

Oracle Health Insurance Back Office

SOAP Service Layer (SVL) Installation & Configuration Manual

Version 1.30

Part number: E97070-01

September 19th, 2018

Copyright © 2011, 2018, 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
10.12.2.0.0	1.8	<ul style="list-style-type: none"> Added additional paragraph regarding securing web service access.
10.12.2.2.0	1.9	<ul style="list-style-type: none"> ohibo.properties changes for the latest releases up to 2012.02 have been added
10.12.3.0.0	1.10	<ul style="list-style-type: none"> Made clear to delete retired deployment before updating again.
10.13.1.1.0	1.11	<ul style="list-style-type: none"> Updated a number of screen prints.
10.13.1.4.0	1.12	<ul style="list-style-type: none"> Added new ohibo.properties for the latest releases up to 10.13.1.4
10.13.2.1.0	1.13	<ul style="list-style-type: none"> Web service consumers are now referenced using this correct term.
10.13.3.0.0	1.14	<ul style="list-style-type: none"> The situation as needed for the properties file for 10.13.3 has been documented. Please be aware of the new quotations and provider contract properties (the latter were introduced in a patch set on 10.13.2) and the rename of the collective agreement to group contract.
10.13.3.0.0	1.15	<ul style="list-style-type: none"> When calling alg_security_pck.svl_grants the grantee parameter name should be passed named as this routine is overloaded. The folder that covers publishing the services now contains also some simple instructions where the WSDL files can be found in the .ear file.
10.13.3.3.0	1.16	<ul style="list-style-type: none"> Added changes in the properties file for patch set release 10.13.3.3.
10.14.1.0.0	1.17	<ul style="list-style-type: none"> Added changes in the properties file for major release 10.14.1.0.0.
10.14.1.3.0	1.18	<ul style="list-style-type: none"> Added changes in the properties file for patch set release 10.14.1.3.0.
10.14.2.0.0	1.19	<ul style="list-style-type: none"> Introduction of secure deployment by default. Rewrote most of Ch. 3 ('Installation of SVL provider services'). Described security configuration. Added Appendix A ('Removing a WLS domain') Added Appendix B ('Compare version information in EAR files')
10.15.1.0.0	1.20	<ul style="list-style-type: none"> Some additional information added regarding the security configuration of the Service Layer web services. Added Appendix C ('Managing security policies using WSLT') Added Appendix D ('fixear.sh') Added Appendix E ('polman.py')
10.15.3.0.0	1.21	<ul style="list-style-type: none"> Added changes in the properties file for patch set release 10.15.1.1. Added changes in the properties file for patch set release 10.15.1.3. Added changes in the properties file for major release 10.15.3.0
10.16.1.0.0	1.22	<ul style="list-style-type: none"> Added missing changes in the properties file for major release 10.16.1.0 Added a small paragraph about sizing/load impact
10.16.2.0.0	1.23	<ul style="list-style-type: none"> Adapted for FMW 12.2.1.1.0 Removed old properties file definitions, which break the links above.
10.16.2.3.0	1.24	<ul style="list-style-type: none"> Some minor adjustments in SVL domain configuration. Replaced Log4J configuration by Java Util logging configuration
10.17.1.0.0	1.25	<ul style="list-style-type: none"> Updated reference from FRS12211 to FRS12212 Changed grant instructions
10.17.2.0.0	1.26	<ul style="list-style-type: none"> No changes
10.17.2.3.0	1.27	<ul style="list-style-type: none"> Added changes in the properties file relevant starting with patch set release 10.17.2.2. Added JDK version specific information regarding JSSE configuration.
10.18.1.0.0	1.28	<ul style="list-style-type: none"> Revised introduction and document title Revised Architectural Overview
10.18.1.2.0	1.29	<ul style="list-style-type: none"> Use setUserOverrides.sh instead of modifying startManagedWebLogic.sh and Server Start arguments. Support for FMW 12.2.1.3. Numerous small changes and updates.
10.18.1.3.0	1.30	<ul style="list-style-type: none"> Corrected error in setUserOverrides.sh: removed double quote after disableCaptureStackTrace. Added warning about patch 28278427

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]** Oracle Health Insurance Back Office HTTP Service Layer - Installation and Configuration Guide (CTA 13681)

Contents

1	Introduction.....	7
1.1	Provider web services and web service consumers.....	7
1.2	PL/SQL and SOAP interface.....	7
1.2.1	SVL web service consumers.....	8
1.3	Licenses.....	8
2	Architectural overview.....	9
2.1	Provider web services.....	9
2.2	Web service consumers.....	10
3	Installation of SVL provider web services.....	12
3.1	Sizing/load aspects.....	12
3.1.1	Deployment choices.....	13
3.2	Database installation.....	13
3.3	WLS Preparation.....	14
3.3.1	Requirements.....	15
3.3.2	Creating a domain.....	16
3.3.3	Creating Managed Server(s).....	18
3.3.4	Creating a machine definition.....	19
3.3.5	Creating a data source.....	21
3.4	Security Configuration.....	27
3.4.1	Set up security realm.....	27
3.4.2	Enable SSL.....	28
3.4.3	Configure JSSE.....	29
..3.4.3.1	JDK 1.8.0_162 and above.....	30
..3.4.3.2	JDK 1.8.0_151 .. 1.8.0_161.....	30
..3.4.3.3	Below JDK 1.8.0_151.....	30
3.4.4	Setting up a key store.....	30
3.4.5	Configure logging level.....	31
3.4.6	Set user lockout.....	33
3.5	(Re)deployment of the SVL Application.....	33
3.5.1	Deploy to a single Managed Server.....	33
3.5.2	Deploy to multiple Managed Servers.....	36
3.5.3	Deploy to cluster.....	36
3.5.4	Deploy for multiple environments (DTAP).....	37
3.5.5	Publishing the deployed services.....	37
..3.5.5.1	Retrieving the WSDL from the EAR file.....	38
3.6	Security Aspects.....	38
3.6.1	Using the default security policy.....	39
3.6.2	Overruling the default policy.....	39
3.6.3	Restricting access with custom roles.....	40
3.6.4	Testing with SoapUI.....	41
4	Configuration files for provider web services.....	44
4.1	Back Office web services properties file.....	44
4.1.1	10.18.1.2.0 properties.....	45
4.1.2	10.17.2.2 properties.....	45
5	OHI release upgrade and provider web services.....	48

6	Installing and deploying web service consumers.....	50
6.1	Preparation of your database environment	50
6.2	Prepare a secure setup	50
6.3	Deployment of web service consumers	50
7	Appendix A - Removing a WLS domain	52
8	Appendix B - Compare version information in EAR files	53
8.1	Invocation	53
8.2	Operation.....	53
8.3	Output.....	54
9	Appendix C - Managing security policies using WSLT.....	55
9.1	Relevant WLST commands	55
9.2	Requirements	55
9.2.1	WLS version	55
9.2.2	OWSM needed to attach policies using WLST	55
9.3	Restrictions in WLST	56
9.4	Tips and Tricks.....	56
9.4.1	Capturing listWebServices output.....	56
9.4.2	Listing available OWSM policies.....	56
9.4.3	Activate application before running WLST commands	57
9.4.4	Remove temp directories	57
9.5	Example WLST script (polman.py).....	58
9.5.1	CSV format	58
9.5.2	Script source	59
10	Appendix D - polman.py	60

1 Introduction

OHI Back Office web services are used to integrate with existing applications or provide a back end to bespoke self-service portals for insurance members.

OHI Back Office provides two types of web services:

- **Business services (aka. SVL services)** – this document generic object-oriented SOAP/HTTP operations on core OHI Back Office data, with ‘find’ and ‘get’ operations to retrieve data and ‘write’ operations to update/add data.
- **Use Case services (aka HSL services)** – see **Doc[2]** REST operations to support typical use cases for Dutch healthcare payers. Examples: requesting a new policy, adding an insured member, changing insured products, changing payment method etc.

The default security for all web services is Basic Authentication over SSL.

SVL services are deployed to WebLogic Server to be accessed over SOAP/HTTP. Service operations can also be called using PL/SQL since the service functionality is implemented in PL/SQL.

This document describes the generic technical details regarding the SVL services, how to install and update them and how to change configuration settings.

1.1 Provider web services and web service consumers

There are two types of SVL components:

- **provider web services**
SOAP/HTTP services built by OHI Back Office to be called (‘consumed’) by client applications in the surrounding environment.
- **web service consumers**
Java classes built by OHI Back Office to ‘consume’ third party SOAP/HTTP services from within the OHI Back Office database.

The implementation of the web service consumers is very different from that of the (provider) web services. Common for both is the use of SOAP/HTTP technology which implies that WSDL is used to describe the interface and that XML is used to serialize objects.

It depends on the requirements of your organisation whether you need to set up both types of web service ‘components’.

1.2 PL/SQL and SOAP interface

SVL services are primarily invoked through SOAP/HTTP as document-style web services in a Service Oriented Architecture. However, they can also be called from PL/SQL. Using the PL/SQL interface can be attractive if your client code is a PL/SQL script or package.

The Java classes which provide the SOAP/HTTP interface are a light weight wrapper around the functional implementation in PL/SQL.

The web service consumers were designed to call a third-party web service from within the database.

Each web service consumer has:

- a JAR file with Java client code generated from the WSDL.
This JAR file must be loaded into the OHI Back Office database for running in the Oracle JVM (Oracle's JVM in the database)
- A PL/SQL wrapper package providing a PL/SQL interface to the public methods of the Java classes in the consumer JAR file.
The PL/SQL wrapper functions are called from within the OHI Back Office database.

1.3 Licenses

No license is required to use the SVL web service consumers.

No license is required to use the 'Vecozo-specific' provider web services. The PreAuthorization service is an example of this.

Customers are required to have the appropriate license for using all other SVL provider web services.

Separate licenses can be obtained for the Claims-related functions and for the Policy-related functions within SVL.

Customers with a Connect to Back Office license are currently permitted to install and use the provider web services component of the Service Layer. This is valid until further notice.

A separate license is required for the use of the get function to obtain Composite Relation Details (part of the Relation Service). This function can only be used when the so-called Connector option for the Oracle Service Cloud has been purchased.

The underlying PL/SQL service of a provider web service may not be used when no Connect to Back Office or Service Layer license is obtained for the provider web service.

For further information, please consult your OHI sales representative.

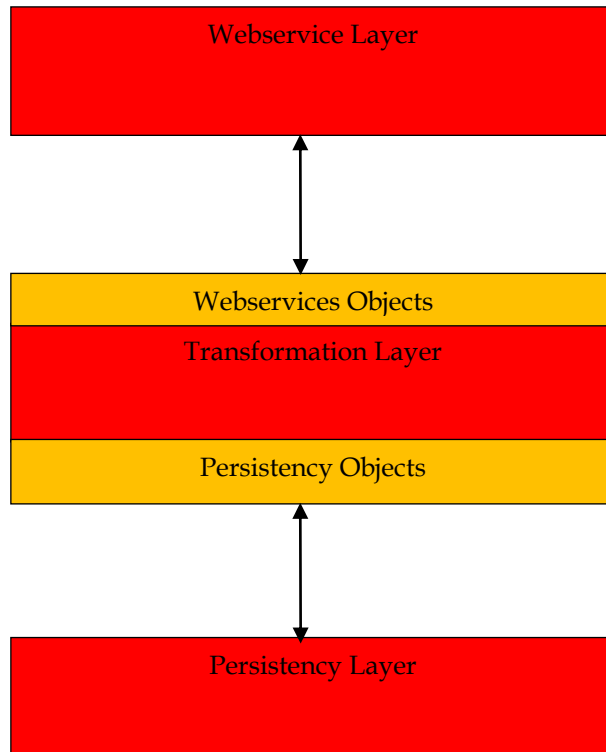
2 Architectural overview

This chapter gives a high level architectural overview of the current SVL provider web services and web service consumers.

