

# Oracle Health Insurance Back Office

## HTTP Service Layer Installation & Configuration Manual

Version 1.3

Part number: E91383-01

December 11th, 2017

Copyright © 2017, 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"><li>• Creation</li></ul>
10.16.2.2.0	1.1	<ul style="list-style-type: none"><li>• Revision</li></ul>
10.17.1.0.0	1.2	<ul style="list-style-type: none"><li>• Changed grant instructions</li></ul>
10.17.2.0.0	1.3	<ul style="list-style-type: none"><li>• Documented hsl.&lt;app&gt;.developermode and hsl.developermode</li><li>• Added reference to Doc[2] (Back Office HTTP Service Layer User Manual)</li></ul>

## 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.....	7
1.1	Usage rights.....	7
2	Architectural overview .....	8
2.1	Services components .....	8
3	Installation of HSL services .....	10
3.1.1	Terminology .....	10
3.2	Sizing/load aspects .....	10
3.2.1	Deployment choices.....	11
3.3	Database installation .....	11
3.3.1	Creating a HSL database account .....	11
3.4	WLS Preparation.....	12
3.4.1	Requirements.....	13
3.4.2	Creating a domain .....	14
3.4.3	Creating Managed Server(s).....	16
3.4.4	Creating a machine definition.....	17
3.4.5	Creating a data source.....	19
3.5	Security Configuration.....	24
3.5.1	Set up security realm.....	24
3.5.2	Enable SSL.....	25
3.5.3	Configure JSSE .....	25
3.5.4	Setting up a key store .....	26
3.5.5	Configure logging level .....	27
3.5.6	Set user lockout .....	27
3.6	(Re)deployment of the HSL Application.....	28
3.6.1	Deploy to a single Managed Server.....	28
3.6.2	Deploy to multiple Managed Servers .....	31
3.6.3	Deploy to cluster .....	31
3.6.4	Deploy for multiple environments (DTAP) .....	31
3.6.5	Publishing the deployed services .....	32
3.7	Security Aspects.....	32
3.7.1	Restricting access with custom roles .....	33
3.7.2	Swagger definition.....	33
3.7.3	Testing with SoapUI.....	34
4	Configuration files for HSL services .....	37
4.1	Back Office HSL properties file .....	37
4.1.1	hsl.jndiname .....	37
4.1.2	hsl.<app>.jndiname.....	37
4.1.3	hsl.usercontext.....	38
4.1.4	hsl.<app>.usercontext .....	38
4.1.5	hsl.developermode .....	38
4.1.6	hsl.<app>.developermode .....	38
4.1.7	hsl.<app>.logfile .....	38
4.1.8	hsl.<app>.loglevel.....	39
4.1.9	hsl.<app>.log.limit.....	39
4.1.10	hsl.<app>.log.count.....	39
4.1.11	hsl.<app>.log.append .....	39
4.1.12	Activating changes to hsl.properties .....	40
4.1.13	Troubleshooting hsl.properties.....	40

4.1.14	Example hsl.properties file .....	40
4.1.15	Keeping hsl.properties up to date .....	40
5	Upgrading HSL services .....	41
6	Appendix A - Service Information.....	43
7	Appendix B - Removing a WLS domain.....	44

---

# 1 Introduction

The OHI Back Office HTTP Service Layer is an optional component, which offers operations to support use cases. These use cases may be scenarios for typical tasks performed by for example self-service users (insured members, care providers 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

Typically this HTTP Service Layer is targeted 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 Usage rights

Customers are required to have the appropriate license rights for using the HTTP Service Layer. Customers who have acquired a Connect to Back Office 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 or HTTP Service Layer license is obtained.

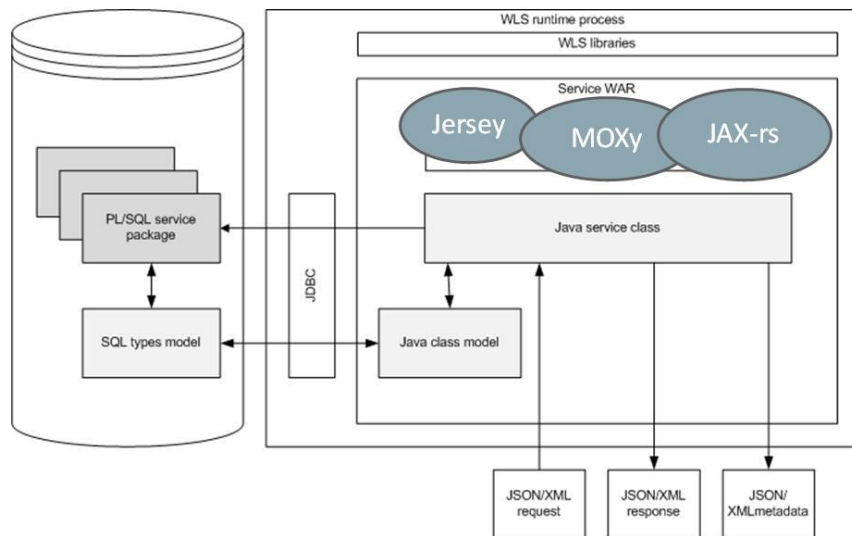
For further information please consult your OHI sales representative.

