ORACLE
INSURANCE

Oracle Health Insurance Back Office

Back Office Service Layer
Installation & Configuration Manual

version 1.17

Part number: E54856_01

April 23, 2014

CHANGE HISTORY

Release	Version	Changes
10.12.2.0.0	1.8	• Added additional paragraph regarding securing web service access.
10.12.2.2.0	1.9	• ohibo.properties changes for the latest releases up to 2012.02 have been added
10.12.3.0.0	1.10	• Made clear to delete retired deployment before updating again.
10.13.1.1.0	1.11	• Updated a number of screen prints.
10.13.1.4.0	1.12	• Added new ohibo.properties for the latest releases up to 10.13.1.4
10.13.2.1.0	1.13	• Web service consumers are now referenced using this correct term.
10.13.3.0.0	1.14	• The situation as needed for the properties file for 10.13.3 has been documented. Please be aware of the new quotations and providercontract properties (the latter were introduced in a patchset on 10.13.2) and the rename of the collectiveagreement to groupcontract.
10.13.3.0.0	1.15	• 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	• Added changes in the properties file for patchset release 10.13.3.3.
10.14.1.0.0	1.17	• Added changes in the properties file for major release 10.14.1.0.0.

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)

Contents

1 Introduction

With release 2011.02 of the OHI Back Office application a first version of the 'Service Layer' has been released.

The Service Layer is an optional component, which in the long term should offer all mainstream services to retrieve and manipulate the core OHI Back Office data.

Typically this service layer is targeted to ease integration in a Service Oriented environment. However, in order to leverage investments, knowledge and experience and to benefit throughput the services are also offered as plsql services in the database.

This document describes the generic technical details regarding the service layer, how to install and update it and how to change configuration settings.

This Service Layer will at some moment completely replace the existing Connect to Back Office functionality. The pre 2011.02 Connect to Back Office functionality will be deprecated in a future release. In the long run also the existing API layer will be deprecated because the Service Layer offers replacing functionality.

1.1 Provider web services and web service consumer

The first release of the Service Layer in OHI Back Office release 2011.02 offers some initial 'provider web services'. These are services which are offered by OHI Back Office and which can be called ('consumed') from the surrounding environment.

With release 2011.03 also a new framework has been implemented for offering web service consumers. These are to support 'external' services, as seen from the OHI Back Office application, which can be consumed by OHI Back Office.

There is a clear distinction for the implementation of the provider web services and the web service consumers. This manual contains instructions for both but it depends on the requirements of your organisation if you need to setup both type of web service 'components'.

1.2 PL/SQL and SOAP interface

The Service Layer in its current setup offers 2 implementation choices for the provider web services. All functional services are packed in pl/sql packages and can be used for direct calls in the database. This reduces the overhead involved with service calls when the calling code is located in the same database or in a database connected through a database link.

These 'provider' pl/sql services are 'published' as SOAP/HTTP web service in order to use them as document style web services in a service oriented environment.

The provider web service layer is a light weight 'wrapper' around the functional implementation in pl/sql.

For the web service consumers a requirement is to 'consume' the web services from the business layer logic as present in the database. For that reason the implementation choice has been made to implement them in the database so they can be called in plsql. This means that for the web service consumers a plsql based wrapper package will be available. This implementation uses the JVM in the database.

1.3 Usage rights

Customers are required to have the appropriate license rights for using the Service Layer. Customers who have acquired a Connect to Back Office license are currently permitted to install and use the web service component of the Service Layer. This is valid until further notice.

When no Connect to Back Office license is obtained it is allowed to use the web services that are specific to Vecozo functionality. The PreAuthorization web service is an example of this. It is allowed to deploy the web services for implementing this web service.

The corresponding plsql services may not be used when no Connect to Back Office license is obtained.

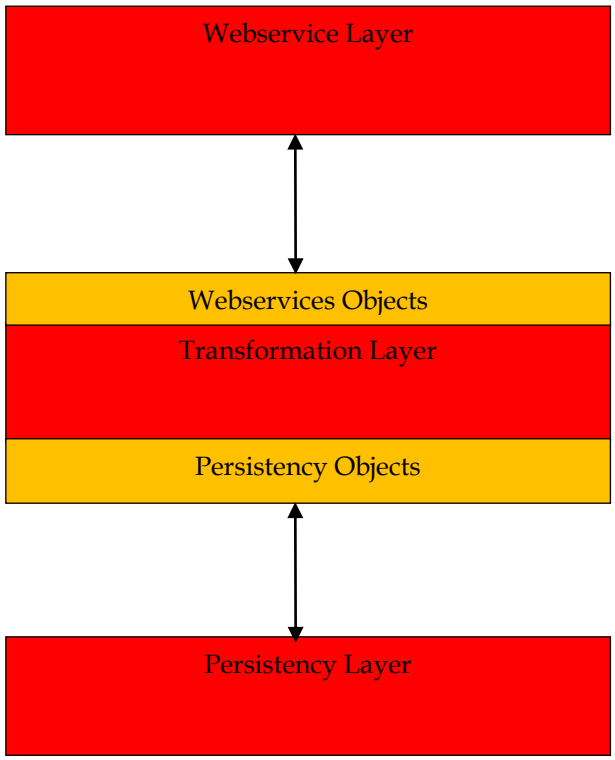
For further information please consult your sales representative.

2 Architectural overview

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

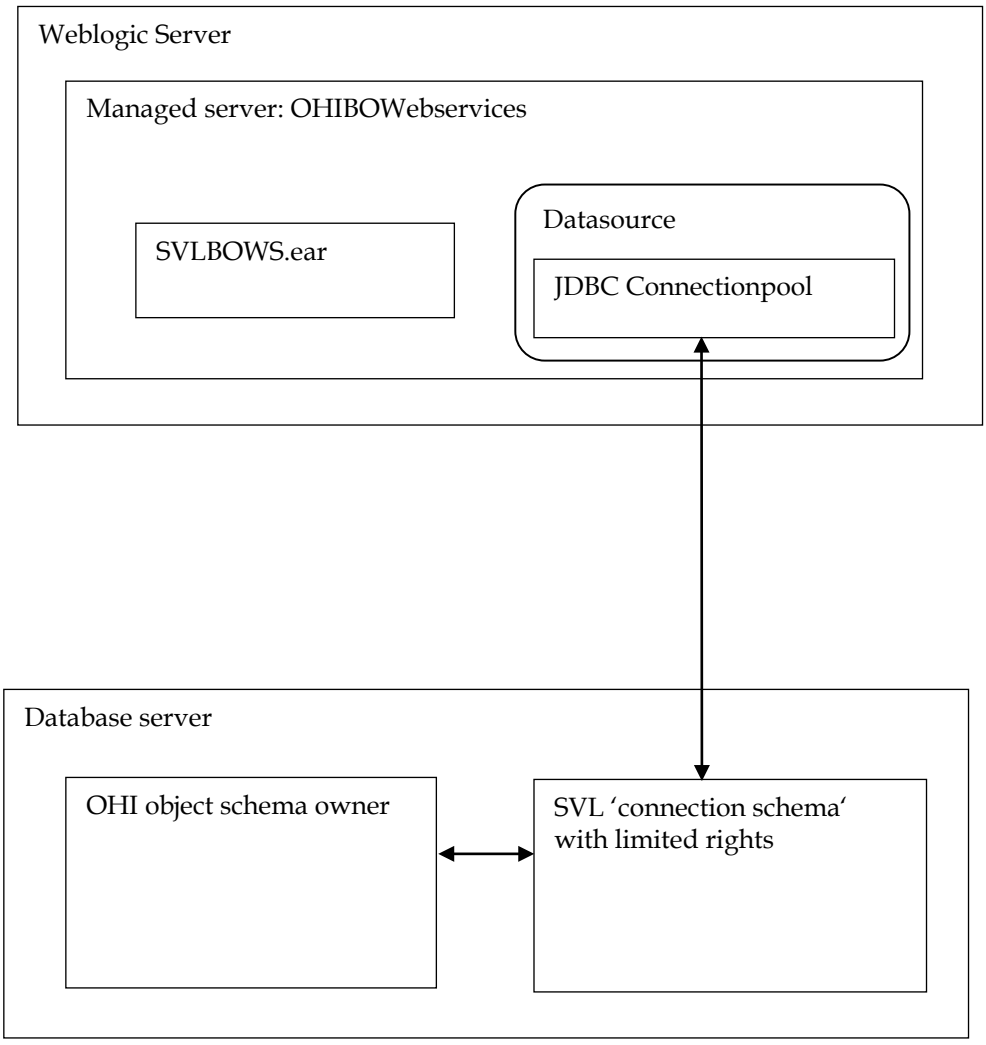
2.1 Provider web services

The provider web services part of the Service Layer is setup as shown below. This is all implemented in Java code and deployed in a WebLogic server.