2.1 Provider web services

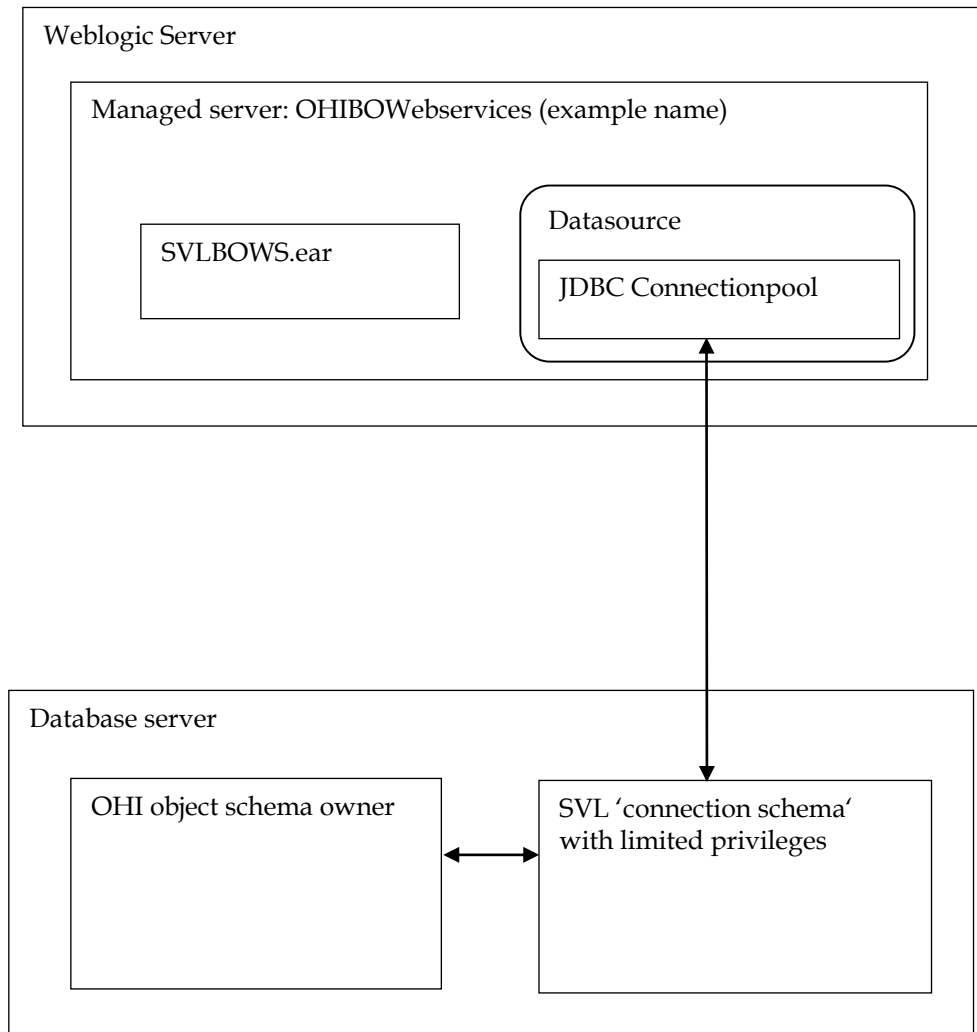
The Java classes of each SVL provider web service are stored as a WAR file. The SVL services are then bundled into a single EAR file and deployed to WebLogic Server.

Each SVL service has the following architecture:



The persistence layer is used to map Java objects to SQL types and vice versa when calling the PL/SQL services in the OHI Back Office database.

The schema below shows how the deployed SVL service connects to the OHI Back Office database.
 The SVL 'connection schema' is a separate database account with limited access rights which is used to call the PL/SQL implementation of the SVL service.



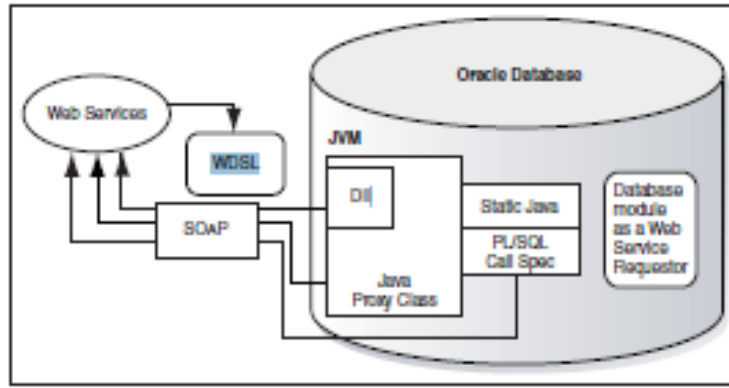
2.2 Web service consumers

Each web service consumer consists of a JAR archive with Java JAX-RPC client classes and a PL/SQL wrapper package. Both components are derived from the same WSDL definition. Note that the WSDL will closely resemble the WSDL provided by the third-party web service.

The JAR archive must be uploaded into the OHI Back Office database so that its code can be executed by Oracle's JVM in the database (OJVM).

The PL/SQL wrapper package provides the PL/SQL interface to execute the Java client code to call the third party web service.

The diagram below shows how the PL/SQL code works with the Java client code in the database to make a web service callout.



3 Installation of SVL provider web services

This chapter describes the steps to (re)install the SVL provider web services (as opposed to the SVL web service consumers).

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

1. Sizing/load aspects
2. Database installation
3. WLS preparation
4. Security configuration
5. (Re)deployment of the SVL application (SVLBOWS.ear)
6. Security aspects
7. Miscellaneous

3.1 Sizing/load aspects

From the “Introduction” and the “Architectural overview” chapters it should be clear that the actual functionality of the services is offered by the PL/SQL implementation in the database.

The SOAP interface as implemented in Java within the application server is a very light weight pass through layer for the request and response messages. It only validates and transforms the actual XML request call to a PL/SQL call and transforms the result from the PL/SQL routine back to an XML response message.

As a result of this choice the load on the application server is very limited. The processing on the application server is typically less than 10% of all processing involved. As a rule of thumb you may assume that when you have a heavy load situation, where 10 CPU threads are involved on the database server handling all the incoming web service requests, you should not need more than 1 CPU thread busy on the application server handling these web service calls.

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 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.

An exception to this rule is processing a large provider contract write request, this may take minutes to process (on the database server by the PL/SQL implementation).

These response times are based on production experiences with the OHI services as observed until early in 2018.

3.1.1 Deployment choices

Of course the overall load of the OHI application and the portion of the load that is related to the web service calls is customer specific and may change over time. When all insured members use a healthcare payer website that directly calls the OHI web services quite some load may be expected during the commercial season. Offloading choices to standby databases and potentially caching may reduce this load.

It is expected that the actual load of the application functionality that calls the OHI web services and the related database load still widely exceed the web service application server load given only the low level pass through functionality implemented on the application server.

Knowing this, the application server that is used for the OHI user interface processes (implemented through Oracle Forms and WebLogic Server) may be an obvious and valid choice for the deployment of the application server part of the web services. When the application server load of the service calls grows a lot over time, additional processing power may be required. Monitoring the load of the Forms processes and the Service Layer processes will show whether this might be needed at some moment.

An advantage of deploying the web services on the same application server is that existing WebLogic Server licenses for Oracle Forms can be used for the web services. The OHI web services are typically certified for the same WebLogic Server version that is certified for using the Oracle Forms user interface.

Of course requirements like high availability and fail over may influence the deployment choices as well as the use of a service bus. This may lead to re-using existing infrastructure and licenses for other Oracle products using the same WebLogic Server technology stack, provided the same certified versions of these technology products are used.

3.2 Database installation

The database installation for the Service Layer consists of the creation of a separate account (or even several) with Service Layer access privileges. All functional Service Layer database objects are owned by the OHI Back Office schema owner and should have been installed as part of the database part of the OHI Back Office release installation.

The separate database account(s) must be created separately as part of the Service Layer installation, provided you have a license for the Service Layer.

Before creating the account(s), check if you will be able to use the Service Layer:

You should find a database object (package) SVL_UTILS_PCK in the OHI Back Office schema owner. If not, something went wrong with the installation of the Service Layer code as part of the OHI Back Office release installation. If this is the case, please contact the OHI Support department.

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

The use of a separate database account / schema owner for accessing the Service Layer components is required for improved security. This account needs to receive the necessary object privileges.

One or more of these accounts can be created. It is an option to use this account also as schema owner for custom code development. If you choose to do that, please follow the directions as described in [Doc\[1\]](#). We advise to use separate accounts for these purposes, though.

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

1. Create a schema owner, for example SVL_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 an enforced password policy.
2. Grant create session system privilege to this account.
3. Grant the Service Layer object privileges: logon as the OHI Back Office schema owner, enable server output, and run
“alg_security_pck.svl_grants(pi_owner => '<your OHI schema owner>', pi_grantee => '<your account>')”, for example:

```
execute
alg_security_pck.svl_grants
(pi_owner => 'OZG_OWNER'
,pi_grantee => 'SVL_USER')
```

IMPORTANT: This command does not have to be repeated after each new deployment of a new .ear file. During the database installation of OHI patches any existing grantees of the SVL objects receive any required additional grants. However, if you run into ORA-01403 errors during a web service execution your first check should be to run this command in SQL*Plus, enabling server output before running, and see whether missing grant privileges were granted.

3.3 WLS Preparation

When the database account has been created and granted successfully, a WebLogic Server environment (software home) must be prepared for deploying the SVL 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 that case you are required to create a new WebLogic domain (with a new Admin Server) to run the SVL services, in order to prevent interference with the GUI application.
- Deploy the web services in a separate WebLogic environment (possibly on a separate server). This has the advantage that you can separately upgrade or patch the different WebLogic environments, or implement a workload distribution.

Deploy the web service application in multiple environments for better scalability. Be sure to deploy the SVL services 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 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:

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

3.3.1 Requirements

The following requirements/limitations must be taken into account:

- ✓ A certified WebLogic Server version including JAX-WS (SOAP/JMS) extensions. The web services must be deployed on a single Managed Server or a cluster of Managed Servers (the ‘target’).

For the certification of specific OHI Back Office versions against WebLogic 12.2.1.2 and 12.2.1.3, please check the Certification information in My Oracle Support.

- ✓ The web 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 SVL 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.

SVL can be deployed on the same Managed Servers as C2B or HSL.

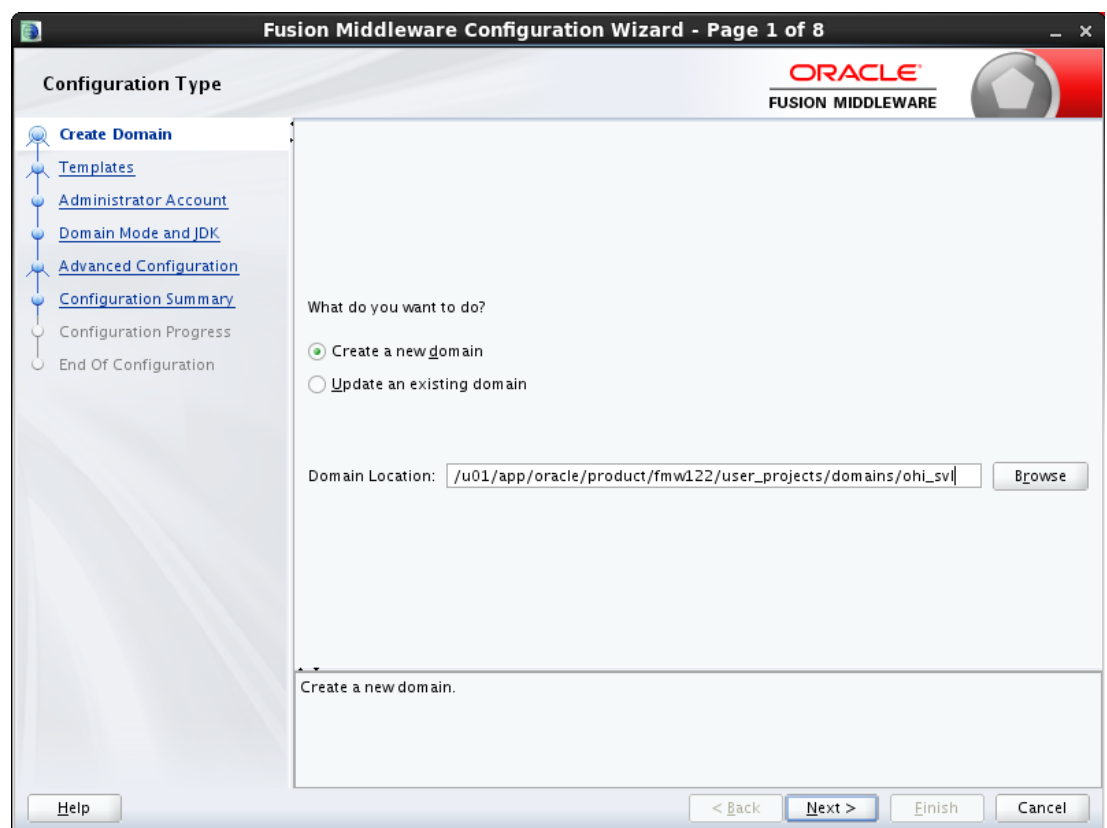
3.3.2 Creating a domain

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.