## 2 Architectural overview

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

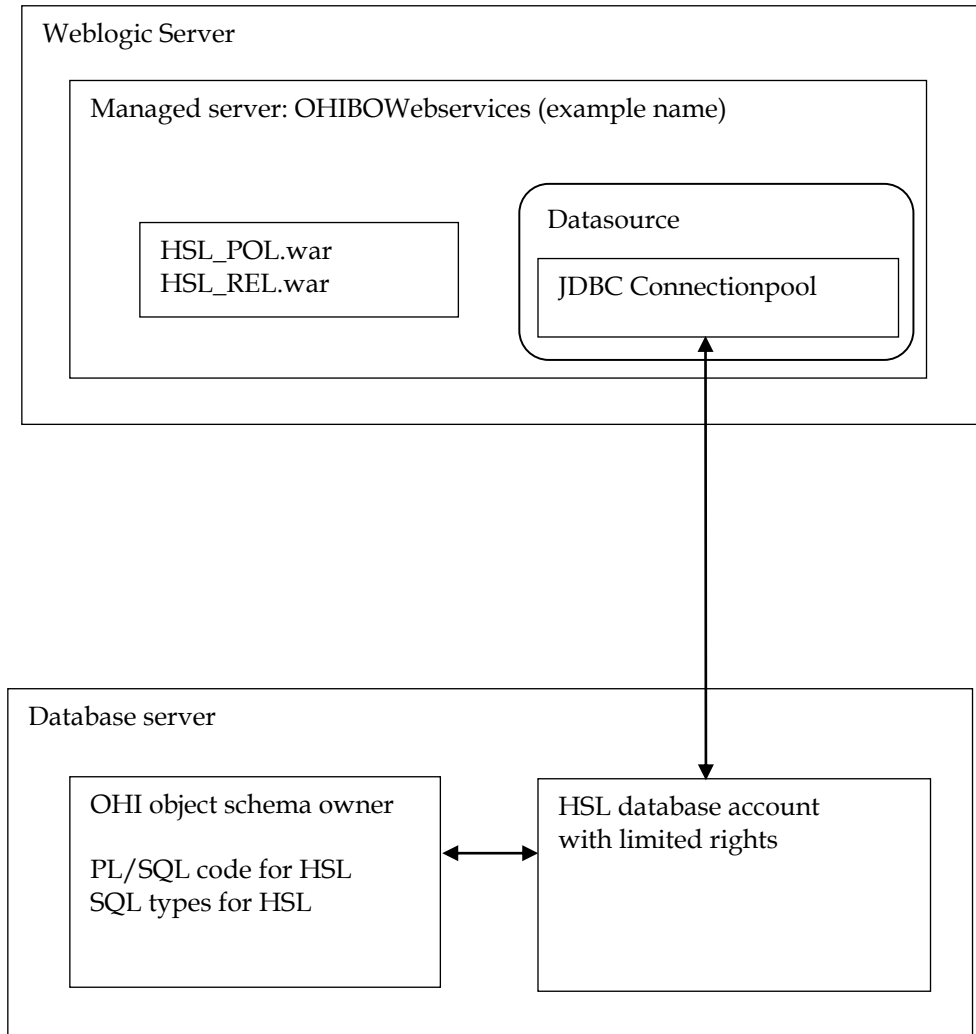
### 2.1 Services components

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 objects to and from JSON or XML and to help validate input. 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
- Security aspects
- Miscellaneous

### 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 of HSL accounts can be created:

- Primary 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.
- 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 => '<your OHI schema owner>', pi\_grantee => '<your account>')”, for example:

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

IMPORTANT: This command does not have to be repeated after each new deployment of a new .war file. During the database installation of OHI patches any existing grantees of the HSL database objects receive any required additional grants. 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 [oracle\_common]
- o Weblogic JAX-WS SOAP/JMS Extension - 12.2.1 [oracle\_common]

When you have not installed these components your web services will respond with ‘There are error messages.’ While 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

### 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 HSLservices 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.

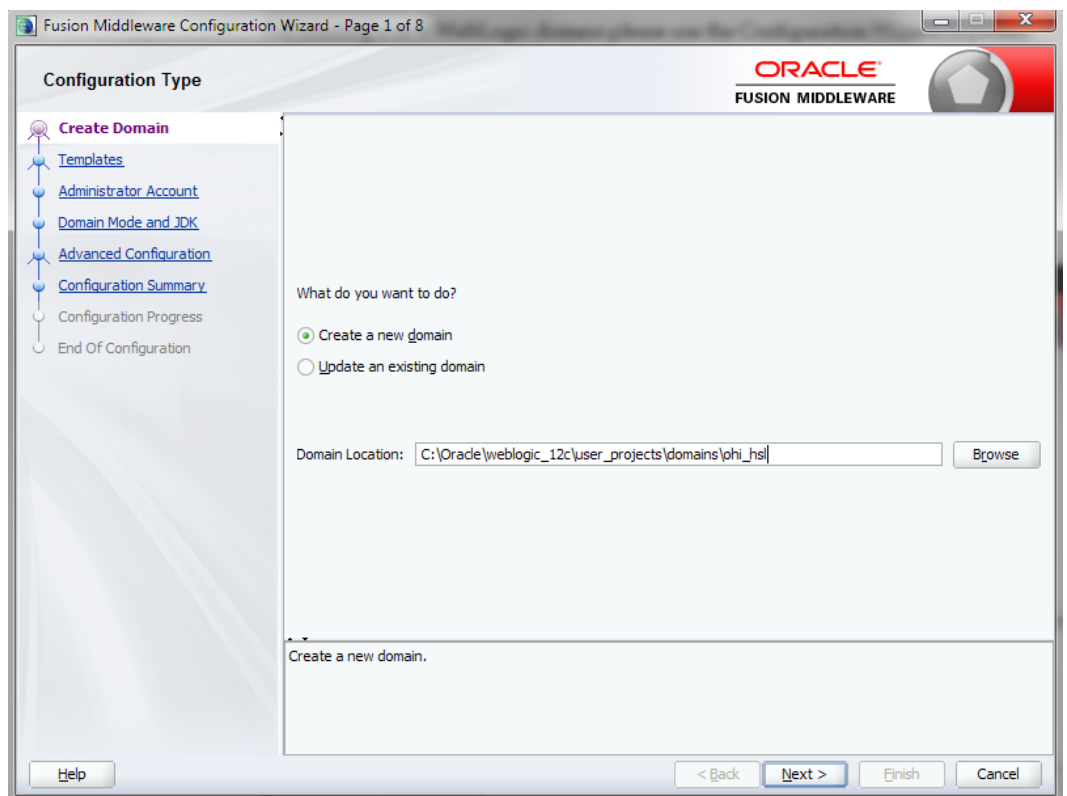
### 3.4.2 Creating a domain

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.