The persistence layer as shown above offers access to the plsql services in the database.

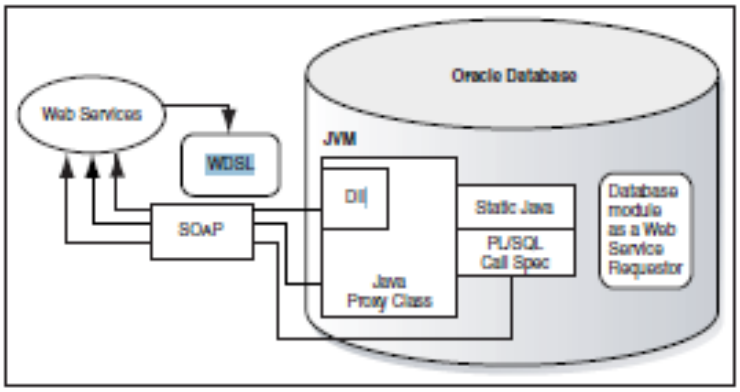
In the schema below it is shown high level how the provider web services are deployed and how the connection to the OHI database is arranged. A separate database account is used with only access rights (execute grants) to the objects that are needed for the Service Layer functionality.



2.2 Web service consumers

For offering functionality to call external web services from within the business logic layer in OHI Back Office the JPublisher tooling is used as provided with the Oracle database. This enables the Oracle Database to be used as a Web Service Consumer.

The figure below shows how a Java client and plsql code implement a web service callout.



JPublisher is used to generate JAX-RPC (java) client proxy classes. This is done based on the WSDL that specifies the external web services. Additional wrapper classes are generated to expose instance methods as static methods, for which plsql wrappers are generated.

The generated java code is loaded into the database as part of the OHI Back Office application components.

3 Installation provider web services

This chapter describes the additional steps necessary for installing the provider web services component of the service layer. Both initially as well as when a new version is distributed.

It is clearly divided into two parts. Which may require different system engineers with different privileges for both parts.

The first part is regarding the installation of the service layer at the database side. And this is mandatory for the optional second part, which consists of deploying the web service layer as application in a WebLogic application server environment.

3.1 Database installation

The first thing is to verify whether you are able to use the Service Layer. Please check, after you implemented the database part of the OHI Back Office release installation, if you have an object (package) SVL_UTILS_PCK in the OHI Back Office schema owner. If not something went wrong regarding the installation of the Service Layer code. If this is incorrect please contact the Support organisation.

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

For improved security you are required to use a separate database account / schema owner for accessing the service layer components. 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. In such a situation please follow the directions as described in Doc[1]. But it is wise and advised to use separate accounts for these purposes.

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

1. Create a schema owner (determine yourself the password policy, temporary tablespace, etc.), for example SVL_USER.

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_grantee => '<your account>')", for example '`execute alg_security_pck.svl_grants(pi_grantee => 'SVL_USER')`'.
   IMPORTANT: This command should be repeated after each new deployment of a new .ear file, so each time the .ear file is delivered in a new release. If you run into ORA-01403 errors during a web service execution your first check should be to run this command again in sqlplus (

This is sufficient for the 2011.02 web services.

For future web services it may be that the consumption requirements of these web services require to activate an additional setting on your database. The web service developers should indicate this based on their needs. If needed a database reorganisation is needed where each indicated table needs to be activated with the ROWDEPENDENCIES setting to enable the row-level dependency tracking mechanism. This will be used to implement an optimistic locking strategy.

3.2 Application Server deployment

When the database account has been created and granted successfully the next step is to deploy the actual web service application.

For this a WebLogic environment is required. This may be the WebLogic environment that is also used for servicing the OHI Back Office user interface and batches. When that same software stack is used it still may be decided whether a new Managed Server in the already existing WebLogic domain is used or whether a new domain will be created. It is strongly advised to create a new WebLogic domain, despite the additional overhead of a separate domain. This prevents any unintended interaction and influence, or contradicting requirements, of the user interface and the web service application.

Of course it is also possible to deploy the web services in a completely different WebLogic environment on a separate server. This has the advantage that you can separately upgrade or patch the different WebLogic environments. Or implement a workload distribution.

It is also an option to deploy the web service application more than once in different environments. Beware however that you deploy the service layer web services only once in a Managed Server or a cluster of Managed Servers.

For testing purposes it may be wishful to deploy the web service application more than once in a WebLogic installation, within the same domain. In such situations you should use separate Managed Servers to deploy the web service application to.

This paragraph will provide you with an outline how to deploy the OHI Back Office Service Layer web services. It should give you a good starting point to make your own detailed deployment choices.

The following requirements/limitations apply in order to prevent potential issues as much as possible:

- ✓ A certified WebLogic Server version is needed.
- ✓ The web services must be deployed on a single Managed Server or a cluster of Managed Servers (the 'target').
- ✓ The web services may not be deployed on a Managed Server where also the forms and reports services as used by OHI Back Office are deployed on. This means that the Managed Server may not be part of a cluster on which the forms and reports services are deployed.
- ✓ One deployment can only service one single Back Office environment (it connects to a specific OHI Back Office 'instance').

When the web service application needs to be deployed more than once (for servicing different OHI Back Office environments) each deployment should be on its own Managed Server or Cluster.

Several options are described (where it is expected you are familiar with the WebLogic concepts like 'domain', ' Managed Server', 'Cluster', etc.):

- ✓ Creating a separate domain with a single Managed Server

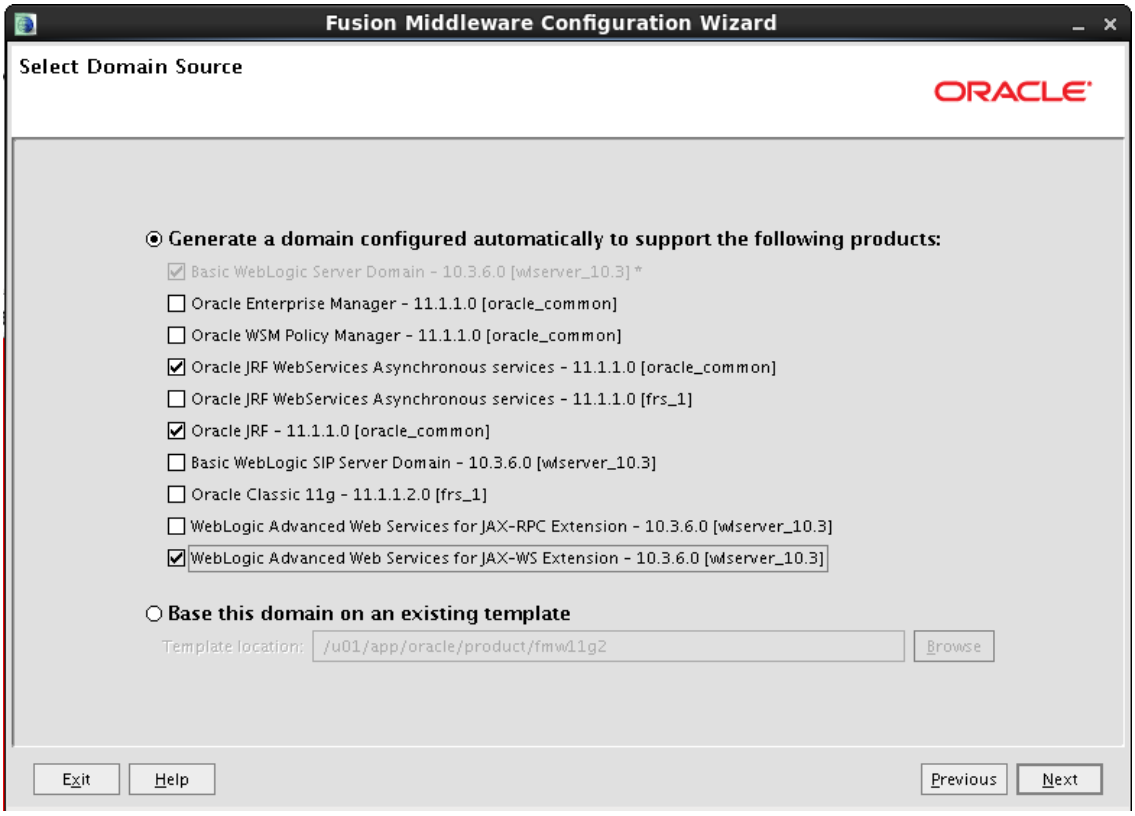
- ✓ Creating a separate domain with a cluster of 2 Managed Servers
- ✓ Adding a Managed Server to an existing domain