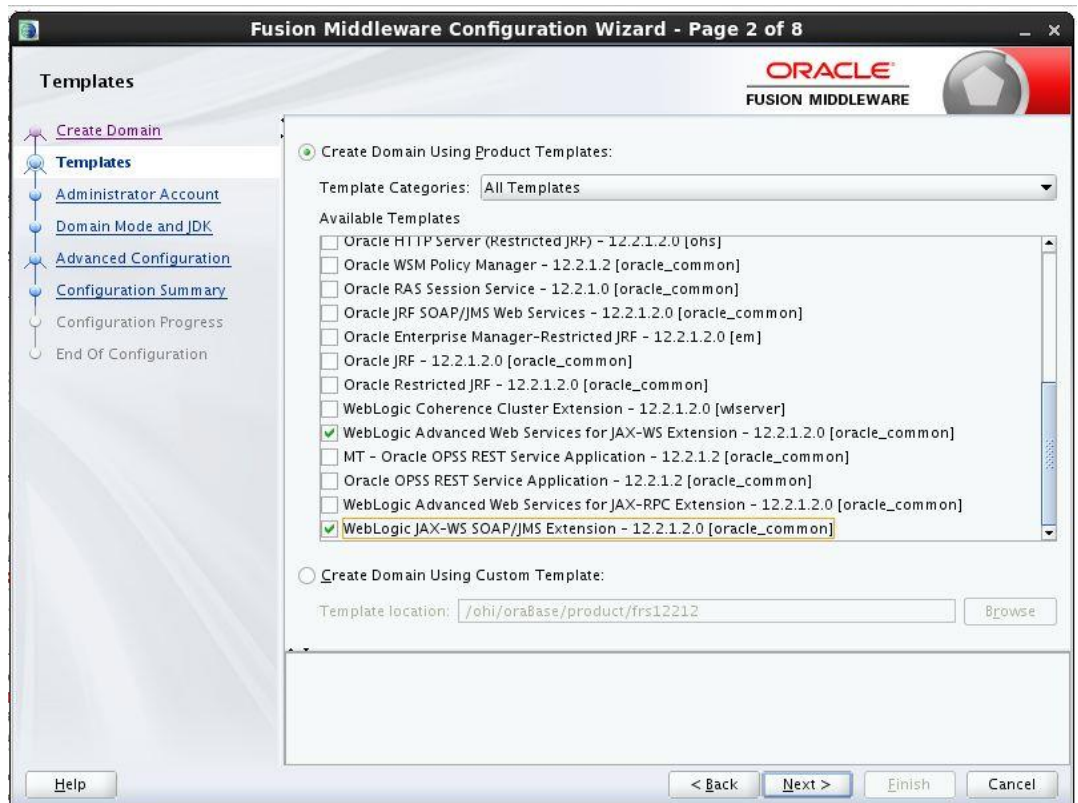
If you create the new WebLogic Domain from the same software home as the Forms Domain, you have to choose the same “Domain Mode” (Development or Production), to avoid errors during start-up of the new Managed Server(s).

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.

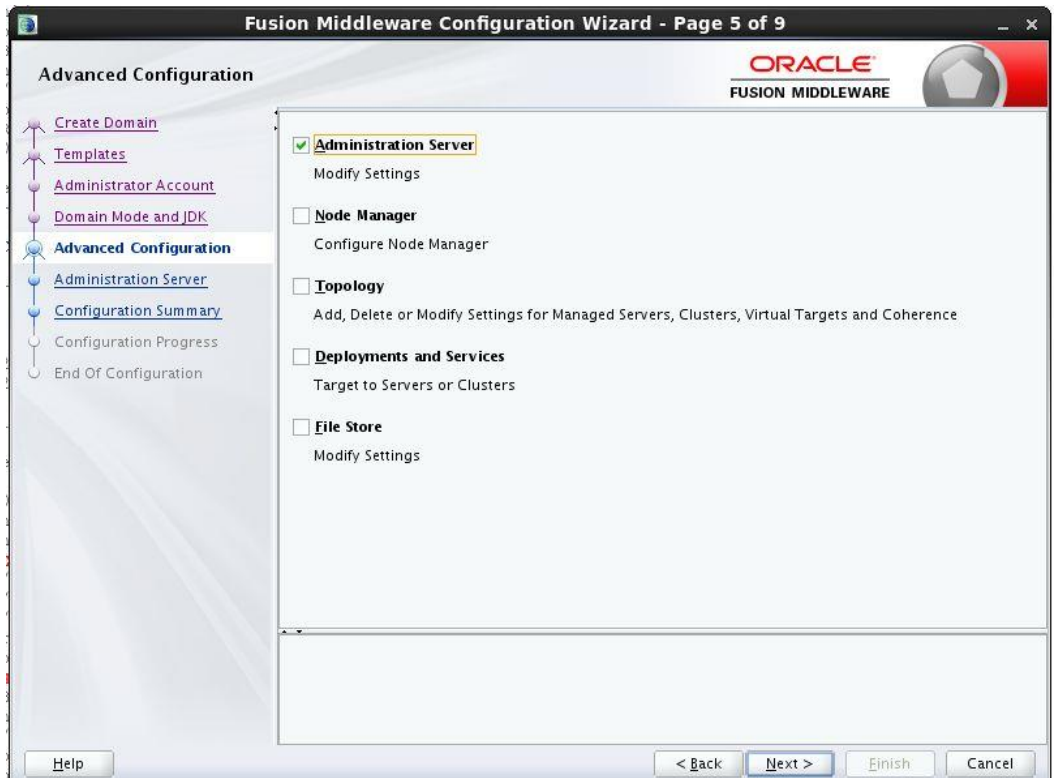


When creating a new domain select at least the options as shown below.

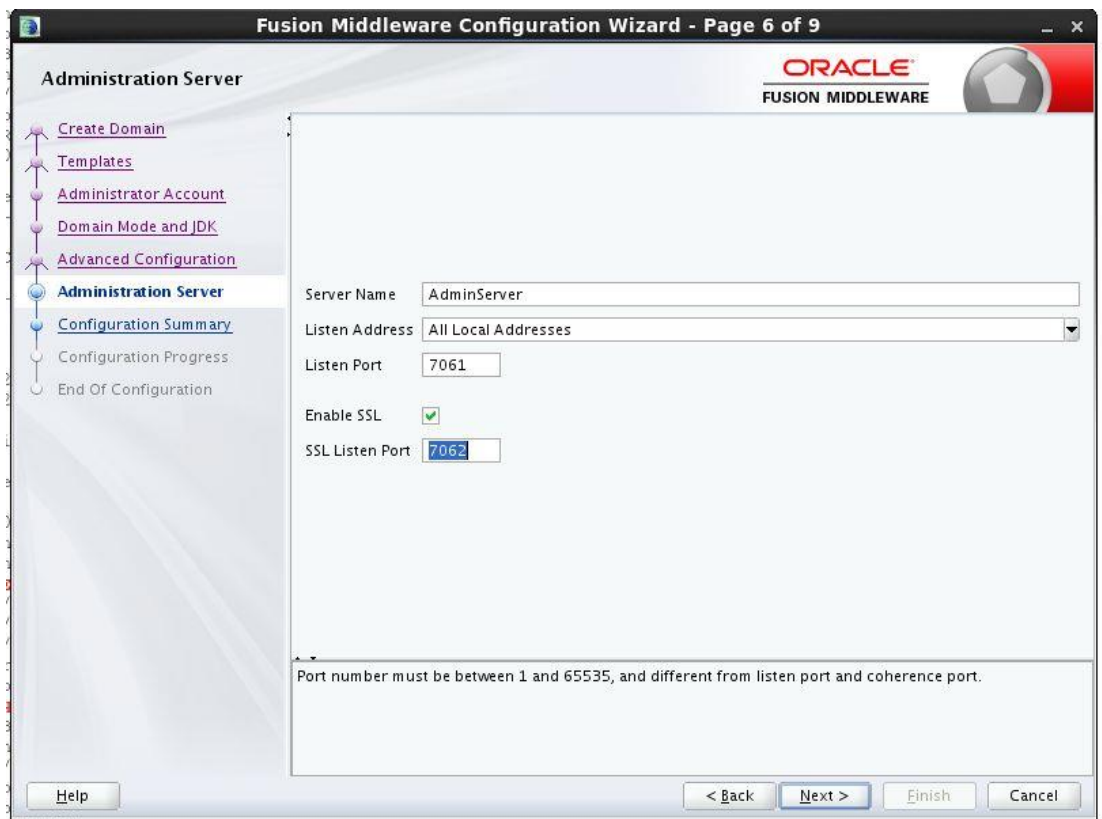


In the next screens, specify the *username* and *password* for the domain administrator account. When prompted for developer or production mode choose *production mode* and pick a JDK.

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.

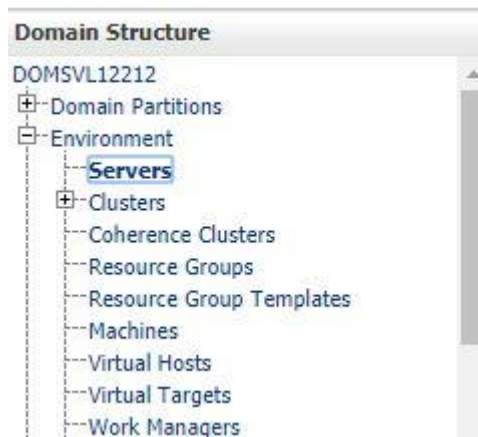


Finish the domain creation by the Configuration Wizard.

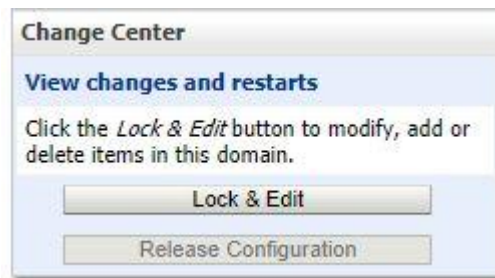
3.3.3 Creating Managed Server(s)

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).