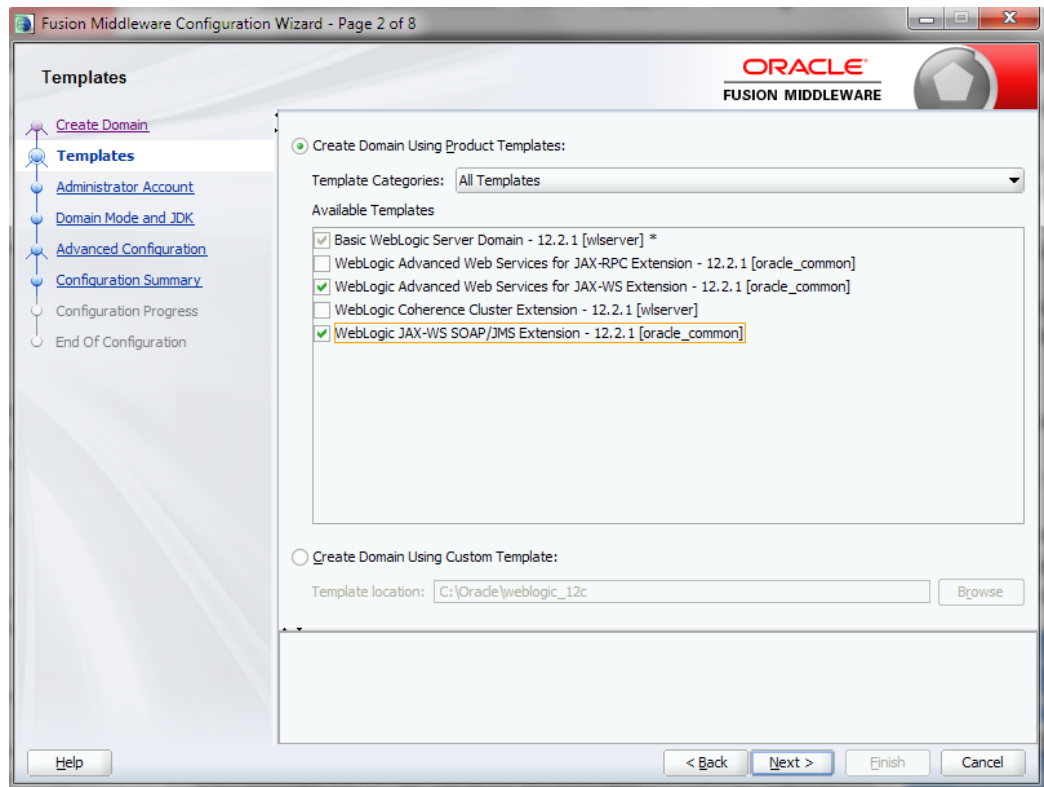
If you create the new Weblogic Domain from the same software home as the Forms Domain, you have to choose the same "Domain Mode" (Development or Production), to avoid errors during startup of the new Managed Server(s).

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.

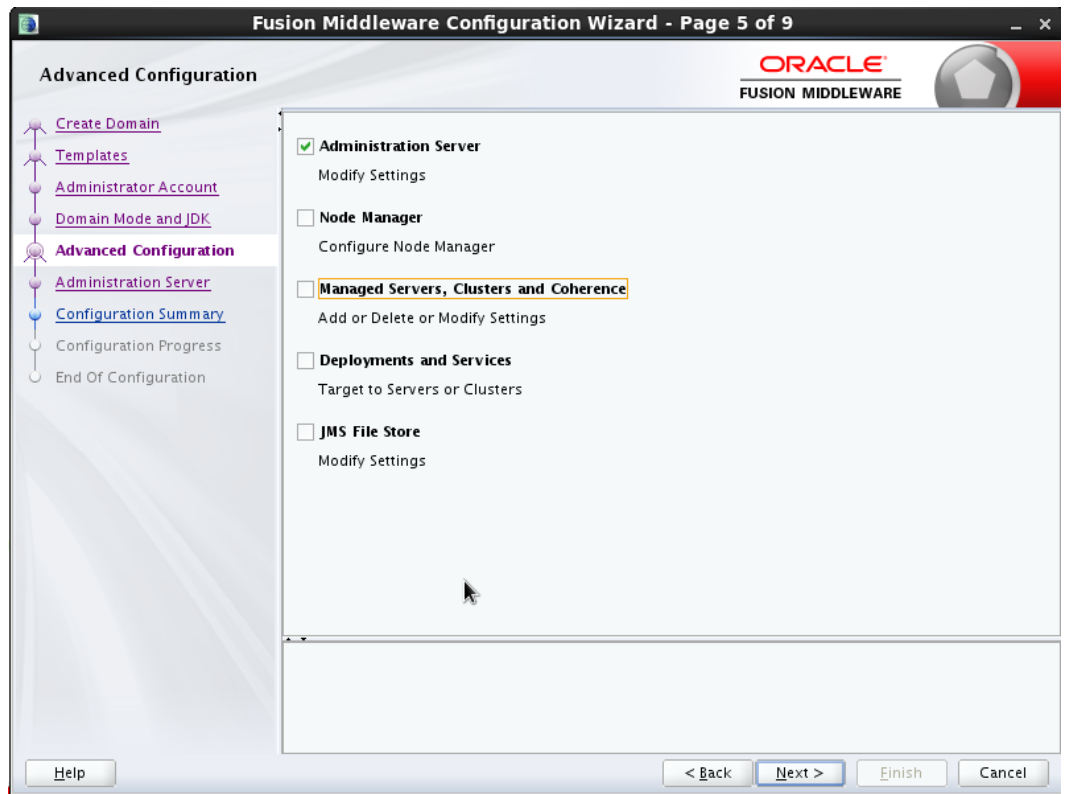


When creating a new domain select at least the options as shown below.