In the paragraphs below the setup of a new domain is discussed including configuring Managed Servers, a machine definition, datasources, etc. In these paragraphs several deployment options are described for the web services.

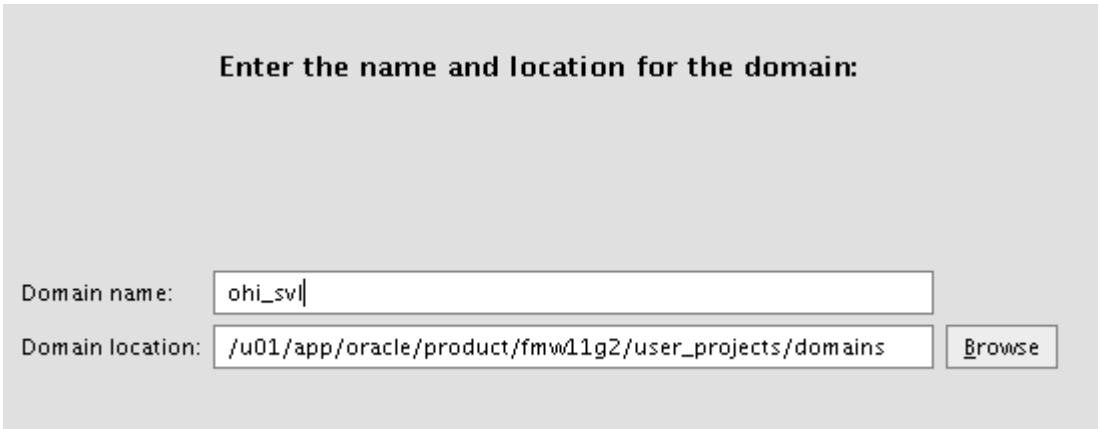
3.2.1 Creating a domain

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

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



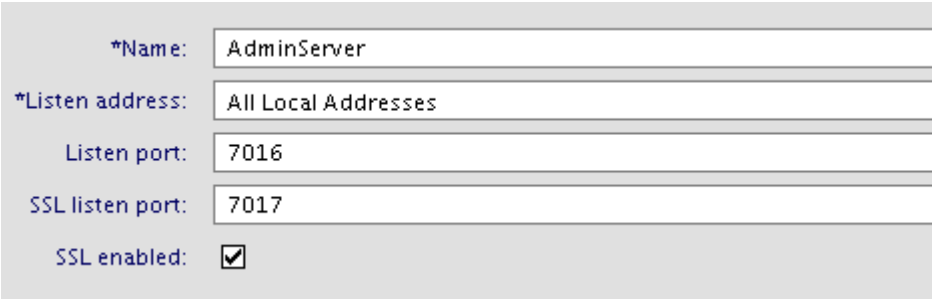
Next specify a domain name and eventually a non default location:



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

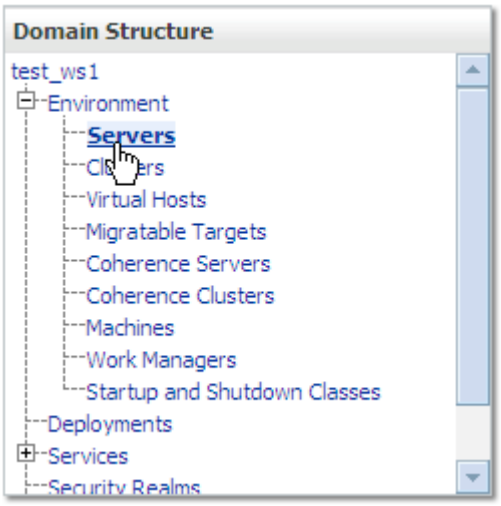
In this documentation as only optional additional configuration is chosen for the Administration Server in order to have the same starting point for additional configuration options you may want to choose later:

For the Administration Server a port number should be specified which is not yet used. Enable SSL for enabling secure connections. An example using non default ports is shown below.

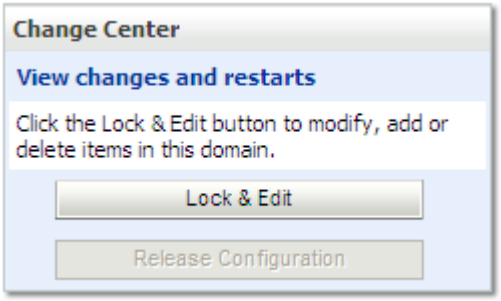


3.2.2 Creating Managed Server(s)

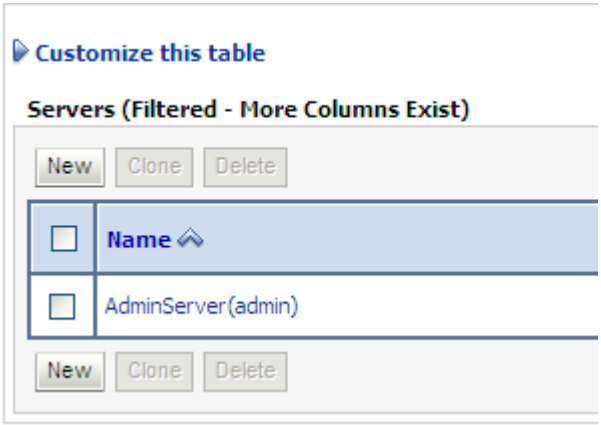
Start the Administration Server (of the existing or newly created domain) using the startWebLogic.sh script (this is present in the root folder of the domain folder, which you created through the Configuration Wizzard). When it is started choose the Servers option in the left panel:



In the change centre choose for Lock & Edit:



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

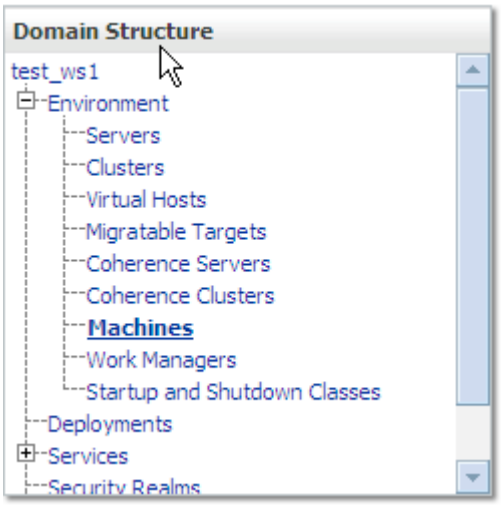


Main things you have to specify are a name for the Managed Server and the port where it listens on. It might be wise to specify the name of the environment in the name of the Managed Server.