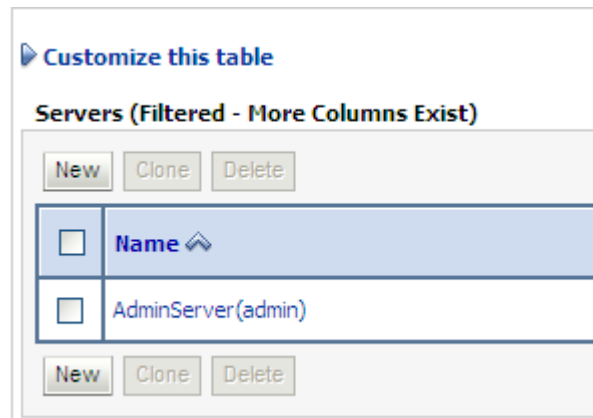
When it is started log on to the console and choose the Servers option in the left panel:



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



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



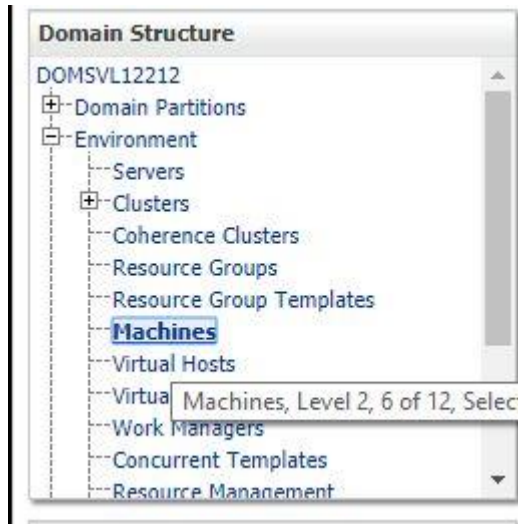
You need to provide a name and listening port for the Managed Server. For easy reference you may want to include the domain name in the name of the Managed Server, for example 'ms_svl_ohi'.

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.

3.3.4 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_svl_ohi` is assigned to `Machine1`.

► **Customize this table**

Servers (Filtered - More Columns Exist)

<input type="button" value="Add"/> <input type="button" value="Remove"/>			
<input type="checkbox"/>	Name	Cluster	Machine
<input type="checkbox"/>	<code>ms_svl_ohi</code>		<code>Machine1</code>
<input type="button" value="Add"/> <input type="button" value="Remove"/>			

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).

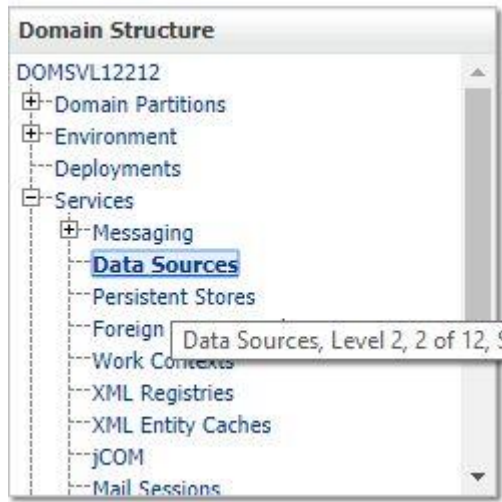
When you are running in Development Mode, a `boot.properties` file is automatically created for the AdminServer.

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.

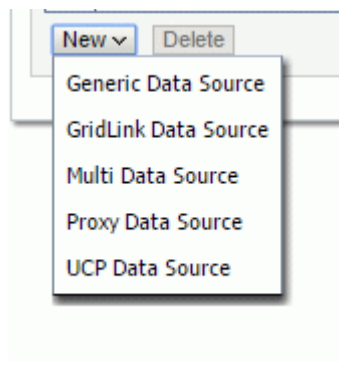
3.3.5 Creating a data source

The SVL 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 SVL 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 a "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 > ol6ohi.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.

Home > Summary of JDBC Data Sources

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"/>	AdminServer
<input checked="" type="checkbox"/>	ms_svl_ohi

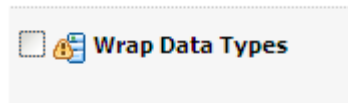
Press Activate Changes to conclude your configuration.

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:



3.4 Security Configuration

All SVL provider web services are configured to use a default security policy (policy:Wssp1.2-2007-Https-BasicAuth.xml). The default policy enforces basic authentication and SSL encryption.

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

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

3.4.1 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.

Home Log Out Preferences Record Help

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

Summary of Security Realms

A security realm is a container for the mechanisms—including users, groups, security roles, security policies, and security providers—that are used to protect WebLogic. This Security Realms page lists each security realm that has been configured in this WebLogic Server domain. Click the name of the realm to explore and configure it.

[Customize this table](#)

Realms (Filtered - More Columns Exist)

New Delete

Name	Default Realm
myrealm	true

New Delete

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

3.4.2 Enable SSL

The SVL 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 SVL services application.

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

Settings for ms_ohi_svl

Configuration Protocols Logging Debug Monitoring Control Deployments Services Security Notes

General Cluster Services Keystores SSL Federation Services Deployment Migration Tuning Overload

Save

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

[View JNDI Tree](#)

Name: ms_ohi_svl

Template: (No value specified) [Change](#)

Machine: ol6ohi.ohi.oracle.com

Cluster: (Stand-Alone)

Listen Address:

Listen Port Enabled

Listen Port:

SSL Listen Port Enabled

SSL Listen Port:

Client Cert Proxy Enabled

Java Compiler:

Diagnostic Volume:

Default Datasource:

[Advanced](#)

Save

3.4.3 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 (and what) 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>

..3.4.3.1 *JDK 1.8.0_162 and above*

No action is needed.

..3.4.3.2 *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.

..3.4.3.3 *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:

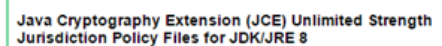
SVL domain creation – JSSE continued

- 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`.

## 3.4.4 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\_svl

Configuration Protocols Logging Debug Monitoring Control Deployments Services Security Notes

General Cluster Services **Keystores** SSL Federation Services Deployment Migration Tuning Overload Concurr

Save

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

Keystores: Demo Identity and Demo Trust

— Identity —

Demo Identity Keystore: kss://system/demoidentity

Demo Identity Keystore Type: kss

Demo Identity Keystore Passphrase:

— Trust —

Demo Trust Keystore: kss://system/trust

Demo Trust Keystore Type: kss

Demo Trust Keystore Passphrase:

Java Standard Trust Keystore: /usr/java/jdk1.8.0\_92/jre/lib/security/cacerts

Java Standard Trust Keystore Type: jks

Java Standard Trust Keystore Passphrase:

Confirm Java Standard Trust Keystore Passphrase:

Save

**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!

### 3.4.5 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 SVL 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
```

Until WebLogic 12.1.2, you had to specify start up options for WebLogic servers (admin and managed servers) in multiple locations:

- Via the console: for each server, in the tab “Server Start” in the field “Arguments”
- In file \$DOMAIN\_HOME/bin/startManagedWebLogic.sh for managed servers
- In file \$DOMAIN\_HOME/bin/startWebLogic.sh for admin servers

WebLogic 12.1.2 introduced a better way to pass start up parameters 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.

This can replace the previous methods and will be described here.

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_svl_ohi"]
then
 # add settings for SVL
 # Custom Setting for ms_svl_ohi to set debug level for SSL
 JAVA_OPTIONS="-Dweblogic.wsee.security.debug="true" ${JAVA_OPTIONS}"
fi
export JAVA_OPTIONS
```

Replace the server name ms\_svl\_ohi 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. After you have deployed the services and are testing them (through for example SoapUI as described later), you might get a message like this in the Response message:

```
weblogic.xml.crypto.wss.WSSecurityException: Timestamp validation failed.
```

This would indicate that you forgot to add a timestamp when calling the SVL application.



While setting up the SVL web services 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

## 3.5 (Re)deployment of the SVL Application

For deploying the SVL application you need to obtain an EAR file. This file is typically named SVLBOWS.ear. It should reside in the \$OZG\_BASE/java directory on the application server containing the OHI Back Office software release and you can copy it to another location if required.

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

Note that you cannot use an older EAR 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)

### 3.5.1 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

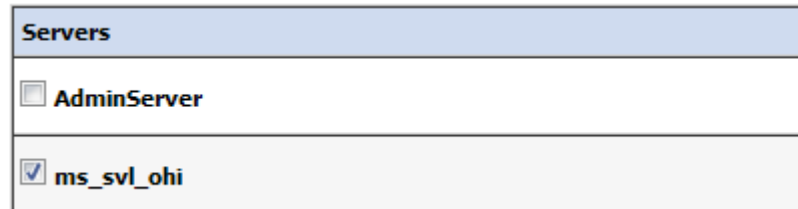
If you want to shorten this list, use 'Customize this table' to exclude the libraries.

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 .ear file 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\_svl\_ohi is chosen.



Press 'Next' and decide about a deployment name and security model. At this moment the version of the .ear 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: v4.297

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

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 policy of each web service.

Regarding source accessibility, select 'Copy this application....' if you want to remove the EAR 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.

Add two lines to the file \$DOMAIN\_HOME/bin/setUserOverrides.sh you created earlier. Add the lines to the part for the SVL server, as indicated below:

```
specify additional java command line options for specific servers
if ["${SERVER_NAME}" = "ms_svl_ohi"]
then
 # add settings for SVL
 # Set debug level for SSL
 JAVA_OPTIONS="-Dweblogic.wsee.security.debug="true" ${JAVA_OPTIONS}"
```

```
Set location for SVL properties file
JAVA_OPTIONS="-Dapp.properties="/u01/app/oracle/product/OHI/vohi/svl_ws.properties"
${JAVA_OPTIONS}"
Disable stack traces in soap faults
JAVA_OPTIONS="-Dcom.sun.xml.ws.fault.SOAPFaultBuilder.disableCaptureStackTrace="false"
${JAVA_OPTIONS}"
fi
export JAVA_OPTIONS
```

- Make sure to keep the parts with `${JAVA_OPTIONS}` on the same line
- This example uses a properties file with the custom name `svl_ws.properties` which is located in the `$OZG_BASE` folder of your OHI Back Office application server environment, but you can also specify the default name: the `$OZG_BASE` folder and `ohibo.properties`.