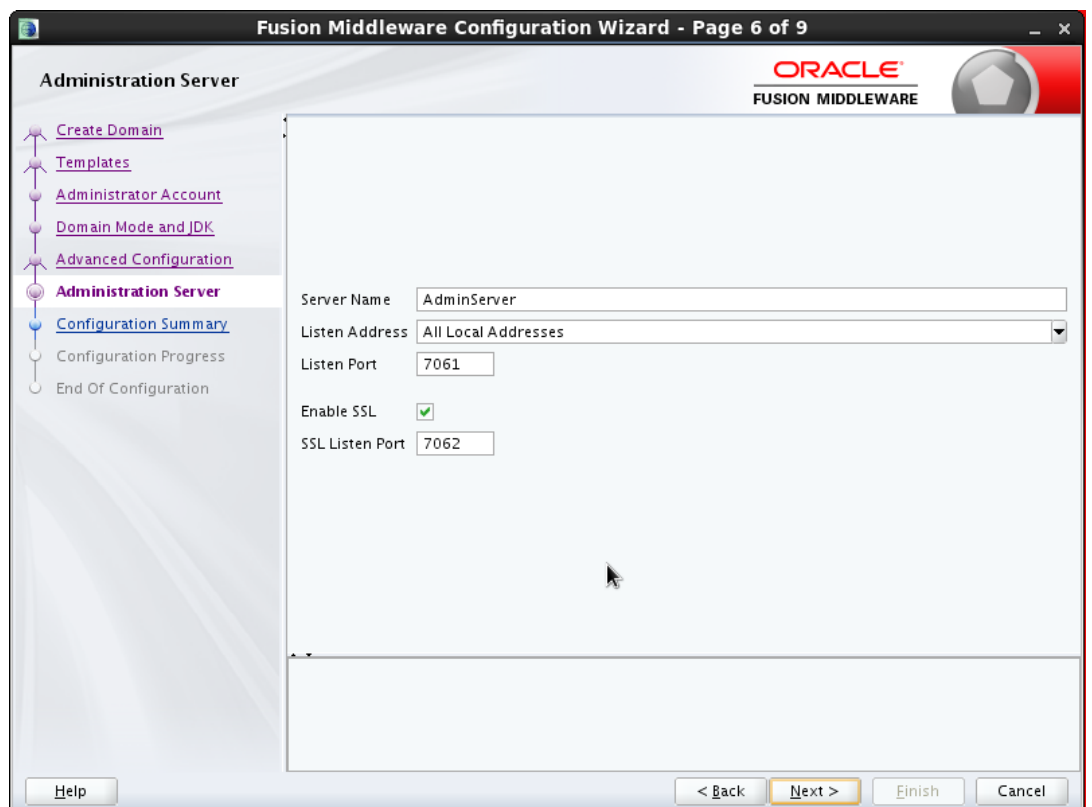


In the next screens, specify the *username* and *password* for the domain administrator account. When prompted for developer or production mode choose *production mode* and pick a JDK.

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.

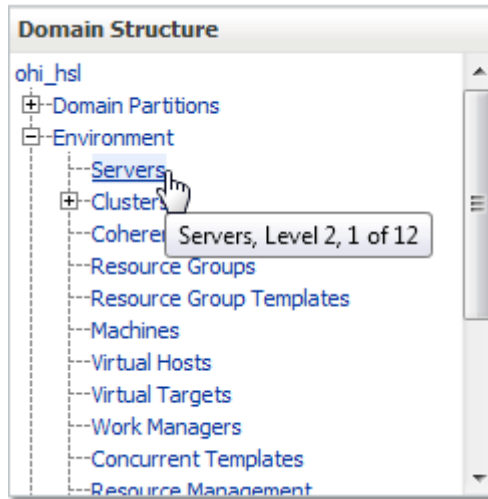


### 3.4.3 Creating Managed Server(s)

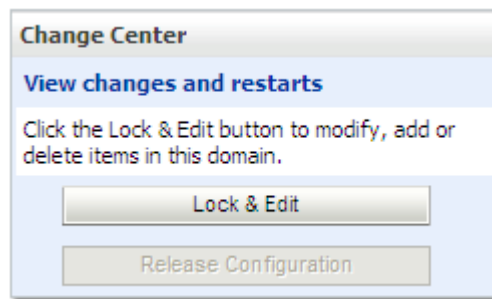
Start the Administration Server (of the existing or newly created domain) using the startWebLogic.sh script (this is present in the root folder of the domain folder, which you created through the Configuration Wizard).



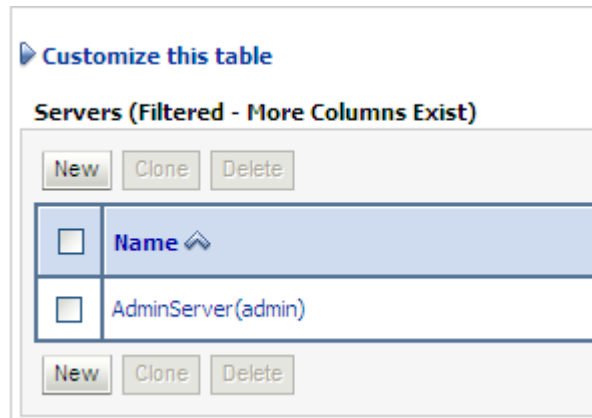
When it is started logon to the console and choose the Servers option in the left panel:



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



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



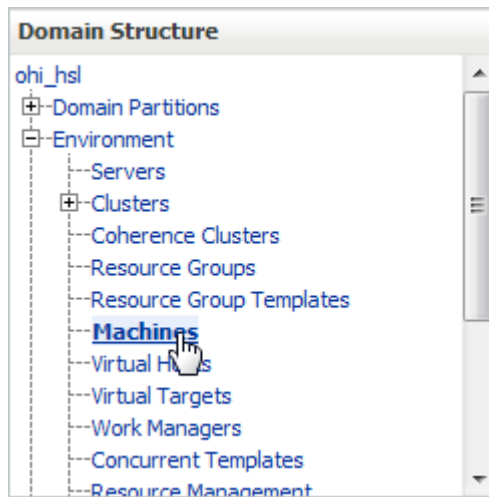
You need to provide a name and listening port for the Managed Server. For easy reference you may want to include the domain name in the name of the Managed Server, for example 'ms\_ohi\_HSL'.

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"/>	<code>ms_ohi_hsl</code>	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.