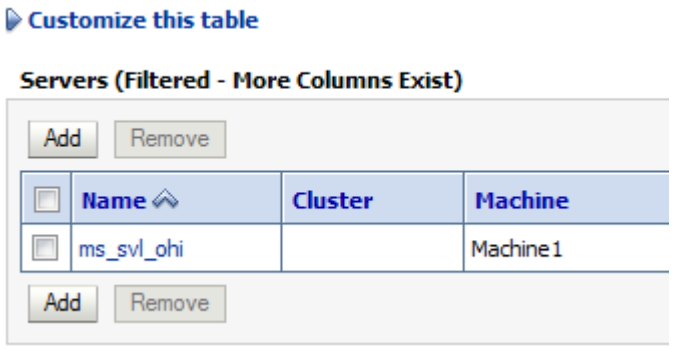
At this moment you can choose whether you want to make the Managed Server part of a Cluster or not. If no Cluster exists yet you can create one; if there is an existing Cluster you can make the Managed Server a member of the Cluster.

3.2.3 Creating a machine

In order to easily start-up Managed Servers it is advised to create a machine definition:



When you create a new machine you can assign Managed Servers to it. In the example below Managed Server ms_svl_ohi is assigned to Machine1.



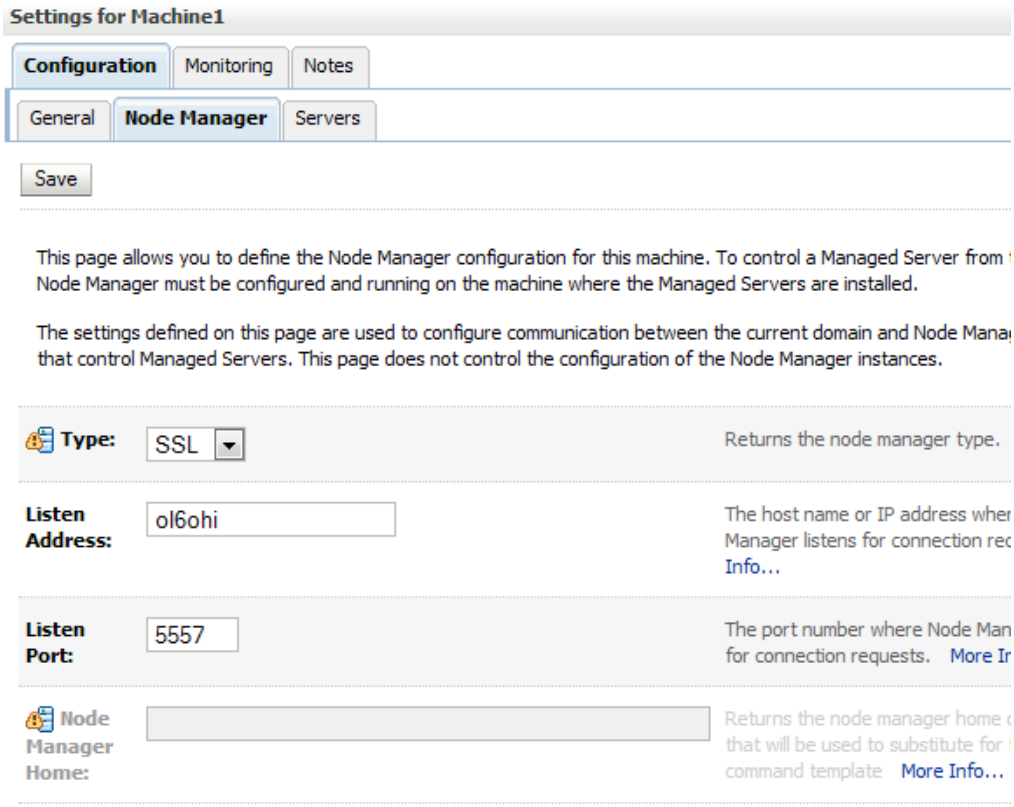
When you start a Node Manager you can use the console to start the Managed Servers through the Node Manager. When you are going to use this instead of starting a Managed Server script wise be sure you set the following property:

StartScriptEnabled=true

Do this in the file nodemanager.properties which is located in the <WebLogic home>/common/nodemanager folder. This is necessary in order to have the Node Manager apply the startup script which sets the environment variables.

It is required that you associate the machine with the node manager in order to have the node manager being able to start the server within the domain of the machine definition.

Do this in the node manager tab for the machine definition like in the example below:



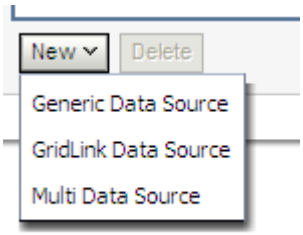
Make sure the listen address is the actual listen address that is used by the Node Manager (this is passed as first parameter to the `$WL_HOME/server/bin/startNodeManager.sh` shell script).

Next to that 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 will be done automatically when you start the Managed Server in the console after you have started the AdminServer for the domain). You need to do this as well for the AdminServer in the …/security folder of the AdminServer folder if you want to prevent specifying the username/password when running startWebLogic.sh.

3.2.4 Creating a datasource

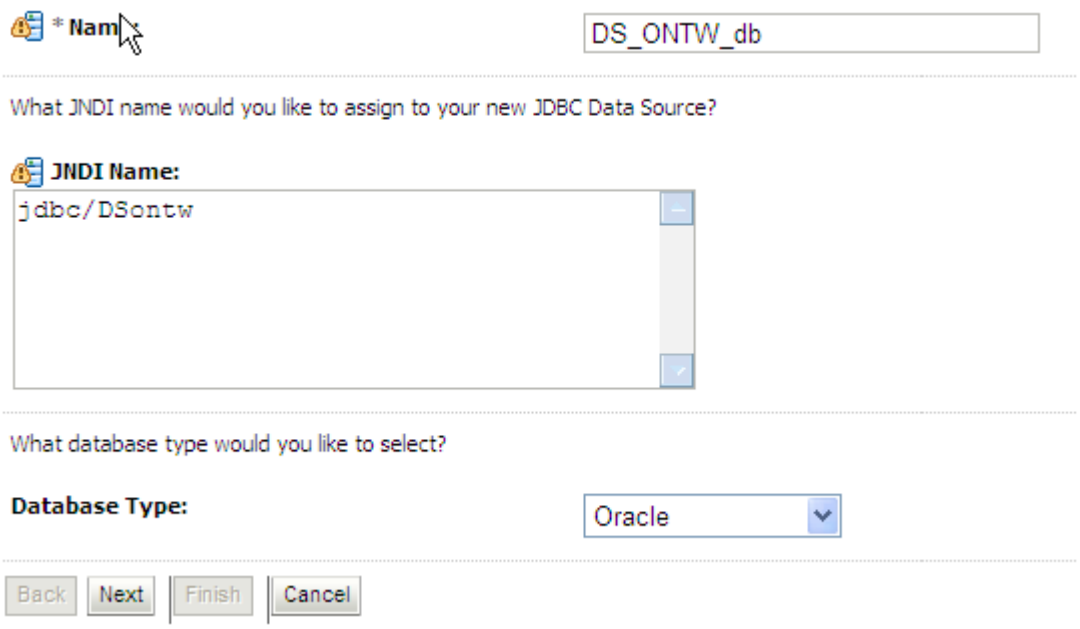
For deploying the web services application a data source is needed. To create one navigate in the Domain Structure panel on the left to the data sources option. Choose 'Lock & Edit' so you are able to create a new data source.