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

- The second line adds a setting to disable stack traces in soap faults

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_svl_ohi http://ol6ohi:7016
```

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 `app.properties` cannot be read, messages as below will show up:

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

### 3.5.2 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 (`app.properties`).

### 3.5.3 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.

### 3.5.4 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 SVL 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 app.properties file for each Managed Server.
- Configure each Managed Server to start up with the appropriate app.properties. Add multiple tests for Managed Server names in setUserOverrides.sh to specify a different file name for each Managed Server.
- Deploy the appropriate version of the SVL application to its corresponding Managed Server and give it a unique deployment name to identify its deployment.

### 3.5.5 Publishing the deployed services

After you have deployed the web services, perform a few small initial tests.

- In the Admin Console, navigate to an arbitrary SVL web service, and there go to the *Testing* tab.

Settings for OhIPolicyService

Overview Configuration Security **Testing** Monitoring

Use this page to test that your Web service is deployed and that it is working as expected. In the table, expand the name of the Web service to see a list of its test points. Click **?WSDL** to view its dynamic WSDL in a separate browser window. Click **Test Client** to invoke a new browser window where you can test each operation individually by entering parameter values, executing the operation, and viewing the results.

Deployment Tests

| Name                               | Test Point  | Comments                                |
|------------------------------------|-------------|-----------------------------------------|
| OhIPolicyService                   |             | Test points for this Webservice module. |
| /OHIBOWebservices/OhIPolicyService | ?WSDL       | WSDL page on server MS_ONTW             |
| /OHIBOWebservices/OhIPolicyService | Test client | Test client on server MS_ONTW           |

- Click on the *?WSDL* link and check that the WSDL file is displayed in a new browser window.

NOTE: the *Test client* link is only available for domains that run in Development Mode. For domains running in Production mode, please see paragraph 3.6.4 *Testing with SoapUI* to test the *isAlive* request operation.

If you have not tested before with security enabled, please read on in the following paragraph how to test with security enabled. When you are sure the deployment succeeded you can proceed with publishing the WSDL's to your developers and testers.

In case you want to make WSDL's available to selected users in your organization, you can find the WSDL for each service in the *Testing* tab of each service. You can also publish the WSDL URLs for the provider web services.

The format of the service URL is <servername>:<port>/<web application name>.

The WSDL will contain the https based address for the service.

#### ..3.5.5.1 *Retrieving the WSDL from the EAR file*

If necessary you can retrieve the WSDL of a service before the associated EAR file is deployed:

- Open the SVLBOWS.ear file to find the WAR file of the service you are interested in.
- Locate the WSDL and XSD files in the WEB-INF/wsdl folder of the WAR archive.

Notes:

- You can extract files from EAR or WAR files with jar (similar parameters as 'tar') or use a zip utility (also on a Windows based platform).
- After patching OHI Back Office, the latest version of SVLBOWS.ear is stored as \$OZG\_BASE/java/SVLBOWS.ear.

## 3.6 Security Aspects

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

Your SVL services 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 SVL 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 SVL provider web services are configured to use a default policy (policy:Wssp1.2-2007-Https-BasicAuth.xml). This policy requires HTTPS communication and a username/password combination.

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

This paragraph provides some pointers to get started:

- Using the default security policy
- Overruling the default policy
- Restricting access with custom roles
- Testing with SoapUI

### 3.6.1 Using the default security policy

Select 'Configuration > WS Policy' for the web service (not the module!) of your choice:



Note that this information is visible only if the SVLBOWS application is active.

View the WSDL to examine the policy:

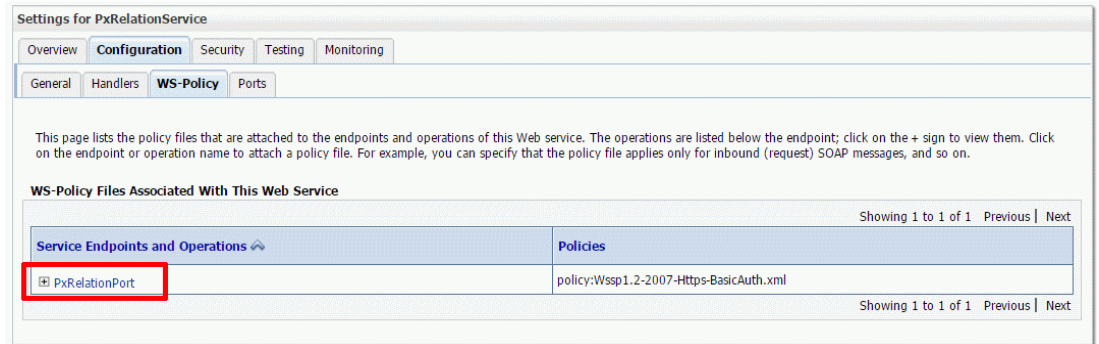
```
-<!--
 Published by JAX-WS RI (http://jax-ws.java.net). RI's version is JAX-WS RI 2.2.11-b150616.1732 svn-revision
-->
- <wSDL:definitions targetNamespace="http://www.oracle.com/insurance/ohibo/pxrelation/pxrelationmessages/v1">
 <wsp:UsingPolicy wssutil:Required="true"/>
 - <ns0:Policy wssutil:Id="Wssp1.2-2007-Https-BasicAuth.xml">
 - <ns1:TransportBinding>
 - <ns0:Policy>
 - <ns1:TransportToken>
 - <ns0:Policy>
 - <ns1:HttpsToken>
 - <ns0:Policy>
 - <ns1:HttpBasicAuthentication/>
 </ns0:Policy>
 </ns1:HttpsToken>
 </ns0:Policy>
 </ns1:TransportToken>
 - <ns1:AlgorithmSuite>
 - <ns0:Policy>
 <ns1:Basic256/>
 </ns0:Policy>
 </ns1:AlgorithmSuite>
 - <ns1:Layout>
 - <ns0:Policy>
 <ns1:Lax/>
 </ns0:Policy>
 </ns1:Layout>
 - <ns1:IncludeTimestamp/>
 </ns0:Policy>
 </ns1:TransportBinding>
</ns0:Policy>
- </wSDL:types>
```

This means the default security policy requires you to send a username, password as well as a timestamp (!) to authenticate a call. When using SoapUI for testing, as described later, this will become more clear.

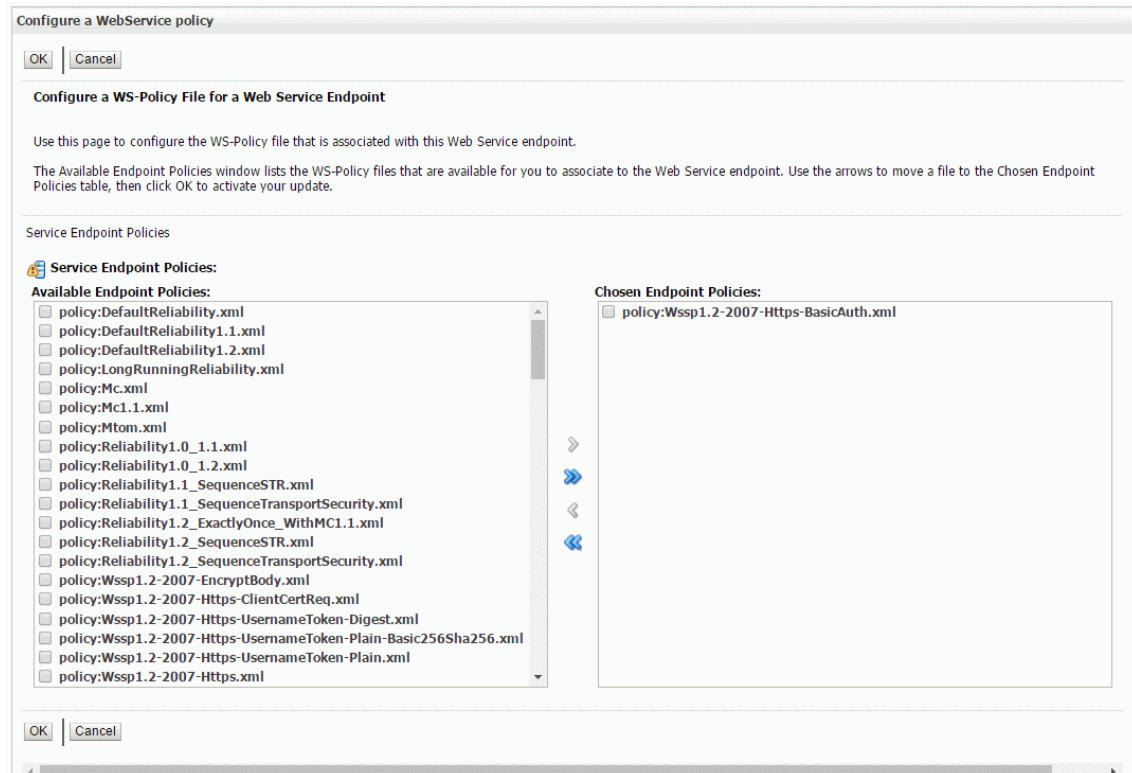
### 3.6.2 Overruling the default policy

You may replace the default policy with a stronger policy.

Select 'Configuration > WS-Policy' for the service of your choice:



Select the end point to list the policies which are applied to the end point:



For testing purposes, on non-production environments only: if you experience problems in calling your web services, you could use this page to remove the security policy. That way, you can test if the Web Service works properly without a security policy enabled. Of course you should only do this temporary and re-enable it again as soon as possible.

Of course you may also opt for using specialized security solutions like Oracle Web Services Manager. In such a situation you may of course also disable the policy and leave the security implementation to such a product.

### 3.6.3 Restricting access with custom roles

The default policy allows any valid WLS user to access the SVL services. This includes the 'weblogic' user (!).

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



**NOTE: As part of our internal testing we found that WLS 12c versions 12.2.1.2.0 and 12.2.1.3.0 allow the creation of custom security policies but do NOT enforce these policies at runtime.**

**Effectively this means that ANY authenticated Weblogic user may access a HSL application deployed with 'Custom Roles and Policies' disregarding the authorization that has been configured.**

**This bug has been fixed in patch 28278427. This patch is only available for Weblogic 12.2.1.3.**

- Create one or more global roles in the security realm used for the SVLBOWS application. For example, SvlAccessRole (to be used for all SVL services) or SvlPxRelationRole (to be used for a single SVL service).
- 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 SVL 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).

### 3.6.4 Testing with SoapUI

---

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

NOTE:

WebLogic 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 older versions of SoapUI (e.g. 5.2.1) do 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 5.3.0 does enable TLS 1.2 by default.

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

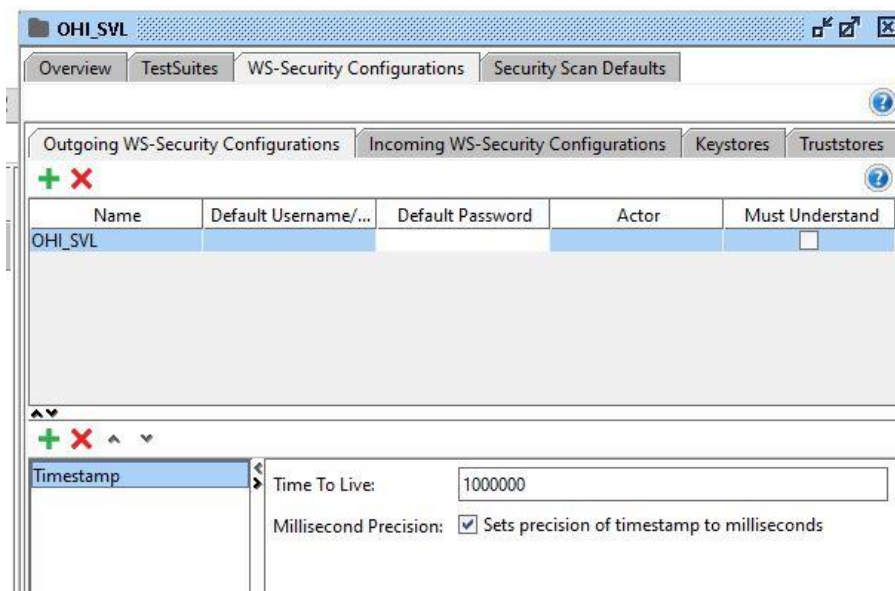
The following procedure should work if you deployed the SVL application using default security policies:

- Create a new SOAP project based on the service WSDL.
- Create requests (these should be SSL requests)
- Create an 'Outgoing' WS-security configuration at project level to include a timestamp and a time to live for the timestamp (e.g. 1000ms)
- Open request for 'isAlive' operation.

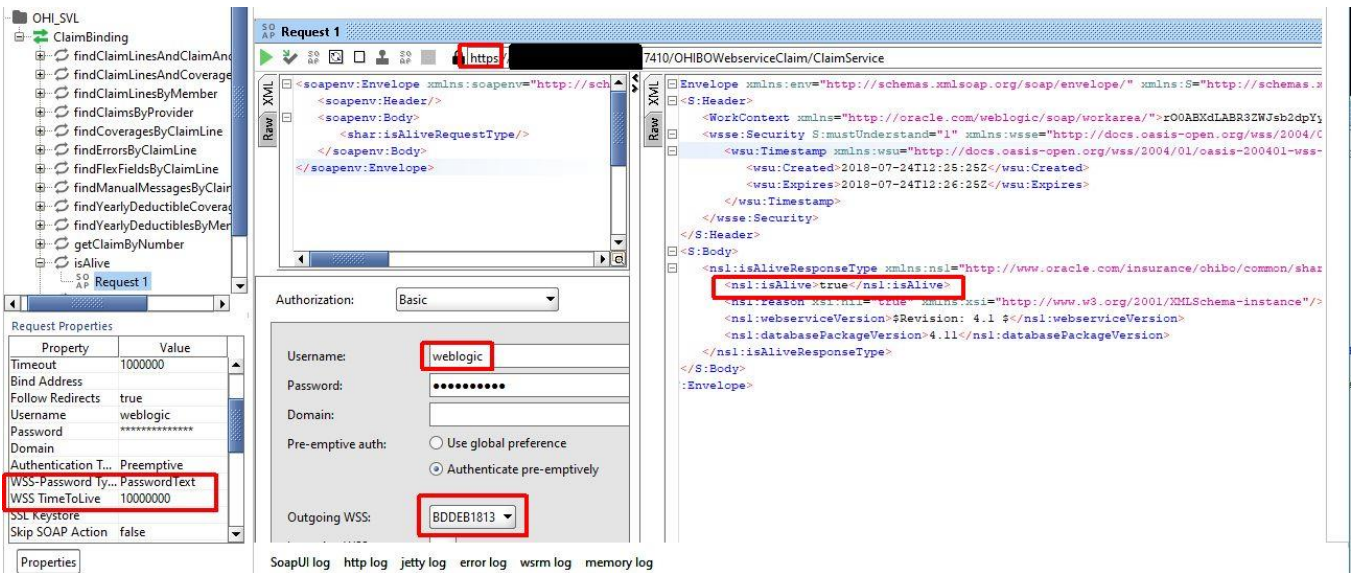
- Set the properties in the “Properties” panel in the lower left corner, or in the “Auth” panel (visible after you have opened the actual request message window, change Authorization from None to Basic):
  - Username= <user defined in weblogic>
  - Password= <password defined in weblogic>
  - Authentication Type = Preemptive
  - WSS-Password Type=PasswordText
  - WSS TimeToLive= 1000000
- Send the “isAlive” request. The request should succeed because ‘weblogic’ is a valid WLS user.
- If your response is empty and returns within a few milliseconds your calling configuration is not ok (leaving for example the password empty results in a HTTP/1.1 200 OK in the raw message)
- 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 start up settings in \$DOMAIN\_HOME/bin/setUserOverrides.sh:

```
JAVA_OPTIONS="-Dssl.debug=true" ${JAVA_OPTIONS}"
```

Example of a WS-security configuration with 1000ms ‘Time To Live’ for the Timestamp setting (this window is accessed on the SoapUI project level by double clicking or choosing the right mouse menu ‘Show Project View’):



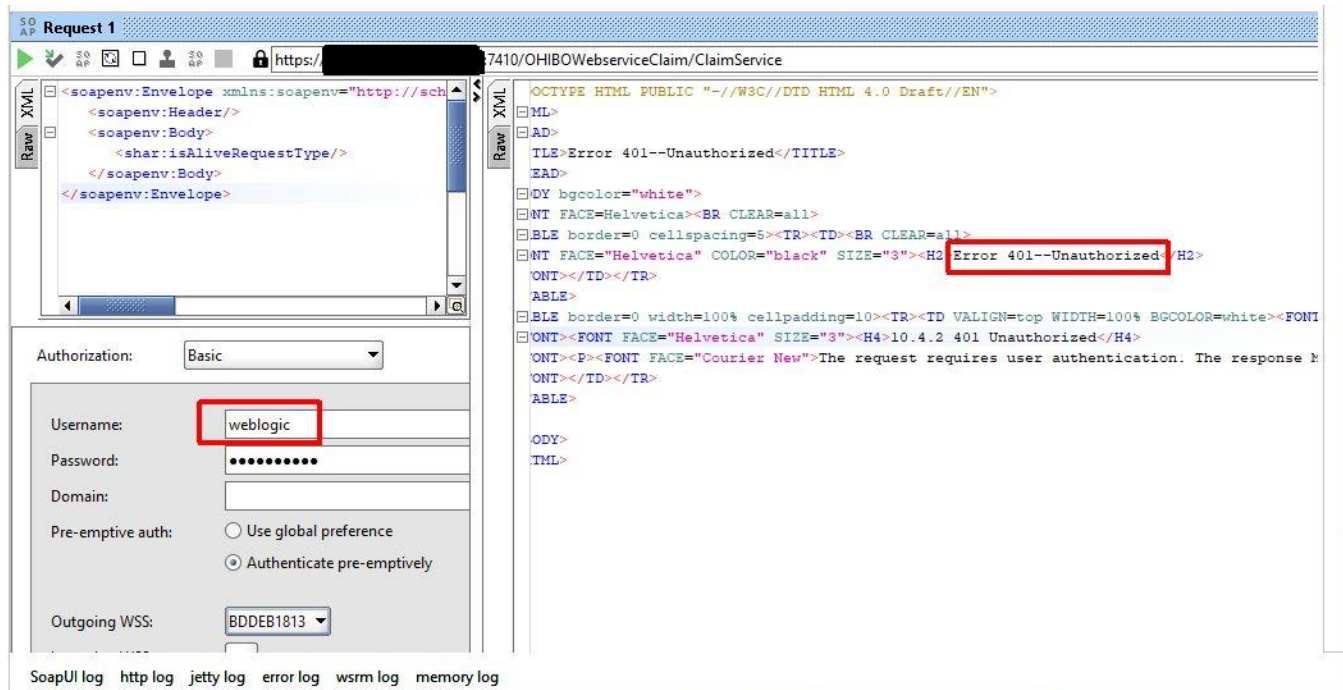
Testing the isAlive operation with an authorized WLS user (created ‘scott’ and linked to the Claim service through group ‘AppTesters’):



Note:

- SSL connection
- Preemptive authentication type
- Basic Authentication
- Outgoing WS-security configuration (choose the configuration you created at the project level)
- Received a valid response.

Finally, an example with the non-authorized WLS user 'weblogic' (we did not assign 'weblogic' to the fictitious 'SvIPxRelationRole' needed to access the PxRelation service):



Note:

- Same authorization but different user credentials as in the example with 'scott'.
- Error message indicates that access was denied to the service.

## 4 Configuration files for provider web 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 web services properties file

The Back Office properties file for the SVL web services is specified in the startup options of the Managed Server with:

```
-Dapp.properties
```

The file (suggested name 'svl\_ws.properties') contains a number of properties. For specifying logging functionality these values are generic for all web services and specified only once.

Other settings can be set specific per web service and may differ for a certain web service (a web service can support several web service operations and typically references a single WSDL). The properties are named below and you should of course adapt the values to the needs of your organisation.

You need to set the logging properties only once as they are not specific for each web service and specify the logging settings for the deployment as a whole (using Java Util Logging, in short JUL):

1. A value to specify the directory + filename that will be created for logging.
2. The severity level for the logging. This can be SEVERE, WARNING, INFO, CONFIG, FINE, FINER or FINEST (not all these levels are actually used within this web services implementation).
3. The maximum size of the file, in bytes. If this is 0, there is no limit. The default, when omitted or an invalid value specified, is 1000000 (which is 1 MB). Logs larger than 1MB roll over to the next log file.
4. The number of log files to use in the log file rotation. The default is 1 which produces a maximum of 1 log file, meaning that when the maximum size is reached the file is emptied and not saved to a 'rotation file' resulting in recent log information being deleted. We advise to specify a value of 2 or higher for that reason.
5. A value to specify to append to the logfile or not when the application is restarted. Possible values are True and False (the default is True).

Additionally you need to set 2 properties for each web service:

- A value for the application user that will be used when web service operations are executed. This determines the user identity that is used for logging changes to the data, and which language is used for messages. This is the 'usercontext.user.username' property. The value must be the Oracle username of a registered BackOffice user (in Dutch: "Functionaris"). The examples in this document use SVL\_FUNC\_USER.

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

NOTE: Do not use the value "MANAGER". Records created and update by SVL 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.

- A value for the data source within WebLogic that will be used for connecting to the database. This can (and generally will) be the same data source for all web services but you can use different data sources. This can be used for example to implement different availability per web service or even prevent a web service from being used. The value must be the JNDI name of an existing DataSource. You need to specify 2 forward slashes to indicate a forward slash in the JNDI name.

**BEWARE:** Be sure you do not add any space after the values, before an end of line character. This may lead to malfunction of the web services.

So there will be five lines in the properties file like:

```
common.logging.filename=/u01/app/oracle/product/OHI/vohi/svl_ws.log
common.logging.loglevel=FINEST
common.logging.loglimit=10000000
common.logging.logcount=10
common.logging.logappend=TRUE
```

And for each web service, there will be two lines in the properties file like:

```
<web service name>.callcontext.usercontext.user.username=SVL_FUNC_USER
<web service name>.datasource.jndiname=jdbc//DSvohi
```

e.g.

```
procedureauthorization.callcontext.usercontext.user.username=SVL_FUNC_USER
procedureauthorization.datasource.jndiname=jdbc//DSvohi
```

As new web services are added in new (patch) releases of OHI BackOffice, you will be notified of required changes during the installation of patches, with instructions to add lines to this file.

Web service names starting with “px” are write services (i.e modifying data) based on new standards for implementing the OHI BO web services. The request message passed is (part of) a ‘pixel’ photo of how the end situation after the write activity should look (the ‘demanded’ situation is described and the service needs to determine intelligently what changes need to be executed to finally reach the situation described in the ‘photo’). Pixel photo is abbreviated to ‘px’, to indicate this new type of services.

#### 4.1.1 10.18.1.2.0 properties

---

No changes from 10.17.2.2.

#### 4.1.2 10.17.2.2 properties

---

In release 10.16.2.3.0 five service generic new properties were introduced for specifying the changed logging functionality in this release.

In the file below the properties are ordered alphabetically, except for the five properties that specify the logging of informational and debugging messages, these are listed as a first set of properties.

This properties file includes the latest enhancement for a new Vecozo Authorization related web service (named ‘Vecozomachtiging’, a generated name, not to be mixed up with the already existing ‘Vecozo’).

```
common.logging.filename=/u01/app/oracle/product/OHI/vohi/svl_ws.log
common.logging.loglevel=FINEST
```

```
common.logging.loglimit=10000000
common.logging.logcount=10
common.logging.logappend=TRUE

broker.datasource.jndiname=jdbc//DSprod
broker.callcontext.usercontext.user.username=SVL_FUNC_USER

debtor.datasource.jndiname=jdbc//DSprod
debtor.callcontext.usercontext.user.username=SVL_FUNC_USER

pxdebtor.datasource.jndiname=jdbc//DSprod
pxdebtor.callcontext.usercontext.user.username=SVL_FUNC_USER

claim.datasource.jndiname=jdbc//DSprod
claim.callcontext.usercontext.user.username=SVL_FUNC_USER

groupcontract.datasource.jndiname=jdbc//DSprod
groupcontract.callcontext.usercontext.user.username=SVL_FUNC_USER

invoice.datasource.jndiname=jdbc//DSprod
invoice.callcontext.usercontext.user.username=SVL_FUNC_USER

payment.datasource.jndiname=jdbc//DSprod
payment.callcontext.usercontext.user.username=SVL_FUNC_USER

paymentscheme.datasource.jndiname=jdbc//DSprod
paymentscheme.callcontext.usercontext.user.username=SVL_FUNC_USER

policy.datasource.jndiname=jdbc//DSprod
policy.callcontext.usercontext.user.username=SVL_FUNC_USER

preauthorization.datasource.jndiname=jdbc//DSprod
preauthorization.callcontext.usercontext.user.username=SVL_FUNC_USER

procedureauthorization.datasource.jndiname=jdbc//DSprod
procedureauthorization.callcontext.usercontext.user.username=SVL_FUNC_USER

provider.datasource.jndiname=jdbc//DSprod
provider.callcontext.usercontext.user.username=SVL_FUNC_USER

providercontract.datasource.jndiname=jdbc//DSprod
providercontract.callcontext.usercontext.user.username=SVL_FUNC_USER

pxclaim.datasource.jndiname=jdbc//DSprod
pxclaim.callcontext.usercontext.user.username=SVL_FUNC_USER

pxpolicy.datasource.jndiname=jdbc//DSprod
pxpolicy.callcontext.usercontext.user.username=SVL_FUNC_USER

pxprocedureauthorization.datasource.jndiname=jdbc//DSprod
pxprocedureauthorization.callcontext.usercontext.user.username=SVL_FUNC_USER

pxprovidercontract.datasource.jndiname=jdbc//DSprod
pxprovidercontract.callcontext.usercontext.user.username=SVL_FUNC_USER

pxrelation.datasource.jndiname=jdbc//DSprod
pxrelation.callcontext.usercontext.user.username=SVL_FUNC_USER

pxyagreement.datasource.jndiname=jdbc//DSprod
pxyagreement.callcontext.usercontext.user.username=SVL_FUNC_USER

quotations.datasource.jndiname=jdbc//DSprod
quotations.callcontext.usercontext.user.username=SVL_FUNC_USER

realtimeclaimfile.datasource.jndiname=jdbc//DSprod
realtimeclaimfile.callcontext.usercontext.user.username=SVL_FUNC_USER

receipt.datasource.jndiname=jdbc//DSprod
receipt.callcontext.usercontext.user.username=SVL_FUNC_USER
```