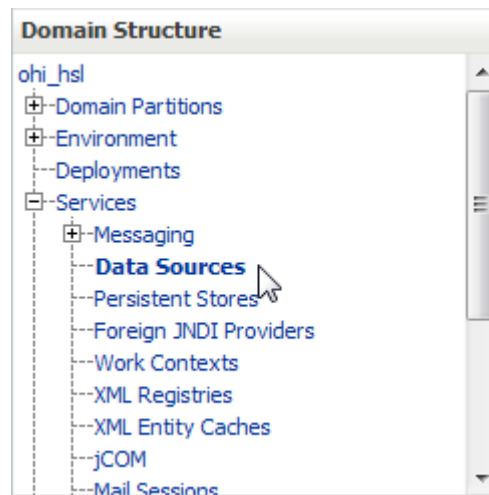
Because you are running in Production Mode, you need to create the file yourself, in the `$DOMAIN_HOME/servers/AdminServer/security` folder. This file is used when the AdminServer is started by the script `startWebLogic.sh`. If the file is not present, the script prompts for the username/password. The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

### 3.4.5 Creating a data source

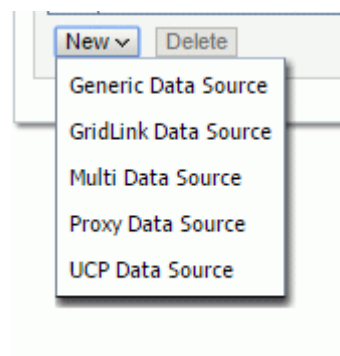
---

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:

The screenshot shows the 'Administration Console 12c' interface. The breadcrumb trail is 'Home > Summary of JDBC Data Sources'. The main heading is 'Create a New JDBC Data Source'. Below this are navigation buttons: 'Back', 'Next', 'Finish', and 'Cancel'. The section is titled 'JDBC Data Source Properties' and includes the instruction: 'The following properties will be used to identify your new JDBC data source. \* Indicates required fields'. The first question is 'What would you like to name your new JDBC data source?'. The 'Name' field is required and contains 'DS\_vohi\_dev'. The second question is 'What scope do you want to create your data source in?'. The 'Scope' dropdown is set to 'Global'. The third question is 'What JNDI name would you like to assign to your new JDBC Data Source?'. The 'JNDI Name' field is required and contains 'jdbc/DSvohi'. The fourth question is 'What database type would you like to select?'. The 'Database Type' dropdown is set to 'Oracle'. At the bottom, there are navigation buttons: 'Back', 'Next', 'Finish', and '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**

<input type="checkbox"/>	<b>AdminServer</b>
<input checked="" type="checkbox"/>	<b>ms_ohi_hsl</b>

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.

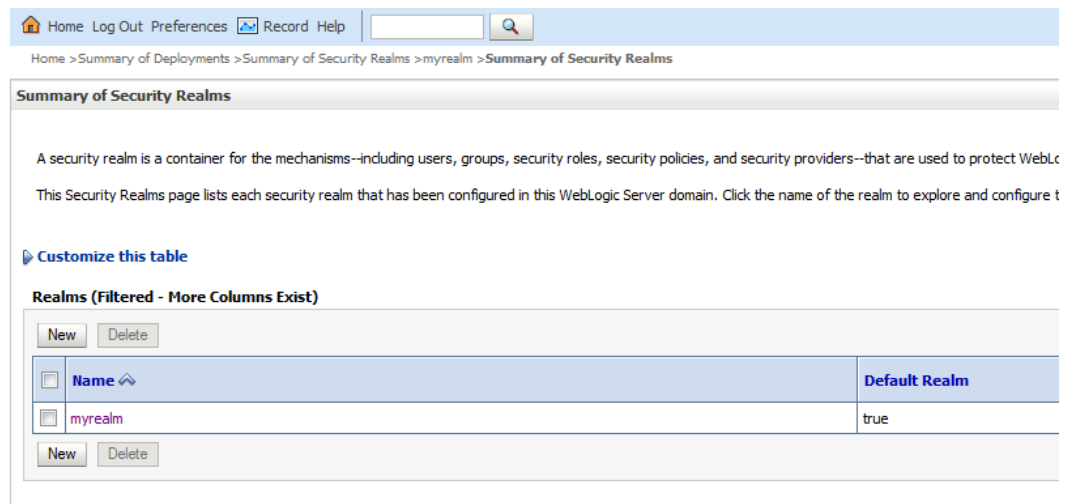
The following steps are needed to set up minimal security for the HSL application:

- Set up security realm
- Enable SSL
- Configure JSSE
- Configure key store
- Configure logging level
- Configure user lockout

#### 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 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:

The screenshot shows the 'Settings for ms\_ohi\_hsl' configuration page. The 'General' tab is selected. The 'SSL Listen Port Enabled' checkbox is checked and highlighted with a red box. The 'SSL Listen Port' is set to 7064. Other settings include: Name: ms\_ohi\_hsl, Template: (No value specified), Machine: Machine1, Cluster: (Stand-Alone), Listen Address: localhost, Listen Port: 7063, Client Cert Proxy Enabled: unchecked, Java Compiler: javac, Diagnostic Volume: Low, and Default Datasource: empty.

### 3.5.3 Configure JSSE

To use SSL with WebLogic you need to configure the use of Java Secure Socket Extension (JSSE). JSSE is not installed as part of Weblogic Server since WLS 12.1.1 (a previous implementation was in WLS 11g).

To configure it, 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`.

## 3.5.4 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 **Change**

— 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\w\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!

### 3.5.5 Configure logging level

---

The standard logging level 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 start parameter for the Managed Server:

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

Depending on the way you start your Managed Servers, this has to be added in the following locations:

- Using the Admin Console, navigate to the Msanaged Server, Tab "Server Start", field "Arguments". Add

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

- In the file `$DOMAIN_HOME/bin/startManagedWebLogic.sh`, find the first line that starts with `JAVA_OPTIONS=`. Immediately before that line, add lines like these:

```
# Custom Setting for ms_ohi_HSL to use JSSE and set debug level
for SSL:
JAVA_OPTIONS="-Dweblogic.wsee.security.debug="true"
${JAVA_OPTIONS}"
```

NOTE: The entries in Server Start field Arguments are used when the server is started through the Node Manager (they are stored in a startup.properties file stored in the data/nodemanager subfolder of the Managed Server).

