the access token has not been tampered with

The postVerify operation should also verify that the requested service operation matches with an item in the claims list in the access token. This functionality has not yet been implemented.

## 13 Appendix G – PSL services

'PSL' stands for 'Private Service Layer'. PSL services are created specifically to support OHI BO application components like the ZRGOHIJET user interface application. These services use the same technology as the HSL services but are not intended as an 'API', so may and should not be used to support custom client applications as they may be altered without any notice.

It is important to know that unless you are using an OHI BO application which requires the use of a PSL service, there is no need to install PSL services. So only deploy them when you are sure your organisation is going to use the ZRGOHIJET user interface, as this is currently the only component which needs them and which in itself is also an optional component for deployment.

Characteristics of PSL services:

- Specifically built to support OHI BO applications components. This means that PSL services are not intended to be called by customer applications. It also means that contents or operation of PSL service operations may be changed by OHI Back Office Development without notice.
- No online help documentation.
- Built on the same technology as HSL services.
- Configured through a 'psl.properties' file, similar to the 'hsl.properties' file used for HSL services.

### 13.1 Installation of PSL services

Like HSL services, PSL services should be deployed through Weblogic Application Server (WLS).

The chapter 'Installation of HSL services' applies also to the installation of PSL services.

#### 13.1.1 PSL database account

---

The PSL services require a PSL database account similar to the HSL database account:

1. Create a schema owner, for example PSL\_USER.
2. Grant create session system privilege to the PSL database account.
3. Log on as the OHI Back Office schema owner, enable serveroutput and run:

```
alg_security_pck.psl_grants
(pi_owner => '<ohibo_owner>'
,pi_grantee => '<psl_user_account>'
);
```

Example:

```
execute alg_security_pck.psl_grants
(pi_owner => 'OZG_OWNER'
,pi_grantee => 'PSL_USER');
```

The notes mentioned in 'Creating a HSL database account' also apply to the PSL database account.

### 13.1.2 WLS Domain

---

You may use the same WLS domain for PSL services as for HSL and SVL services.

### 13.1.3 Data source

---

Create a data source for PSL services along the lines of 'Creating a data source' for the HSL services. Refer to the PSL database account instead of the HSL database account.

### 13.1.4 WLS Managed Server Start arguments

---

You will need to set `-Dpsl.properties=filename`

Example:

```
-Dpsl.properties=/ohi/envBase/vohi/conf/psl.properties
```

Add the line to file `$DOMAIN_HOME/bin/setUserOverrides.sh`:

```
JAVA_OPTIONS="-Dpsl.properties="/ohi/envBase/vohi/conf/psl.properties" ${JAVA_OPTIONS}"
```

The instructions for setting the `psl.properties` parameter are similar to those for setting 'hsl.properties' as described in the 'Installation of HSL services' chapter.

## 13.2 Configuration of PSL.properties

The configuration of the `psl.properties` file is similar to that of `hsl.properties`.

Note that all properties are prefixed with 'psl' instead of 'hsl'.

Example for PSL service `acl` and `zvp` (`zvp` is added in 10.18.2.1.0):

```
psl.acl.jndiname=jdbc//DSVOHI
psl.acl.usercontext=PSL_FUNC_USER
psl.acl.usercontext.control=TOKEN
psl.acl.authorization=TOKEN
The name of the logfile for logging messages
psl.acl.logfile=/u01/app/oracle/product/OHI/vohi/PSL_ACL.log
SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST, OFF, ALL
psl.acl.loglevel=SEVERE
psl.acl.log.limit=1000000
psl.acl.log.count=2
psl.acl.log.append=true
psl.zvp.allowedorigins=http://localhost:8000, https://server.domain.com
psl.zvp.jndiname=jdbc//DSVOHI
psl.zvp.usercontext=PSL_FUNC_USER
psl.zvp.usercontext.control=TOKEN
psl.zvp.authorization=TOKEN
The name of the logfile for logging messages
psl.zvp.logfile=/u01/app/oracle/product/OHI/vohi/PSL_ACL.log
SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST, OFF, ALL
psl.zvp.loglevel=SEVERE
psl.zvp.log.limit=1000000
psl.zvp.log.count=2
psl.zvp.log.append=true
psl.zvp.allowedorigins=http://localhost:8000, https://server.domain.com
```

The value of the `psl.acl.usercontext` must be a registered OHI officer (in Dutch: 'functionaris').