Press new and choose for a Generic data source in the options below:

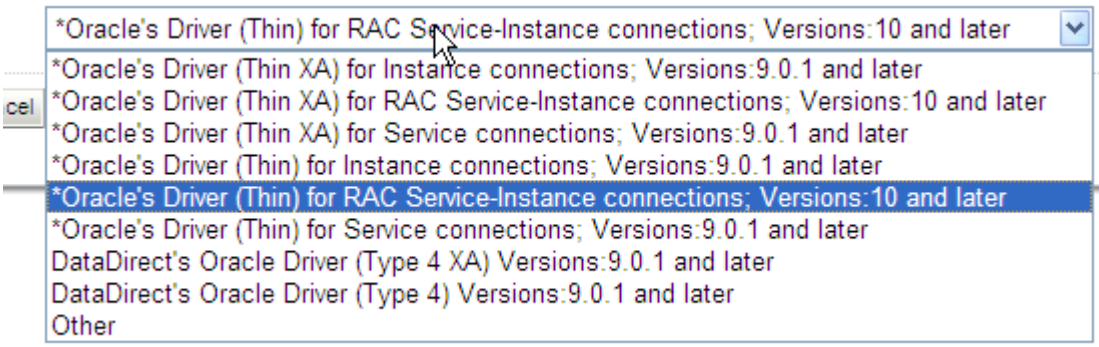


Choose a name for the data source which reflects the connection you are going to use. Reference the database name for example. Specify a JNDI name which must later be specified in a properties file in the web services application. As database type take Oracle.

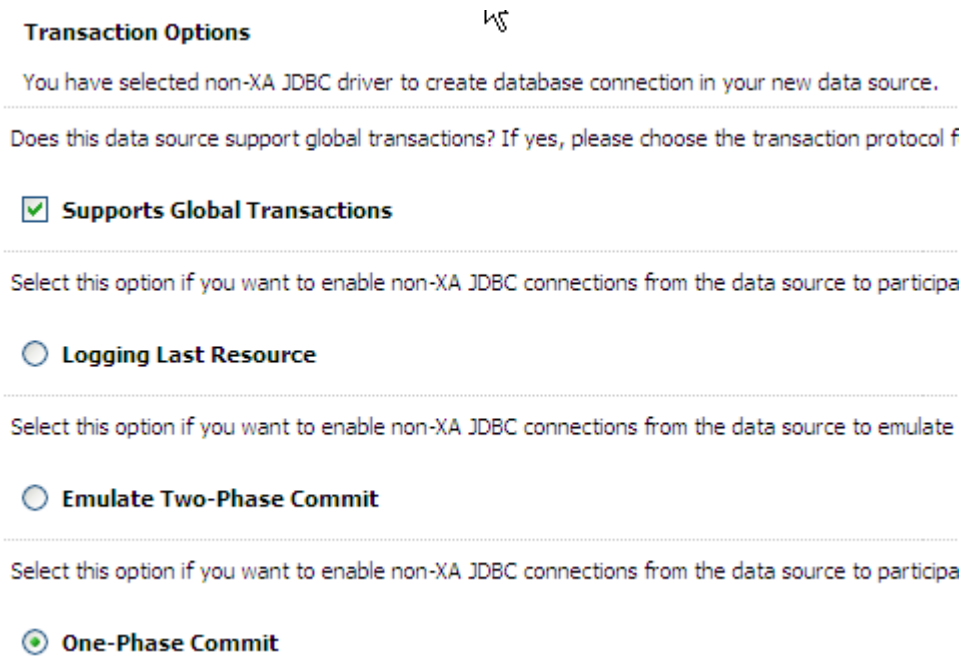
An example:



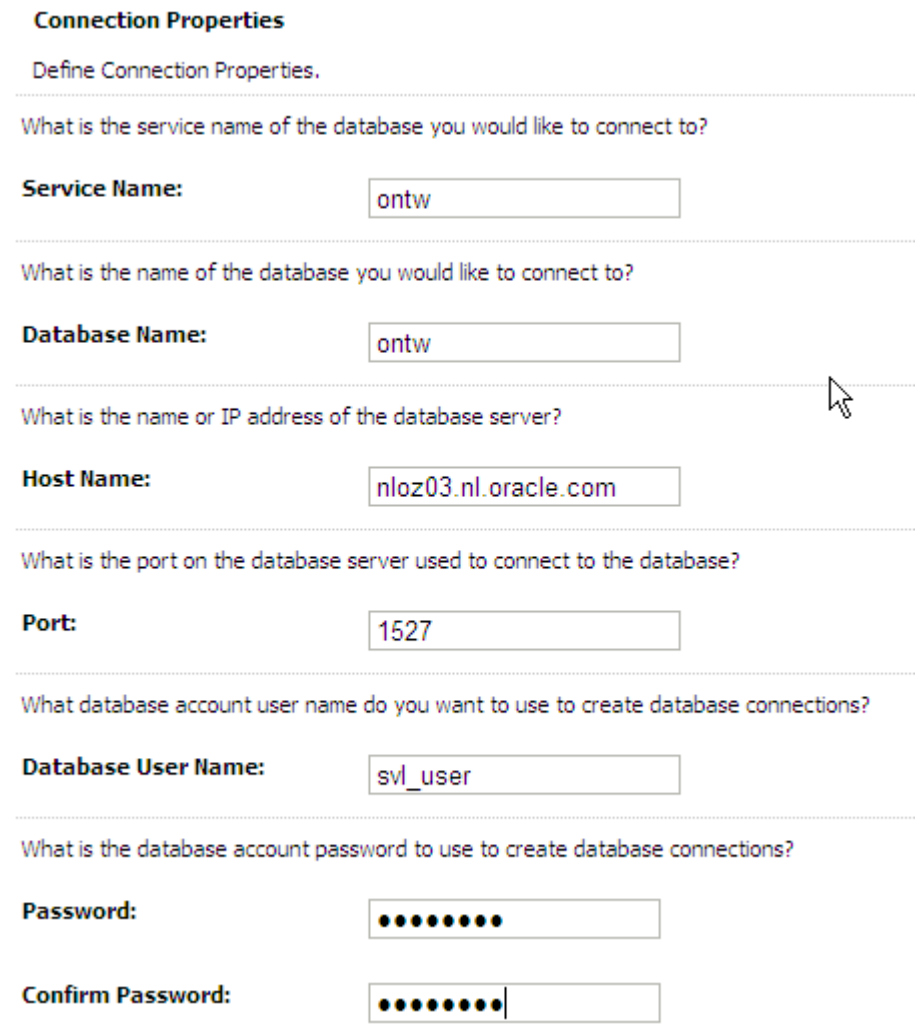
Specify a thin non XA database driver. If you are not sure whether you will use RAC or not do use the RAC driver.



Choose the following options (so Supports Global Transactions and One-Phase Commit).

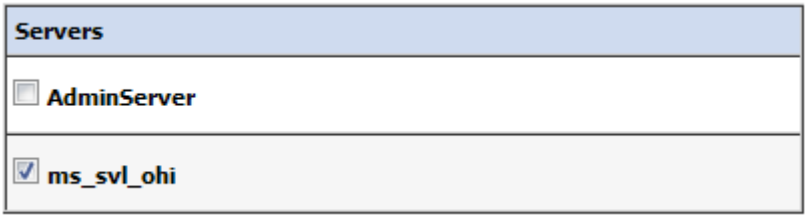


Next specify the connection details like the example on the page below. Be sure to use values which are valid for your environment.



On the next page the result of your answers will be shown. You can test the connection with the data shown.

When going to the next page you can select the targets where the data source should be deployed to. In the example below only one of the Managed Servers in the shown cluster will be used for deploying the data source to.



Press Activate Changes to conclude your configuration.

Afterwards, go back to your data source and open the connection pool tab. Navigate to the 'Advanced' part and verify the setting of the check box 'Wrap Data Types' (present since WLS 10.3.4, not yet in 10.3.3). It should be unchecked! If not, change it to unchecked as shown below. This prevents problems when passing larger objects (CLOB's, etc.).