When you implement the changes in the shell script they will only be used when the Managed Server is started through the startup script which means when you use the Node Manager you need to make sure the nodemanager.properties contains

```
weblogic.StartScriptEnabled=true.
```

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

### 3.5.6 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 co 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)

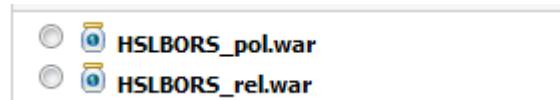
#### 3.6.1 Deploy to a single Managed Server

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

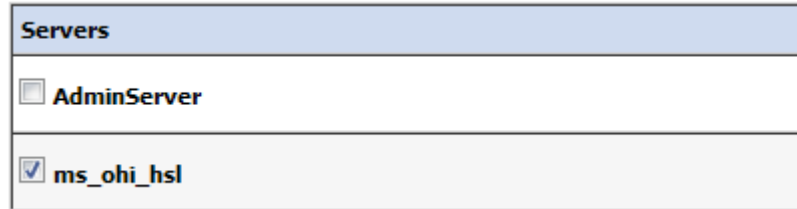
If you want to shorten this list, use 'Customize this table' to exclude the libraries.

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

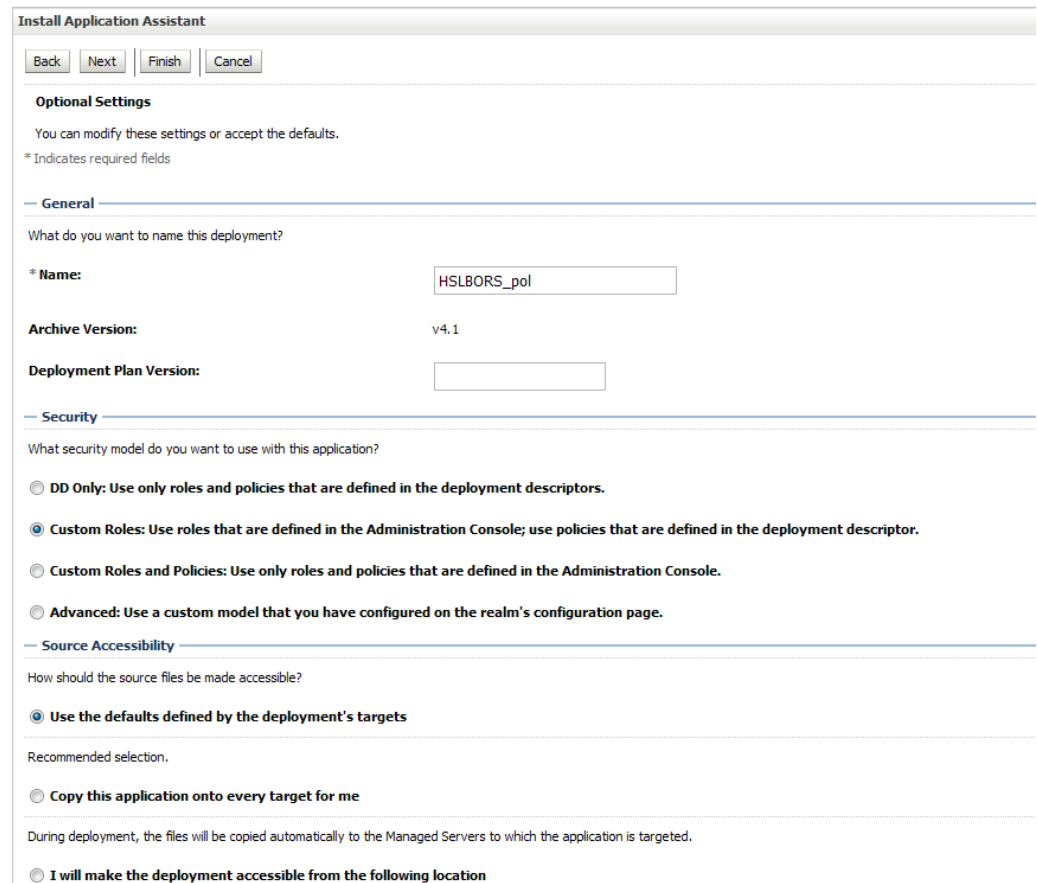
In the new window, locate the .war files on the WLS server, select it and press 'Next':



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.



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).



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).

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 the deployment or delete and install it again.

- ✓ Navigate to the Managed Server tab 'Server Start' and specify a properties file in the Arguments field like the first line shown below:

```
Arguments:  
-Dhsl.properties=/ohi/envBase/BTSPC03/ohiBase  
/conf/hsl.properties
```

Example:

```
-Dhsl.properties=/ohi/envBase/BTSPC03/ohiBase/conf/hsl.properties
```

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 also specify the default name: the \$OZG\_BASE folder and hsl.properties.

The contents of this file is discussed in a separate chapter ('Configuration files for web services').

NOTE: make sure you keep all the arguments in the field on ONE line and keep a space between the settings; and when you do not use the Node Manager implement these changes in the startup script (see earlier this chapter as well as a few paragraphs further the JAVA\_OPTIONS discussion).

When completed, (re)start the Managed Server. 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
```

ATTENTION:

If you want to start up a single Managed Server without using the Node Manager or the Admin Console (which tries to involve the Node Manager) you will typically use the script startManagedWebLogic.sh in \$DOMAIN\_HOME/bin. In order to have the correct settings for the server enabled you need to set the JAVA\_OPTIONS environment variable before starting the managed server.

An example when the current location is the bin folder of the WLS domain and your managed server is named ms\_ohi\_hsl:

```
export JAVA_OPTIONS="-  
Dhsl.properties=/ohi/envBase/BTSPC03/ohiBase/conf/hsl.properties"
```

Now you can start the Managed Server with the command below (assuming your location is the \$DOMAIN\_HOME/bin folder):

```
./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.

To make these changes permanent, put them in the file

`$DOMAIN_HOME/bin/startManagedWebLogic.sh`: find the first line that starts with `JAVA_OPTIONS=` . Immediately before that line, add these lines:

```
JAVA_OPTIONS="-  
Dhsl.properties="/ohi/envBase/BTSPC03/ohiBase/conf/hsl.properties"  
${JAVA_OPTIONS}"
```

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:

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

### 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 rather unlikely scenario, be aware that each Managed Server should have different startup parameter 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 from each Managed Server to its corresponding OHI Back Office database.

- 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 Publishing the deployed services

After you have deployed the web services, perform a small test using the test URL as provided in the Admin Console (only available for domains that run in Development Mode).

