ORACLE
INSURANCE

Oracle Health Insurance Back Office

Oracle Health Insurance Web Services

Installation Guide for WLS

version 1.13

Part number: E91383-01

December 11th, 2017

CHANGE HISTORY

Release	Version	Changes
10.12.2.0	1.1	• Added Appendix B for functional testing.
10.13.1.0	1.2	• Corrected some small typing errors (especially the minus sign in front of the –Dlog4j... setting which is not correct and resulted in errors when copying the line to a WebLogic configuration).
10.13.1.1	1.3	• Adjusted references for FRS11G1 to FRS11G2.
10.13.2.0	1.4	• Added additional information regarding English version
10.13.2.0	1.5	• Added an English example message
10.13.3.0	1.6	• Added small paragraph in Administration chapter about updating an existing deployment.
10.15.1.0.0	1.7	• Removed OOZCLAIMSDATA
10.15.3.0.0	1.8	• Nothing changed
10.16.1.0.0	1.9	• Nothing changed
10.16.2.0.0	1.10	• Adapted for FMW 12.2.1.1.0
10.16.2.2.2	1.11	• Corrected accidentally removed text in the SIC_OOZWEBSERVICEJ deployment paragraph. • Added instructions for setting up a separate database account for the connection pool
10.17.1.0.0	1.12	• Minor changes to reference FMW 12.2.1.2.0.
10.17.2.0.0	1.13	• Replaced Log4J configuration by Java Util Logging (JUL) configuration.

RELATED DOCUMENTS

Doc[1]: Oracle Health Insurance Back Office Service Layer Installation & Configuration Manual (CTA13651)

Contents

.1 Introduction

This document describes the installation and configuration of the OHI Connect to Back Office web service released by OHI Back Office.

...1.1. Purpose

Describes the installation and configuration of the OHI Connect to Back Office web service in an Oracle Weblogic Server 12c environment (part of Oracle Fusion Middleware 12c).

Please be aware that this software is deprecated and will be de-supported in a future OHI release (currently desupport is planned for release 10.18.2.0.0).

...1.2. Audience

This document is intended for administrators of Oracle WebLogic Server. Required knowledge:

- Working knowledge of Oracle WebLogic Server 12c version 12.2.1.2.0 or higher.
- The creation of Domains and Managed Servers in WebLogic Server.
- The creation of JDBC Data Sources in WebLogic Server
- The creation of JMS queues and connection factories in WebLogic Server
- The deployment of EAR files in WebLogic Server.

...1.3. Document structure

This document is organized as follows:

- Overview of the OHI Back Office web services
  Describes the SOAP/JMS (asynchronous) and SOAP/HTTP (synchronous) web services on a conceptual level.
- Installation preparation
  Applies to both SOAP/JMS and SOAP/HTTP implementations, unless stated otherwise.
- Installation of the asynchronous version (SOAP/JMS)
  Describes the minimum procedure for configuring the queues needed by SIC_OOZWEBSERVICEJ, the most common use of this service.
- Administration

Finally, appendix A describes the deployment of the synchronous version of SIC_OOZWEBSERVICE.

...1.4. Release

OHI offers the following web services for OHI Connect to Back Office:

- SIC_OOZWebservice (synchronous and asynchronous)
  This offers multiple service operations to update / retrieve data in OHI Back Office.

When invoked, the web service connects to the OHI Back Office database to perform the required action and returns a response message to the calling application.
The action either retrieves data from the OHI Back Office database or puts data into the OHI Back Office database. In both cases the OHI Back Office database must be online in order to complete the required action.

In many cases the calling application does not depend on an immediate response from the web service. In those cases, an asynchronous web service can be used.

The SIC_OOZWebservice has both a synchronous (SOAP/HTTP) and an asynchronous interface (SOAP/JMS). Apart from the interface, both variants use the same code.

All OHI web services are released as EAR files. Whether a web service is asynchronous or synchronous can be derived from the file name:

- asynchronous web service: <WEBSERVICE NAME>J.ear
- synchronous web service: <WEBSERVICE NAME>S.ear

OHI BackOffice supplies SIC_OOZWEBSERVICES.ear and SIC_OOZWEBSERVICEJ.ear as two variants of the general purpose web services.

...1.4.1. Dutch versus English

As these web services use Dutch XML messages it is important to know that there are also English versions of the OOZWEBSERVICE variants. These can be recognized by having '_EN' just in front of the ...J.ear or ...S.ear, so SIC_OOZWEBSERVICE_ENJ.ear and SIC_OOZWEBSERVICE_ENS.ear.

This documentation will only distinguish between the Dutch and English version when the difference is important. The Dutch names will be used in other cases. If you want the Egnlish version, simply replace the file names for the Dutch versions in the instructions by the file names used for the English version.

...1.5. Software versions

The following software must be installed before OHI Connect to Back Office web services can be installed:

- OHI Back Office (including database software)
- Oracle WebLogic Server

Note that this installation assumes that the OHI web services are installed on the same node as the OHI Back Office application. This is not a requirement.

...1.5.1. OHI Back Office

The interface works with OHI Back Office release 2006.02.4.0000 and above.