3.2.5 Deploy the web services application on a single Managed Server

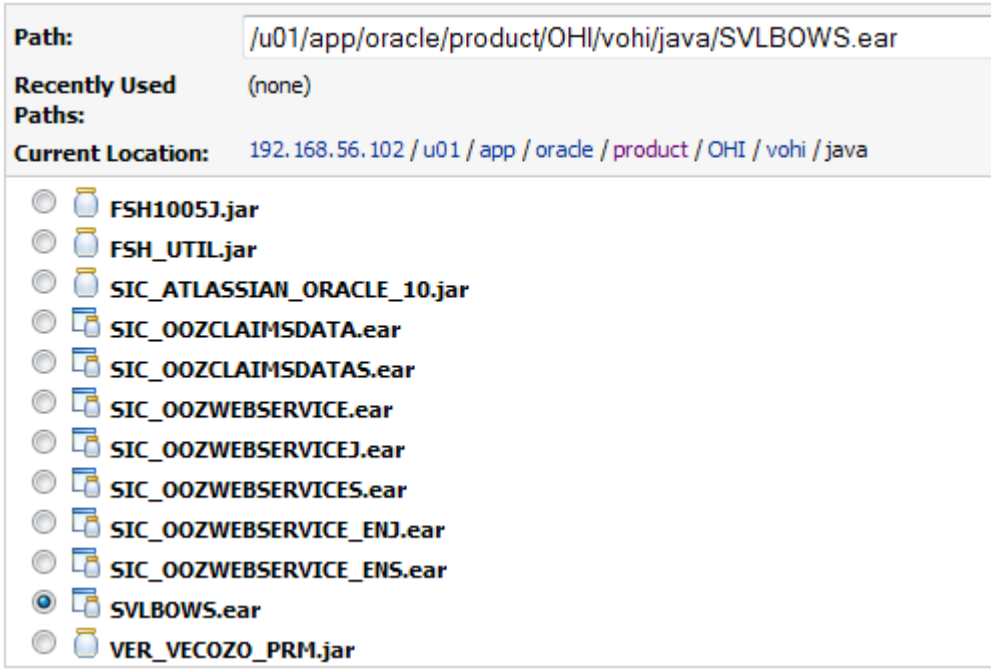
Now you are ready to deploy the web services application. Make sure you have the ear file available at a location on the server where the application server is running. This is the application server that is running the Administration Console.

The ear file is typically name SVLBOWS.ear and is delivered in the $OZG_BASE/java folder of the application server where you installed the OHI Back Office release on for the environment you now want to deploy the web services (be sure you use the correct version of this file, so not from a different OHI environment with a different OHI release).

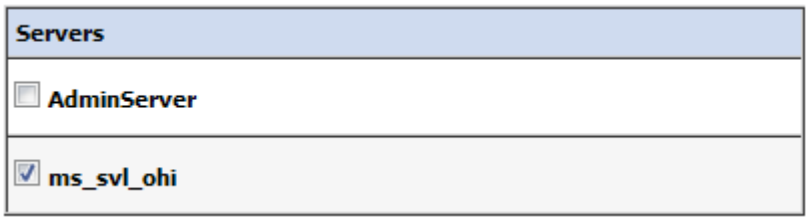
Select the Deployments branch. This will show the already deployed applications (this is a list of more than 20 applications and modules which are mainly deployed on the AdminServer; you can shorten the list by choosing the 'Customize this table' option and choose to exclude the libraries). Choose 'Lock & Edit' which will enable the 'Install' button.

Press the button and use the navigation possibilities to locate the .ear file so you can select it.

An example is shown below:



Press Next and choose for the option 'Install this deployment as an application'. In the next screen you should choose on which target to deploy the application. In the example below only Managed Server ms_svl_ohi is chosen.



Next give the deployment a name, choose the security model you want to apply (DD only is the default; please use 'Custom Roles and Policies' when you like to implement a security policy in a later stage, for limiting access to certain operations), choose the source accessibility (the 'Copy this application….' option is recommended so you can remove the .ear file afterwards from the location you might have placed it temporarily).

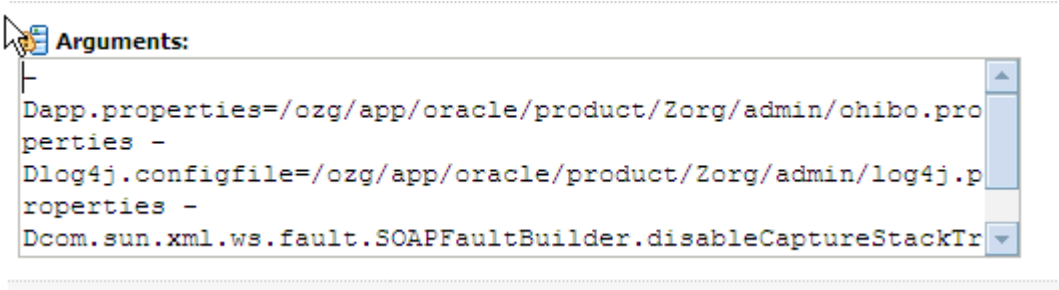
On the final page choose whether you want to go directly to the configuration screen or not.

Beware that 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 and pressing Start → Servicing all requests the State will change in 'Active' (this under the assumption 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. So these do not have to be repeated when you update the deployment or delete and install it again.

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



So like:
-Dapp.properties=/ozg/app/oracle/product/Zorg/admin/ohibo.properties

This example uses a properties file which is located in the $OZG_ADMIN folder of your OHI Back Office application server environment but you can use different locations of course (for example the $OZG_BASE folder).

In the next chapter the contents of this file is discussed.

- ✓ Next add (on the same line in the Arguments field!) the location of a log4j configuration log file to specify a file which contains settings how logging should be configured.
  So extend the Arguments field with something like:
  -Dlog4j.configfile=/ozg/app/oracle/product/Zorg/admin/log4j.properties

  Be sure to place a space between the 2 settings.

  For more information regarding the contents of this file please see the next chapter.

- ✓ To disable stack traces in soap faults add a third setting (on the same line again!) like:
  -Dcom.sun.xml.ws.fault.SOAPFaultBuilder.disableCaptureStackTrace=false

After this be sure to (re)start the Managed Server. Check in the <Mserver>.out file in the logs folder of your Managed Server whether the command line contains the arguments as specified above.

If the log4j properties cannot be interpreted messages as below will show up:

```
log4j:WARN No appenders could be found for logger
(com.oracle.insurance.ohibo.webservices.ohipolicy.OhiPolicyPortTypeImpl).
log4j:WARN Please initialize the log4j system properly.
```

If the app properties cannot be read this will be shown in the log file as specified in the log4j configuration file.

ATTENTION:

If you want to start up a single Managed Server not using the Node Manager typically the script startManagedWebLogic.sh in $DOMAIN_HOME/bin might be used (where $DOMAIN_HOME is assumed to reference your WLS domain). In order to have the correct settings for the server enabled you need to set the JAVA_OPTIONS environment variable before starting the managed server.

An example when the current location is the bin folder of the WLS domain and your managed server is named ms_svl_ohi:

```
export JAVA_OPTIONS="-
Dapp.properties=/u01/app/oracle/product/OHI/vohi/ohibo.properties -
```

```
Dlog4j.configfile=/u01/app/oracle/product/OHI/vohi/svl_log4j.properties -
Dcom.sun.xml.ws.fault.SOAPFaultBuilder.disableCaptureStackTrace=false"
```