```
receivable.datasource.jndiname=jdbc//DSprod
receivable.callcontext.usercontext.user.username=SVL_FUNC_USER

referencedata.datasource.jndiname=jdbc//DSprod
referencedata.callcontext.usercontext.user.username=SVL_FUNC_USER

relation.datasource.jndiname=jdbc//DSprod
relation.callcontext.usercontext.user.username=SVL_FUNC_USER

vecozo.datasource.jndiname=jdbc//DSprod
vecozo.callcontext.usercontext.user.username=SVL_FUNC_USER

vecozomachtiging.datasource.jndiname=jdbc//DSprod
vecozomachtiging.callcontext.usercontext.user.username=SVL_FUNC_USER
```

## 5 OHI release upgrade and provider web services

When you need to redeploy the provider web services (the .ear file) because a new version is delivered in an OHI release this is relatively simple. Please follow the steps below:

- ✓ Check your web service properties file (typically ohibo.properties or svl\_ws.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 .ear 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 implement this.

- ✓ Choose 'Activate Changes'.
- ✓ Refresh the screen a few seconds after having activated the changes. The previous version should now show a Retired state as in the example below:

|                          |                                                                                                                                                                                         |         |      |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|------|
| <input type="checkbox"/> |   SVLBOWS (v4.21) | Retired |      |
| <input type="checkbox"/> |   SVLBOWS (v4.22) | Active  | ✓ OK |



- ✓ Inform the communities which use the web services of the availability and publish the WSDL addresses to them, especially when it has changed.

If the update of the deployment does not succeed delete the deployed application (first stop 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 Installing and deploying web service consumers

This chapter describes how to install and configure your environment for implementing web service consumers. The major part of the activities has to be executed only once.

The web service consumers are implemented by means of the “Web Services Call-Out” feature as provided for the database. This feature enables the publishing of a WSDL into a database PL/SQL package. A java based web services client proxy is created as well as a PL/SQL wrapper for this client proxy.

### 6.1 Preparation of your database environment

In order to enable the web service call-out functionality the database has to be configured to support this. The installation and configuration manual describes this in paragraph “Creating an OHI Back Office database”.

The following subparagraphs in that paragraph describe this:

**Install/check database component Oracle JVM**

**Prepare the OHI owner account for using the Oracle JVM**

**Install the DBWS callout utility in the OHI schema owner account**

### 6.2 Prepare a secure setup

The code for consuming web services, as implemented within OHI Back Office, executes calls originating from the database server. This means these calls must be able to access a certain endpoint (an URL) accessible from the database server.

OHI strongly advises to set up an application server or middleware tier which acts as endpoint, processes the request, passes it on to the target environment, processes the response message and passes this back to the web service consumer.

This kind of functionality is typically implemented using a service bus.

For most web service consumers an application server or middleware tier is a requirement in order to map the OHI specific WSDL to an outside world WSDL. This kind of transformation can easily be implemented through service bus functionality.

Other (security!) requirements can/should (security!) be implemented in or provided by intermediate middleware such as a service bus.

It is outside the scope of this document to describe the middleware setup. These kinds of requirements and functionalities are fairly standard for a service oriented environment.

### 6.3 Deployment of web service consumers

Deployment of a web service consumer typically means:

- the .jar file as provided in the OHI release should be loaded into the database
- the related object types and PL/SQL wrapper package need to be created

These actions are automated and executed during the database installation step of the release installation.

---

## 7 Appendix A - Removing a WLS domain

If 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.

## 8 Appendix B – Compare version information in EAR files

As of release 10.14.2.0.0, OHI Back Office versions the SVL webservice operations, not the XML Schema Definitions or the provider web services.

The SVLCMPRV.pl script compares two EAR files and lists version differences between web service operations.

The script can be used from release 10.14.2.0.0 onwards.

### 8.1 Invocation

SVLCMPRV.pl resides in the \$OZG\_BASE/sh directory

SVLCMPRV.pl should be run from the command prompt. The script requires two .ear files as parameters.

For example, to compare versions 10.15.1.3.0 and 10.15.3.0.0:

```
perl $OZG_BASE/sh/SVLCMPRV.pl \
$OZG_PATCH/10.15.1.3.0/java/SVLBOWS.ear \
$OZG_PATCH/10.15.3.0.0/java/SVLBOWS.ear
```

### 8.2 Operation

SVLCMPRV.pl extracts each EAR file to a temporary folder before comparing the version numbers of the operations defined in the WSDL files.

A higher version number means that at least one definition in the type hierarchy used by the operation has changed since the previous version.

Example:

- Assume that for release 10.15.3.0.0 a region code has been added to the PxAddressType. The PxAddressType is part of the type hierarchy for PxAbstractRelationType, used in the request message for the PxRelationService.WriteRelation operation.
- If the previous version for PxRelationService.WriteRelation was 'v1' (10.15.1.3.0 SVLBOWS.ear), then the new version will be 'v2'.
- If we compare the SVLBOWS.ear files for these two versions we should see that PxRelationService.WriteRelation has 'v1' for 10.15.1.3.0 and 'v2' for 10.15.3.0.0

A comparison between web service operations may yield the following results:

- An operation is present in the 10.15.1.3.0 SVLBOWS.ear but not in the 10.15.3.0.0 SVLBOWS.ear file. This may mean the operation has been removed or renamed.
- An operation is present in the 10.15.3.0.0 SVLBOWS.ear but not in the 10.15.1.3.0 SVLBOWS.ear. This may mean the operation has been added or renamed.
- A change in revision number is reported. This may mean that the message or the underlying XML schema definition has changed.
- If no change has been made to the operations and/or revision numbers the message: INFO: No differences found will be given.

## 8.3 Output

The output is written to the console. Sample output:

```
INFO: BrokerService.findBrokerRequestTypeMessage(v1) only occurs
in [${OZG_OPL}/10.15.3.0.0/java/SVLBOWS.ear].brokerservice.wsdl
INFO: BrokerService.findBrokerResponseTypeMessage(v1) only occurs
in [${OZG_OPL}/10.15.1.3.0/java/SVLBOWS.ear].brokerservice.wsdl
INFO: BrokerService.getBrokerDetailsByBrokerCodeRequestTypeMessage
[${OZG_OPL}/10.15.1.3.0/java/SVLBOWS.ear].brokerservice.wsdl:v1
[${OZG_OPL}/10.15.3.0.0/java/SVLBOWS.ear].brokerservice.wsdl:v2
```

---

## 9 Appendix C - Managing security policies using WSLT

All web services in the SVLBOWS application are configured with a default security policy (policy:Wssp1.2-2007-Https-BasicAuth.xml).

OHI BO administrators can use the WLS console to manage the security policies on a per service basis.

Since there are currently over 20 web services in the SVLBOWS ear application, managing the security policies for the SVL web services manually is a burden if you have a DTAP landscape with many environments for development and testing.

To automate the administration of WLS environments, the WebLogic Scripting Tool (WLST) can be used.

This chapter aims to help you manage the SVL provider web service policies through WSLT and explains some of the restrictions that apply.

### 9.1 Relevant WLST commands

Generally, an administrator records a manual session to generate a WLST script for later editing.

This is not possible for managing web service policies, so you need the WLS documentation to find the required WLST commands.

The relevant commands are:

- `listWebServices(application, detail=true)`  
Lists the web services and security policies of an application.
- `listWebServicePolicies`  
Lists the service policies attached to a given service of an application.
- `attachWebServicePolicy`  
Attach a service policy to a given service of an application.
- `detachWebServicePolicy`  
Detach a service policy from a given service of an application.

### 9.2 Requirements

#### 9.2.1 WLS version

---

The WLST commands for listing, detaching and attaching web service security policies are available by default in the WebLogic 12 Infrastructure installation. See [Oracle® Fusion Middleware WLST Command Reference for Infrastructure Components](#) for documentation.

#### 9.2.2 OWSM needed to attach policies using WLST

---

If you use WLST to attach a security policy to a web service you will use the `attachWebServicePolicy` command. For this command you must have OWSM installed (and have a license to use OWSM).

Notes:

- this requirement does not apply to detaching security policies.

- you can only attach OWSM security policies with WLST
- you can manually attach (Weblogic) security policies using the WLS console.

### 9.3 Restrictions in WLST

The WLST command for attaching a security policy to a web service is `attachWebServicePolicy`.

Unfortunately it can only attach OWSM policies. This was a design decision by the WebLogic development team.

It has the following implications:

- You can detach the default WLS security policy (`policy:Wssp1.2-2007-Https-BasicAuth.xml`) using WLST.
- You cannot re-attach `policy:Wssp1.2-2007-Https-BasicAuth.xml` or any WLS policy using WLST.
- If you want to attach any policy using WLST you must configure OWSM into your domain and select a valid OWSM policy.

Note that you can re-attach `policy:Wssp1.2-2007-Https-BasicAuth.xml` and other non-OWSM policies manually using the WLS console.

## 9.4 Tips and Tricks

### 9.4.1 Capturing `listWebServices` output

---

The `listWebServices` command lists the web services and their current policies. Unfortunately you cannot assign the `listWebServices` objects to a variable for further processing. The next best solution is to capture the output to a file for further processing. We found that it is not possible to redirect the output of `listWebServices` from within WLST. The best option is to write a WLST script and use 'tee' to capture its output to a file like below:

```
wlst.sh myscript.py | tee myoutput.txt
```

### 9.4.2 Listing available OWSM policies

---

If you have OWSM configured, use the following steps for an overview of OWSM policies from the WLS console (example `PxProviderContractService`)

- Lock & Edit
- Select Deployments > SVLBOWSvnnnn
- Select `PxProviderContractService` > Configuration > WS-Policy
- Select `PxProviderContractPort`
- Select Configure .. OWSM
- Select Next

The OWSM policies are now displayed:



**Configure a WebService policy**

Back Next Finish Cancel


**Configure an OWSM WS-Policy File for a Web Service Endpoint**

Use this page to configure the WS-Policy file that is associated with this Web Service endpoint.

The Available Endpoint Policies window lists the WS-Policy files that are available for you to associate to the Web Service endpoint. Use th

---

Service Endpoint Policies

 **Service Endpoint Policies:**

**Available Endpoint Policies:**

- oracle/binding\_authorization\_denyall\_policy
- oracle/binding\_authorization\_permitall\_policy
- oracle/binding\_permission\_authorization\_policy
- oracle/http\_oam\_token\_service\_policy
- oracle/http\_saml20\_token\_bearer\_over\_ssl\_service\_policy
- oracle/http\_saml20\_token\_bearer\_service\_policy
- oracle/no\_authentication\_service\_policy
- oracle/no\_authorization\_service\_policy
- oracle/no\_messageprotection\_service\_policy
- oracle/sts\_trust\_config\_service\_policy
- oracle/whitelist\_authorization\_policy
- oracle/wss10\_message\_protection\_service\_policy
- oracle/wss10\_saml20\_token\_service\_policy
- oracle/wss10\_saml20\_token\_with\_message\_protection\_service\_policy
- oracle/wss10\_saml\_hok\_token\_with\_message\_protection\_service\_policy
- oracle/wss10\_saml\_token\_service\_policy
- oracle/wss10\_saml\_token\_with\_message\_integrity\_service\_policy
- oracle/wss10\_saml\_token\_with\_message\_protection\_service\_policy
- oracle/wss10\_saml\_token\_with\_message\_protection\_ski\_basic256\_service...

**Chosen Endpoint Policies:**

Back Next Finish Cancel

### 9.4.3 Activate application before running WLST commands

Be aware that WLST can only access a running application.

This requires that:

- The AdminServer process is running
- The Managed Server process for is running
- The SVLBOWS application is active

### 9.4.4 Remove temp directories

The following WLST commands create temporary directories which are not removed afterwards:

- listWebServices(detail=true)
- listWebServicePolicies
- attachWebServicePolicy
- detachWebServicePolicy

Each time one of these commands is run, a temporary directory is created with approximately 1GB of data.

This is a known bug (see bug 19295696 for more details).

If you need to free up disk space, the following commands should help:

```
$. ozg_init.env FRS11G2
$ cd $WL_HOME/../../user_projects/domains
$ cd domain
$ cd servers/AdminServer/tmp
$ rm -rf .appmergegen*
```

NOTE: This bug is a duplicate of bug 24929178, which is registered as solved in 12.2.1.3.0 (via backport request bug 25188080, see Doc ID 2067900.1, "Oracle Fusion Middleware 12c Release 2 (12.2) Announcements" -> [12.2.1.3 Bugs Fixed List](#)). The files are still created in Weblogic 12.2.1.3, though.

## 9.5 Example WLST script (polman.py)

The WLST script 'polman.py' was created as an example for managing policies using a CSV file.

The polman.py script supports the following operations:

- 'list' is used to create an ASCII file with the output of the listWebServices command.
- 'csv' is used to create a CSV file from the 'list' output.
- 'detach' is used to detach security policies using a CSV file.
- 'attach' is used to attach security policies using a CSV file.

The idea is that:

1. The administrator uses polman.py to create a CSV file to list the current security policies. This file may be called 'detach.csv'.
2. The file 'detach.py' is then edited to replace the current security policies with the desired policies. The resulting file may be called 'attach.csv'.
3. Finally, the administrator uses polman.py to detach the policies in 'detach.csv' and to attach the policies in 'attach.csv'.

This way it is possible to efficiently manage the security policies of SVL services on a number of environments.

Example of how polman.py is used to support the above steps:

```
$. ozg_init.env SVL12213
$ WLST=$WL_HOME/../../oracle_common/common/bin/wlst.sh
1a. List security policies in text format.
$ $WLST polman.py --username=weblogic --password=welcome1 --host=t3://localhost:7016 --
appName=/ohi_svl/ms_svl_ohi/SVLBOWS --operation=list | tee list.txt
1b. Convert text contents to csv file with current policies.
$ $WLST polman.py --appName=/ohi_svl/ms_svl_ohi/SVLBOWSv4_236 --txtFile=list.txt --
csvFile=detach.csv --operation=csv
3a. Detach policies listed in detach.csv
$ $WLST polman.py --username=weblogic --password=welcome1 --host=t3://localhost:7016 --
appName=/ohi_svl/ms_svl_ohi/SVLBOWS --csvFile=detach.csv --operation=detach
3b. Attach policies listed in attach.csv
$ $WLST polman.py --username=weblogic --password=welcome1 --host=t3://localhost:7016 --
appName=/ohi_svl/ms_svl_ohi/SVLBOWS --csvFile=attach.csv --operation=attach
```

### 9.5.1 CSV format

The CSV format used by polman.py for detaching and attaching policies is as follows:

```
module_name, module_type, service_name, subject_name, policy_name
/ohi_svl/ms_svl_ohi/SVLBOWSv4_236,SVLBOWSv4_236#!ClaimService,wls,ClaimService,ClaimPort,oracle/no_authentication_service_policy
/ohi_svl/ms_svl_ohi/SVLBOWSv4_236,SVLBOWSv4_236#!BrokerService,wls,BrokerService,BrokerPort,oracle/no_authentication_service_policy
/ohi_svl/ms_svl_ohi/SVLBOWSv4_236,SVLBOWSv4_236#!VecozoService,wls,VecozoService,VecozoPort,oracle/no_authentication_service_policy
```

When editing the CSV file, use the module\_name to locate the service, then change the policy\_name to a valid policy.

## 9.5.2 Script source

---

The source of polman.py is given in Appendix D.

## 10 Appendix D – polman.py

This is an example.

```
#
File : polman.py
#
Purpose : Configure security policies using WLST
#
Commands in relevant order.
1.list : list policies for application (redirect to file!)
2.csv : Convert output of 'list' command to a CSV file
3.detach : Detach policies in CSV file
4.attach : Attach (OWSM) policies in CSV file (requires OWSM)
#
Author : Hubert Bakker (Oracle)
#
Note : This is an example, no maintenance is provided.
#
History
03/04/2015 H. Bakker M-4210: Create

import sys
import getopt
import re

Attach policy (requires OWSM)
def attachPolicy(application, module_name, module_type, service_name, subject_name, policy_name):
 print 'Attaching policy ' + policy_name
 print "application : " + application
 print "subject_name: " + subject_name
 try:
 attachWebServicePolicy(application, module_name, module_type, service_name, subject_name,
 policy_name)
 listWebServicePolicies(application, module_name, module_type, service_name, subject_name)

 except Exception, e:
 print "Could not detach:"
 print e

Detach policy
def detachPolicy(application, module_name, module_type, service_name, subject_name, policy_name):
 print 'Detaching policy ' + policy_name
 print "application : " + application
 print "subject_name: " + subject_name
 try:
 detachWebServicePolicy(application, module_name, module_type, service_name, subject_name,
 policy_name)
 listWebServicePolicies(application, module_name, module_type, service_name, subject_name)

 except Exception, e:
 print "Could not detach:"
 print e

process CSV files to detach/attach policies
def processCsv(fileName, operation):
 print('Reading File \'' + fileName + '\'')
 f = open(fileName)
 try:
 for line in f.readlines():
 if line.strip().startswith('#'):
 continue
 else:
 items = line.split(',')
 items = [item.strip() for item in items]
 if len(items) != 6:
 print "=>Bad line: %s" % line
 print "=>Syntax: application, module_name, module_type, service_name, subject_name,
policy_name"
 else:
 (application, module_name, module_type, service_name, subject_name, policy_name) = items
 if operation == 'detach':
 print '=> Detaching ' + policy_name + ' from ' + application + '.' + service_name
 detachPolicy(application, module_name, module_type, service_name, subject_name, policy_name)
 else:
 print '=> Attaching ' + policy_name + ' to ' + application + '.' + service_name
 attachPolicy(application, module_name, module_type, service_name, subject_name, policy_name)

 except Exception, e:
 print "=>Error Occurred"
 print e
 exit()

write a single line to a csv file
```

```

def writeCsv(fout, application, moduleName, moduleType, serviceName, subjectName, policyName):
 fout.write(application + ',' + moduleName + ',' + moduleType + ',' + serviceName + ',' + subjectName +
', ' + policyName + "\n")

convert output of listWebServices() into CSV file for further processing.
def createCsv(appName, txt_file, csv_file):
 process = 0
 moduleName = ''
 moduleType = ''
 serviceName = ''
 subjectName = ''
 policyName = ''

 print('Reading file ' + txt_file + ' looking for ' + appName)
 fin = open(txt_file)
 fout = open(csv_file, 'w')
 try:
 for line in fin.readlines():
 line = line.rstrip()
 if re.match(r'/', line):
 process = 0
 if re.match(appName, line):
 process = 1
 if process == 1:
 # look for module data
 m = re.search(r'moduleName=([^\,]+),\s+moduleType=([^\,]+),\s+serviceName=([^\,]+)', line)
 if m != None:
 # print line to CSV if we have sufficient data
 if (moduleName != ''):
 writeCsv(fout, appName, moduleName, moduleType, serviceName, subjectName, policyName)
 g = m.groups()
 (moduleName, moduleType, serviceName) = m.groups()
 subjectName = ''
 policyName = ''

 # look for port daa
 m = re.match(r'\s+(\w+Port)$', line)
 if m != None:
 subjectName = m.groups()[0]

 # look for policy data, retain only the last policy.
 m = re.match(r'\s+ws-policy\s*:\s*(policy:[^\s]+)', line)
 if m != None:
 policyName = m.groups()[0]

 if (moduleName != ''):
 writeCsv(fout, appName, moduleName, moduleType, serviceName, subjectName, policyName)

 fin.close()
 fout.close()
 print "CSV data written to " + csv_file

 except Exception, e:
 print "==>Error Occurred"
 print e
 exit()

def usage(msg = ''):
 print msg
 print "Usage:\n\tpolman.py"
 print "\t--operation=list"
 print "\t--username=<username> (eg. weblogic)"
 print "\t--password=<password> (eg. secret)"
 print "\t--host=<host> (eg. t3://localhost:7016)"
 print "\t--appName=<application> (eg. /ohi_svl/ms_svl_ohi/SVLBOWsv4)"
 print
 print "\t--operation=csv"
 print "\t--appName=<application> (eg. /ohi_svl/ms_svl_ohi/SVLBOWsv4)"
 print "\t--txtFile=<txt_file> (eg. test.txt)"
 print "\t--csvFile=<csv_file> (eg. test.csv)"
 print
 print "\t--operation=detach"
 print "\t--username=<username> (eg. weblogic)"
 print "\t--password=<password> (eg. secret)"
 print "\t--host=<host> (eg. t3://localhost:7016)"
 print "\t--appName=<application> (eg. /ohi_svl/ms_svl_ohi/SVLBOWsv4)"
 print "\t--csvFile=<csv_file> (eg. test.csv)"
 print
 print "\t--operation=attach"
 print "\t--username=<username> (eg. weblogic)"
 print "\t--password=<password> (eg. secret)"
 print "\t--host=<host> (eg. t3://localhost:7016)"
 print "\t--appName=<application> (eg. /ohi_svl/ms_svl_ohi/SVLBOWsv4)"
 print "\t--csvFile=<csv_file> (eg. test.csv)"

 exit()

```

```

main program starts here.
username = 'weblogic'
password = 'wlPwd77xx'
host = 't3://localhost:7016'
csvFile = ''
operation = ''
txtFile = '/media/sf_shared/test.out'
appName = '/ohi_svl/ms_svl_ohi/SVLBOWSv4_236'

long_opts = ['operation=', 'appName=', 'username=', 'password=', 'host=', 'csvFile=', 'txtFile=']
try:
 opts, args = getopt.getopt(sys.argv[1:], "", long_opts)
except getopt.GetoptError, err:
 print str(err)
 usage()
 sys.exit(2)

Handling get options
operations = { 'list' : 1, 'detach' : 1, 'attach' : 1, 'csv' : 1 }
for opt, arg in opts:
 if opt == "--username":
 username = arg
 elif opt == "--password":
 password = arg
 elif opt == "--host":
 host = arg
 elif opt == "--csvFile":
 csvFile = arg
 elif opt == "--txtFile":
 txtFile = arg
 elif opt == "--operation":
 operation = arg
 elif opt == "--appName":
 appName = arg

if operation.strip() == '':
 usage('No operation specified')

try:
 a = operations[operation]
except KeyError:
 usage('Invalid operation: ' + operation)

if (operation == 'attach' or operation == 'detach') and csvFile.strip() == '':
 usage('no csvFile specified')

if operation == 'list' and appName.strip() == '':
 usage('no appName specified')

if operation == 'csv' and txtFile.strip() == '':
 usage('no txtFile specified')

if operation == 'csv' and csvFile.strip() == '':
 usage('no csvFile specified')

print "Parameters:"
print "appName : " + appName
print 'username : ' + username
print 'password : ' + password
print 'host : ' + host
print 'csvFile : ' + csvFile
print 'txtFile : ' + txtFile
print 'operation:' + operation

if operation == 'list':
 connect(username, password, host)
 listWebServices(appName, detail=true)
 print "Use polman.py --operation=csv to convert output to a csv file."
 exit()
elif (operation == 'detach' or operation == 'attach'):
 connect(username, password, host)
 processCsv(csvFile, operation)
elif operation == 'csv':
 createCsv(appName, txtFile, csvFile)

```