Settings for HSLBORS\_pol(v4.1)

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
[-] HSLBORS_pol	
/HSLBORS_pol/api	/application.wadl
/HSLBORS_pol/rest	/application.wadl
default	http://localhost:7063/HSLBORS_pol

You can find the definition for each service using the following URL:

The format of the service URL is

<servername>:<port>/<application>/api/swagger.json

For example: https://localhost:7064/HSL\_POL/api/swagger.json

## 3.7 Security Aspects

The HSL services provide an additional access channel to retrieve and change OHI Back Office data.

Your HSL application deployment must be sufficiently secure to prevent exposing sensitive data or enabling unauthorized changes to the OHI Back Office data. Therefore, access 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.

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

As a minimal policy to reduce the risk of unauthorized access and network sniffing, all HSL services are configured to use basic authentication. This requires HTTPS communication and a username/password combination.



The default user which is used during the deployment is 'restuser', but can be configured with a custom security policy. You need to create this user in your security realm.

It is your responsibility as an administrator to secure the HSL services within your organization.

This paragraph provides some pointers to get started:

- Restricting access with custom roles
- Swagger definition
- Testing with SoapUI

### 3.7.1 Restricting access with custom roles

---

You can restrict access at the service (or even operation) level by creating and granting global roles in the WLS console:

- Create one or more global roles in the security realm used for HSL applications. For example HTTP SVLAccessRole (to be used for all HTTP HSL services).
- Grant each service (operation) to the appropriate global role. Decide if you want to implement fine-grained access (multiple roles, grant at service operation level) or coarse-grained access (one role, grant at service level) or anything in between.
- Create users for the HSL services.
- Grant the appropriate role(s) to each user.
- Restart the Managed Server to ensure that all changes are processed.
- Verify that the new access rules are now in place (for example using SoapUI).

### 3.7.2 Swagger definition

---

The HSL services are built from a Swagger definition designed by OHI. This is a definition standard (<http://swagger.io>) supported by many leading software vendors including Oracle.

The Swagger definition documents both the operations and the objects used by a service. Apart from providing useful documentation to client application developers, it can also be used as the basis for code generation.

The Swagger definition is exposed as follows:

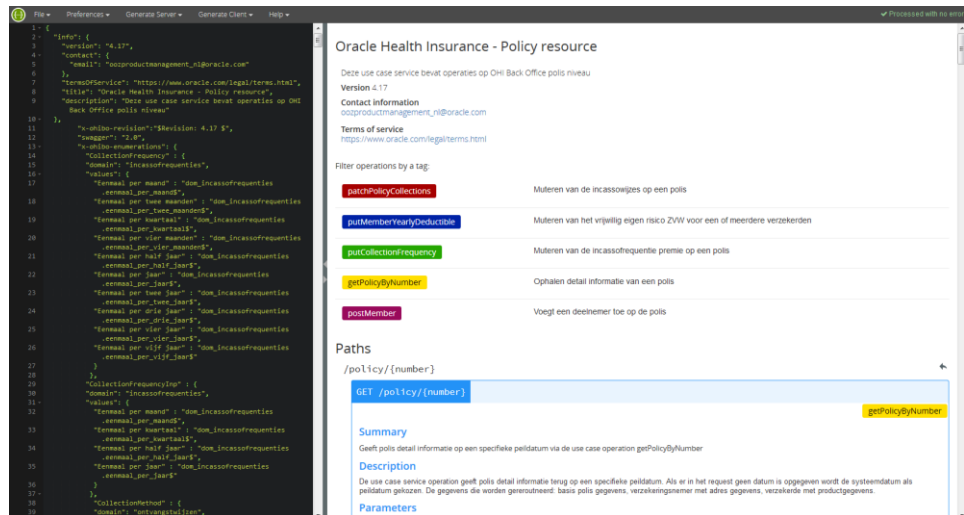
- <http://<server>:<port>/<application>/api/swagger.json>  
Returns the Swagger definition in JSON format
- <http://<server>:<port>/<application>/api/swagger.yaml>  
Returns the Swagger definition in YAML format

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

In the following example we use the online Swagger editor to view the Swagger definition of the POL service:

- Browse [http://<server>:<port>/HSL\\_POL/api/swagger](http://<server>:<port>/HSL_POL/api/swagger)

- Copy the contents of this page.
- Open the online Swagger editor by browsing <http://editor.swagger.io>
- Use 'File > Paste JSON' to paste the contents into the editor
- You can now navigate through the paths, operations and type definitions of the HSL service.



More information can be found on <http://swagger.io>

### 3.7.3 Testing with SoapUI

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

NOTE:

FMW 12.2.1 has removed the support for the security protocols SSLv3 and TLS 1.1, because they are now considered insecure. This means you have to test and use the OHI Web Services with a client that uses TLS 1.2.

We found that the latest version of SoapUI (5.2.1) does not enable TLS 1.2 by default. To fix this, add a line to soapui.bat:

```
set JAVA_OPTS=%JAVA_OPTS% -Dsoapui.https.protocols="TLSv1.2,TLSv1.1"
```

And make sure your shortcut uses that soapui.bat file.

SoapUI is not only useful for testing the functionality of the HSL services, but it is also suitable for testing their security settings.

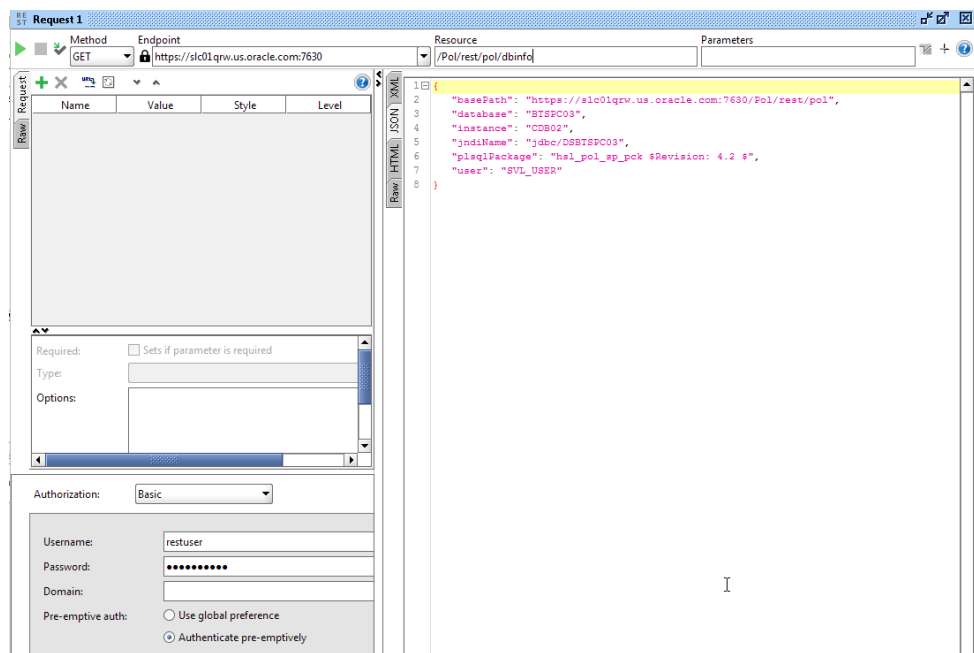
The following procedure should work if you deployed the HSL application using default security policies:

- Create a new project.
- Add the swagger.json definition to the project
- Create requests (these should be SSL requests)
- Set the properties in the "Properties" panel in the lower left corner, or in the "Auth" panel (visible after you have opened the actual request message window, change Authorization from None to Basic):

- Username= <user defined in weblogic, 'restuser' in the example>
  - Password= <password defined in weblogic>
  - Authentication Type = Preemptive
  - WSS-Password Type=PasswordText
  - WSS TimeToLive= 1000000
- Send the request. The request should succeed because restuser is a valid WLS user (if you created it).
  - If you want to get some HTTPS related information from the SSL requests in the .out or .log file you need to enable additional debugging (by default only HTTP requests will show up for example in the access log file). Please add the following to the Managed Server startup settings (in the console and/or in startManagedWebLogic.sh depending on your startup method; do not forget to remove this at a later stage):
 

```
-Dssl.debug=true
```

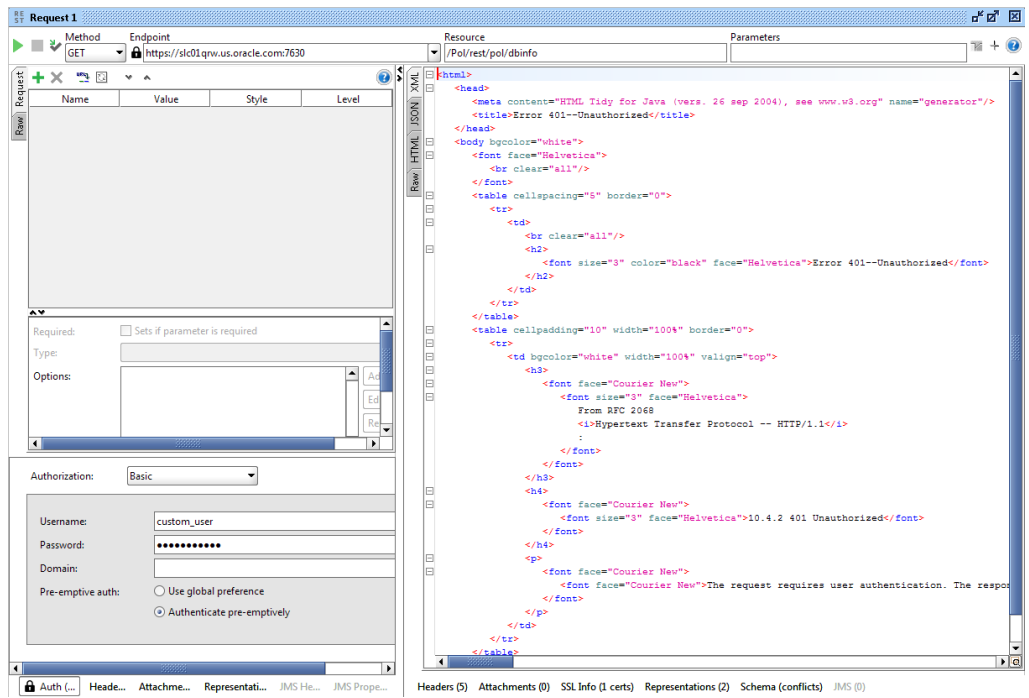
Testing the dbinfo operation with an authorized WLS user (restuser):



Note:

- SSL connection
- Preemptive authentication type
- Basic Authentication
- Received a valid response.

Finally, an example with the non-authorized WLS user 'custom\_user':



Notes:

- Error message indicates that access was denied to the service.

---

## 4 Configuration files for HSL services

In the previous chapter a properties file was referenced in the web service application server deployment description for which more information is provided below.

### 4.1 Back Office HSL properties file

The location of the Back Office properties file for the HSL services is specified as a start parameter 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.

---

#### 4.1.1 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
```

---

#### 4.1.2 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
```

### 4.1.3 hsl.usercontext

---

The OHI officer 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.

### 4.1.4 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=MANAGER
```

Note that the user context from the `hsl.properties` file may be overwritten at the HSL application level. This should be documented in the functional specification(s) which apply to the given HSL application.

### 4.1.5 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.

### 4.1.6 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

### 4.1.7 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
```

#### 4.1.8 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
```

#### 4.1.9 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
```

#### 4.1.10 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
```

#### 4.1.11 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
```

#### 4.1.12 Activating changes to hsl.properties

---

To activate changes to hsl.properties you must restart the managed server.  
If the HSL applications are deployed only to the AdminServer, you need to restart the AdminServer.

#### 4.1.13 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 induce a malfunction in the operation of HSL services.

#### 4.1.14 Example hsl.properties file

---

The following properties file is for documentation purposes only.

```
hsl.pol.jndiname=jdbc//DSVOHI
hsl.pol.usercontext=manager
# 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=manager
# 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
```

#### 4.1.15 Keeping hsl.properties up to date

---

As new HSL services are released through (patch) releases of OHI BackOffice, you will be notified to change the hsl.properties file if required.



## 5 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.

## 6 Appendix A – Service Information

The following URI gives you version information about a running HSL application:

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

For example:

[https://localhost:7002/HSL\\_POL/dbinfo](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",
  "plsSqlPackage": "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)
- **plsSqlPackage**  
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.

---

## 7 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 (where your WebLogic home folder resides in).

For more information please use the standard WebLogic documentation.