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

```
./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.

3.2.6 Deploy the web services application on 2 Managed Servers

As described in the previous paragraph you can deploy the web service application to more than one target if you choose so. When you choose for example to target on Managed Servers MS1 and MS2 the web service will become available on 2 separate end points, where the URL only differs in the port number as used by each Managed Server.

This is however not a very likely situation, unless you want to deploy the application on different servers for different purposes (you are able to specify on a per server basis different settings files for the application, as indicated by the app.properties and log4j.configfile filenames).

3.2.7 Deploy the web services application on 'the whole cluster'

It is an option to deploy the application on all the Managed Servers of a cluster. This eases extending the capacity by adding Managed Servers to the cluster. However, if you want to have a transparent deployment you need a kind of load balancer to hide the ports of the different Managed Servers in the cluster.

There are numerous options for this and it depends heavily on what your organisation prefers how to solve this. When you are going to use a load balanced environment with a number of Managed Servers in a cluster servicing the same application be sure you configure the Managed Servers correct. For ease of use it is wise to redirect logging to a central file disregarding which Managed Server logs messages. In the standard Apache log4j information more information can be found about this topic.

3.2.8 Deploy the web services application multiple times (for different environments)

When you are using several environments (for example an OTAP street) you may have the need to deploy the web service for each environment. In that situation make sure you have as much Managed Servers as needed and target different data sources for the different environments to the different Managed Servers.

For each environment deploy the corresponding version of the .ear file for that environment to the Managed Server for that environment and be sure to specify the correct 'Arguments' value for starting the Managed Server.

3.2.9 Removing a domain

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

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

Next perform the following actions:

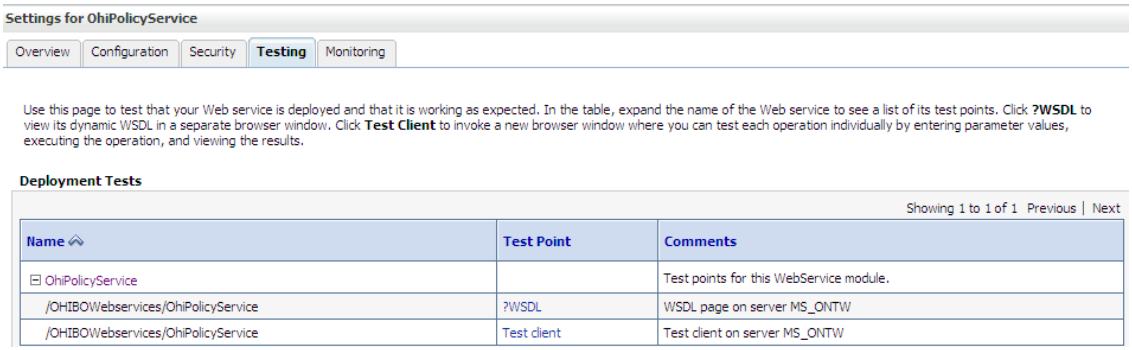
- ✓ Completely remove your domain directory including all contents.
- ✓ Remove any reference in start and stop scripts to this domain.
- ✓ Remove the domain from the <WebLogic home>\common\nodemanager\nodemanager.domains.
- ✓ Remove the domain from the domain-registry.xml file which is located in the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

For more information please use the standard WebLogic documentation.

3.2.10 Publishing the deployed services

After you have deployed the web services perform a small test using the test URL as provided. Typically the isAlive request operation can be used for this.

And next publish the URL's to the involved people within your organisation (typically the WSDL URL). The URL's for this can be retrieved using the Test tab for the deployed services. An example is shown below.



Another option is to open the web application module which is one on one present for each web service and navigate to the 'Test' tab. This contains an URL which redirects you to a simple HTML page that publishes the WSDL and some other information per web service.

The URL per web service for this page is typically in the format <servername>:<port>/<web application name>.

If you want to retrieve the WSDL's of the services while they are not deployed you might retrieve them from the .ear in which the web service implementations are delivered. After the installation of a new OHI release you can find them in $OZG_BASE/java/SVLBOWS.ear.

When you open the file with a utility that can read archives like an .ear file you will find in the top level folder a .war file per web service. This .war file contains a WEB-INF subfolder with a folder wsdl in it. In this folder you can find the WSDL file and accompanying .xsd files.

3.3 Implement security to prevent unauthorized access

The provider web services provide an additional access channel to retrieve and change data within the OHI Back Office application.

It is of vital importance to implement an appropriate security solution to prevent any unauthorized access to the web services. To prevent exposing privacy sensitive data or unauthorized and illegal changes to data access should be limited to trusted systems and/or interfaces. Otherwise people might be tempted to try to misuse this functionality.

Please consult the 'Oracle Health Insurance Security Aspects' guide for more information about this and other security aspects.

4 Configuration files for provider web services

In the previous chapter two files were referenced in the web service application server deployment description for which more information is provided below.

4.1 Back Office properties file

The Back Office properties file specifies a number of properties with values that can be generic for all web services or specific 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.

For each web service there are at least 2 properties you need to set:

1. A value for the application user which should be used to act upon for executing the web service operations (this influences for example under which user identity the changes are made or which language is used for messages). This is the 'officer username' property.

2. A value for the datasource within WebLogic which should be used for connecting to the database. This can (and will mostly) be the same datasource 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. Typically the jndi name is used for this.
   The JNDI name must be identical to the JNDI name as given to the datasource when creating it. 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.

4.1.1 2011.02 properties

In the first Service Layer release, 2011.02, only the four properties as shown below apply:

```
gebruiker.username=MANAGER
policy.datasource.jndiname=jdbc//DSontw
preauthorization.callcontext.usercontext.officerusername=MANAGER
preauthorization.datasource.jndiname=jdbc//DSontw
```

The first 2 properties apply to the OHI policy web service and the last 2 properties apply to the OHI preauthorization web service. For each web service the first property specifies the officer username of an application user which must be valid.

Per web service the datasource JNDI name is specified th. In the example above the same datasource is specified for the Policy web service and the PreAuthorization web service.

4.1.2 2011.02.2 properties

As of release **2011.02.2** the first property as shown above has been changed in name and should be adapted as shown in the properties file:

```
policy.callcontext.usercontext.officerusername=MANAGER
policy.datasource.jndiname=jdbc//DSontw

preauthorization.callcontext.usercontext.officerusername=MANAGER
```

```
preauthorization.datasource.jndiname=jdbc//Dsontw
```

Next to that 2 new services are delivered as of release **2011.02.2** which require additional properties when used (specified values are example values):

```
relation.datasource.jndiname=jdbc//OHIDatabaseDS
relation.callcontext.usercontext.officerusername=MANAGER


collectiveagreement.datasource.jndiname=jdbc//OHIDatabaseDS
collectiveagreement.callcontext.usercontext.officerusername=MANAGER
```

4.1.3 2011.03 properties

In release **2011.03** some more properties are added for an additional service:

```
policy.callcontext.usercontext.officerusername=MANAGER
policy.datasource.jndiname=jdbc//DSontw

preauthorization.callcontext.usercontext.officerusername=MANAGER
preauthorization.datasource.jndiname=jdbc//DSontw

relation.datasource.jndiname=jdbc//OHIDatabaseDS
relation.callcontext.usercontext.officerusername=MANAGER

collectiveagreement.datasource.jndiname=jdbc//OHIDatabaseDS
collectiveagreement.callcontext.usercontext.officerusername=MANAGER

realtimeclaimfile.callcontext.usercontext.officerusername=MANAGER
realtimeclaimfile.datasource.jndiname=jdbc//OHIDatabaseDS
realtimeclaimfile.claimorigin=EI
realtimeclaimfile.reimbursementrequestorigin=EI