The version of OHI Connect to Back Office must match the version of OHI Back Office. As part of your OHI Back Office patching routine, you should always deploy the version of OHI Connect to Back Office that was most recently delivered with a patch release for OHI Back Office.

...1.5.2. Oracle WebLogic Server

Starting with release 10.16.2.0.0 of OHI Back Office, OHI Connect to Back Office must be deployed on Oracle WebLogic Server (WLS) 12c. This manual assumes WLS version 12.2.1.2 is configured and used with 'oratab identifier' FRS12212.

A WebLogic Server environment (software home) must be prepared for deploying the Connect to Back Office application.

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 Connect to Back Office 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.

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 Connect to Back Office 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:

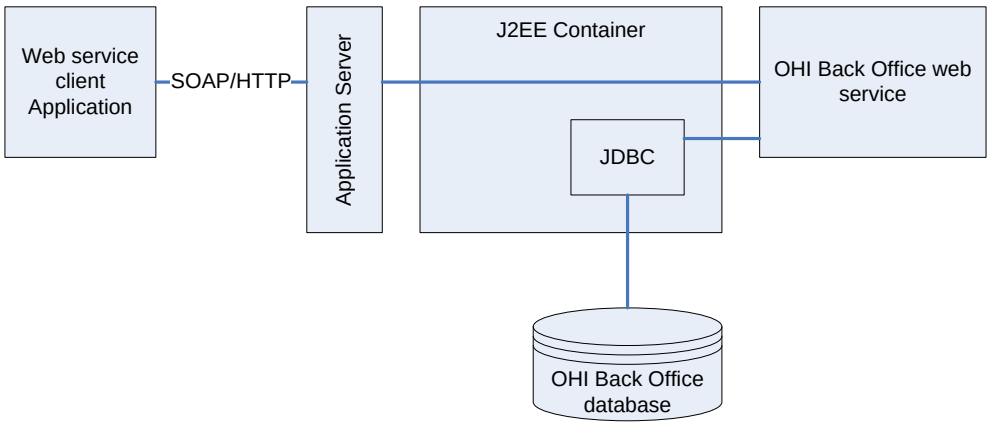
    - Weblogic Advanced Web Services for JAX-WS Extension - 12.2.1 [oracle_common]

    - Weblogic JAX-WS SOAP/JMS Extension - 12.2.1 [oracle_common]

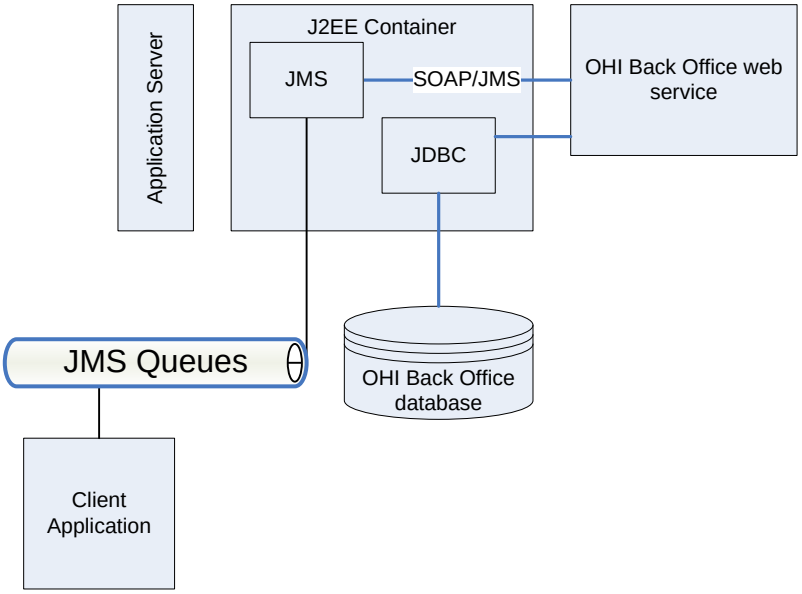
.2 Overview of the OHI Back Office web services

As indicated before, there are synchronous and asynchronous OHI Back Office web services.

The diagram below shows the components for a synchronous (SOAP/HTTP) web service:



In the case of an asynchronous web service, the request is stored in a message queue. The message remains in the queue until it can be processed by the OHI Back Office web service. The web service creates a response which is stored in a different queue until it is processed by the calling application:



The advantage of asynchronous web services is that the server application can be offline when the request is created and that the client application can be offline when the request is processed. The price for this is that all communication between client and server is temporarily stored in message queues.

...2.1. SOAP/JMS web service

The asynchronous web service (SOAP/JMS) is installed within a J2EE container (a J2EE runtime environment within the application server).

The communication with this web service is handled by JMS queues.

The invocation is triggered by a message on the inbound queue, jms/OOZWebserviceQueue.

The payload of this message is an XML document. The XML document contains the data to create PL/SQL calls in OHI Back Office through a JDBC database connection.

If the request is handled successfully, the response is also an XML document which is put in a message on the outbound queue, jms/OOZWebserviceResponseQueue.