claim.callcontext.usercontext.officerusername=MANAGER
claim.datasource.jndiname=jdbc//OHIDatabaseDS
```

Beware that the claimorigin and reimbursementrequestorigin properties, as named in the last 2 lines of the realtimeclaimfile settings above, must contain a value which is present in table GEB_HERKOMST_DECLARATIES.

4.1.4 2012.01 properties

In release 2012.01 the 2 specific additional properties for the realtimeclaimfile web service have been moved. They are now implemented as Back Office parameters, a new way of defining settings for the OHI Back Office application. The reason for the move is the usage of these settings also when the plsql implementation of the service is used. Now the web service and the plsql service implementation can both use these settings. The functional specification of theme M-2623 contains a little more detail about these settings.

Another change is the rename of the `officerusername` property to `user.username`. This is enforced by internal standards for having more consistency in naming elements.

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

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

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

collectiveagreement.callcontext.usercontext.user.username=MANAGER
collectiveagreement.datasource.jndiname=jdbc//DSprod

realtimeclaimfile.callcontext.usercontext.user.username=MANAGER
```

```
realtimeclaimfile.datasource.jndiname=jdbc//DSprod

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

4.1.5 2012.01.03 properties

In release 2012.01.03 a new web service was added for payment schemes. This is the complete list.

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

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

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

collectiveagreement.callcontext.usercontext.user.username=MANAGER
collectiveagreement.datasource.jndiname=jdbc//DSprod

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

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

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

4.1.6 2012.02.02 properties

In release 2012.02.02 a new web service was added for procedure authorizations. This is the complete list.

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

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

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

collectiveagreement.callcontext.usercontext.user.username=MANAGER
collectiveagreement.datasource.jndiname=jdbc//DSprod

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

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

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

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

4.1.7 10.13.1.4 properties

In release 10.13.1.4 a couple of new web services were added. This is the complete list.

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

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

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

collectiveagreement.callcontext.usercontext.user.username=MANAGER
collectiveagreement.datasource.jndiname=jdbc//DSprod

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

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

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

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

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

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

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

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

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

4.1.8 10.13.3 properties

In release 10.13.3 a couple of new web services were added. This is the complete list.

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

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

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

collectiveagreement.datasource.jndiname=jdbc//DSprod
collectiveagreement.callcontext.usercontext.user.username=MANAGER

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

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

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

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

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

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

payment.datasource.jndiname=jdbc//DSprod
```

```
payment.callcontext.usercontext.user.username=MANAGER

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

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

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

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

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

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

4.1.9 10.13.3.3 properties

In patchset 10.13.3.3 some additional services were added. The new services are 'provider' and 'pxrelation'. The last one is a new write service according to new standards for implementing them, based on sending (part of) a picture how the end situation after the write should be. Picture is abbreviated to 'px', to indicate this new type of services. The 'collectiveagreement' web service has been replaced by groupcontract, so can be removed.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

4.1.10 10.14.1.0 properties

In major release 10.14.1.0 an existing service to create provider contracts has been rewritten so it also supports changes according to the new standards for write services. The service 'providercontract has been renamed to 'pxprovidercontract'.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

provider.datasource.jndiname=jdbc//DSprod
```

```
provider.callcontext.usercontext.user.username=MANAGER

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

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

4.2 LOG4j configuration file

For the configuration of what messages should be logged by the application server, where the web services are deployed on, the log4j standard is used. Log4j is one of the widely used standard logging frameworks for Java coding environments.

For more information please use the links below.

For generic log4j information:

http://logging.apache.org/log4j/

And more specific the documentation:

http://logging.apache.org/log4j/1.2/manual.html

And some Wiki information:

http://en.wikipedia.org/wiki/Log4j

In the OHI Back Office implementation the following log levels can be used for logging messages:

Loglevel FATAL
Loglevel ERROR
Loglevel WARN
Loglevel INFO
Loglevel DEBUG

The file contents below can be used as an example for a log4j.properties file. The web services specific class prefix is contained as example.

```
#
# Console appender
#
log4j.appender.C=org.apache.log4j.ConsoleAppender
log4j.appender.C.target=System.out
log4j.appender.C.layout=org.apache.log4j.PatternLayout
log4j.appender.C.layout.ConversionPattern=%-5p %d [%t] %c: %m%n

#
# File appender; this should be a RollingFileAppender to prevent it
# from becoming to large because only in that case MaxFileSize etc.
# will be interpreted.
#
log4j.appender.F=org.apache.log4j.RollingFileAppender
log4j.appender.F.File=/u01/app/oracle/product/OHI/prod/svl_Webservices.log
log4j.appender.F.MaxFileSize=50000KB
log4j.appender.F.MaxBackupIndex=1
log4j.appender.F.layout=org.apache.log4j.PatternLayout
```

```
log4j.appender.F.layout.ConversionPattern=%-5p %d [%t] %c: %m%n
#
# Log settings:
#
log4j.rootCategory=INFO, F
log4j.logger.com.oracle.insurance.ohibo=INFO
```

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:

- ✓ Beware: This step is necessary up to release 2011.03. From release 2012.01 and onwards this step is automated as long as you have once granted the schema owner these rights.

  Make sure you run the step again to actualize the grants to the schema owners you created for connecting the connection pool(s) to the database. Connect to the OHI schema owner and run a command like below (typically in sqlplus):

  ```
  set serveroutput on
  execute alg_security_pck.svl_grants(pi_grantee =>
  'SVL_USER')
  ```

- ✓ Check your web service properties file (typically ohibo.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 have an already 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.

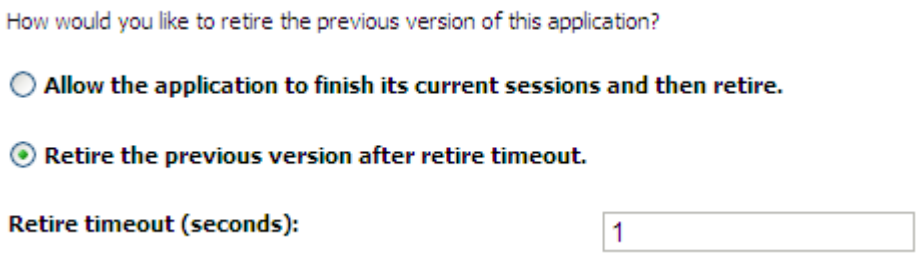
- ✓ When the previous version was 'unversioned' (no version is shown between brackets behind the deployment name) please proceed with these instructions, otherwise proceed to the next bullet.

  You can only press Finish as next option. If the new version is also still 'unversioned' (a 'versioned' delivery starts with release 2011.03) the Finish option will succeed. If the new version is 'versioned' you will get an exception that the previously deployed application was without version. In that situation you are required to stop (force shutdown) the deployed application and subsequently delete the deployed application and deploy ('install' deployment) the new version of the application as described in the paragraph(s) regarding the deployment of the provider web services.
  Skip the next bullet.

- ✓ When the previous version was 'versioned' (a version is shown between brackets behind the deployment name) please proceed with these

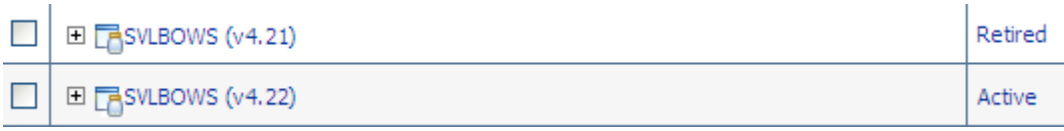
instructions, otherwise proceed to the next bullet.

You now have two options for 'retiring' the previous version. Because normally the Back Office application is not available you can retire the previous version 'immediately', meaning using a timeout of 1 second:



Press Finish to implement this.

- ✓ Choose 'Activate Changes'.
- ✓ If you have deployed a versioned version of the application refresh the screen a few seconds after having activated the changes. The previous version should now shown a Retired state as in the example below:



- ✓ Inform the community which uses the web services of the availability and publish the WSDL address to them, especially when it has changed.

If the update of the deployment does not succeed delete the deployed web service (stop first with the force option) and deploy it again completely (using the 'install' option for deployments). It might be (depending on the changes) that a repetition of the Deployment delete/install is one more time needed when the install results in errors. In case of continuation of errors restart of the Managed Server(s) is necessary 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 database environment and deployed version are correct, meaning that they 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. These web services are expected to become available starting with release **2012.01.0.0000**. So instructions do not apply to earlier releases.

The web service consumers are implemented by means of the "Web Services Call-Out" feature as provided for the database. This enables the publishing of a WSDL into a database plsql package. A java based web services client proxy is created as well as a plsql 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 consuming web services code as implemented within OHI Back Office execute 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.

It is strongly advised and in fact in most situations required to setup an application server 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.

Typically this kind of functionality is implemented using a service bus.

For most web service consumers this is a requirement in order to map the OHI specific WSDL to an outside world WSDL. This kind of transformations can be easily implemented through service bus functionality.

Also other (security!) requirements can/should (security!) be implemented in or provided by the intermediate middleware including the service bus.

It is outside the scope of this document to describe the middleware setup but these kind of requirements and functionalities are typically 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. And the related object types and plsql wrapper package need to be created.

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