In case of an error, a message is created in the error queue, jms/oozErrorQueue.

The standard queue connection factory for accessing these queues is jms/oozQueueConnectionFactory.

...2.1.1. English queue names

When you use the _EN (English) versions of the OOZWebService the names of the request and response queue contain 'C2B' instead of 'ooz'.

...2.2. SOAP/HTTP web service

The SOAP/HTTP web service is a Stateless Session Bean. This receives a SOAP message, processes the request to OHI Back Office and returns the response.

The internals of a SOAP/HTTP service are identical to the SOAP/JMS service.

.3 Installation preparation for SOAP/JMS

The installation preparation contains the following steps:

- Verification of required files
- Database preparation
- WebLogic Server Preparation

...3.1. Verification of required files

...3.1.1. Required Files

Depending on the agreements made, the following files are supplied as part of OHI Back Office release deliveries by Oracle via My Oracle Support patches:

- SIC_OOZWEBSERVICEJ.EAR
  (SOAP/JMS web service containing various services for Connect to Back Office)

...3.2. Database preparation

...3.2.1. Database objects

Before the installation the required database objects in the OHI Back Office must be installed. This task is usually carried out by the OHI Back Office DBA.

...3.2.2. Oracle account to be used

Before the start of the installation, determine how the web service should connect with the OHI Back Office database.

Item	Description
Host	Host of the TNS listener for the OHI Back Office database. Normally this is the database server.
Port	Port number of the TNS listener Typically this is 1521 or 1526
Service	The ORACLE service of the OHI Back Office database
Oracle account	The Oracle account used to access the OHI Back Office database
Password	Password for the Oracle account

These data will be used to create a JDBC data source.

Note: the most obvious scenario would be to use the owner of the database objects as the oracle account, e.g. ozg_owner.

However, for security reasons it is desirable to use an alternative Oracle account to which only the minimum set of required privileges has been granted.

Such an account can be setup using the following guidelines:

1. Create a separate database user account with a sufficient complex password.
2. Make sure no password expiration policy is active for this account to prevent it suddenly becomes inoperational.
3. Implement your organizational standards for this accounts
4. Grant CREATE SESSION to this account.
5. Connect to the OHI table owner and grant EXECUTE privilege on SIC_OBJECT_PCK and SYS_MESSAGE_HANDLING_PCK.

Use this account during the configuration of the connection pool later on in this document.

...3.3. WebLogic Preparation

A certified version of the Weblogic application server must be installed prior to this step.

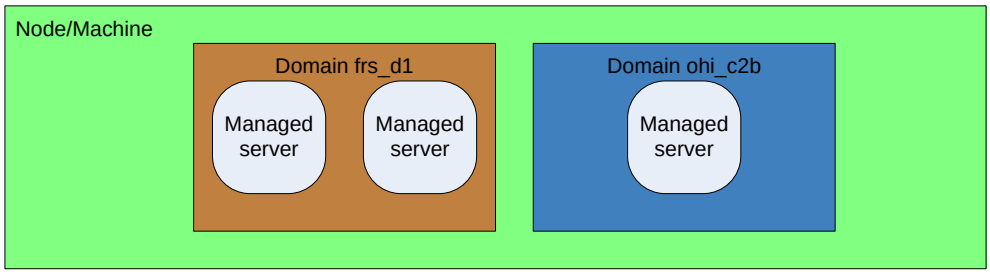
The Weblogic preparation consists of the following actions

- Create domain, managed server, machine
- Create data source

...3.3.1. Create domain, managed server, machine

Instead of adding the web services to the domain you are already using for OHI Back Office, we strongly advise to create a new domain for the Connect to Back Office web services.

We must now create managed server(s) and machine definitions for this new domain. Note in the diagram below how an existing frs_d1 and new ohi_c2b domain coexist:



The steps for creating domain, managed server and machine are documented in paragraph 3.2 of **Doc[1]** but can also be found in the WebLogic documentation.

...3.3.1.1. *Create domain*

Make sure any other services present on the server are running, to avoid port conflicts by being able to examine which ports are not used (you should check this yourself).

Start the configuration wizard for creating a new domain:

- Log in as the UNIX account running WebLogic (e.g. oracle)

- `. ozg_init.env $OZG_ORATAB_FRS12212`
- `$MW_HOME/oracle_common/common/bin/config.sh`

Generate a new domain for supporting web services with its own administration server as per paragraph 3.3.2 "Creating a domain" of **Doc[1]**.

Ensure that you configure an admin server within the new domain and that the port number for the admin server has not been used (we chose 7070).

Start the administration server for this domain (suppose it is named ohi_c2b):

`$MW_HOME/user_projects/domains/ohi_c2b/startWebLogic.sh`

...3.3.1.2. Create managed server, machine etc.

The remainder of the procedure is done through the HTML interface of the administration server.

Follow the steps in chapter 3.3.3 "Creating Managed Server(s)" of **Doc[1]** to create a single managed server in the new domain. Suggested names:

- Domain: ohi_c2b
- Machine: machine1
- Managed Server: ms_ohi_c2b (we used port 7071)

...3.4. Creating a JDBC Data Source

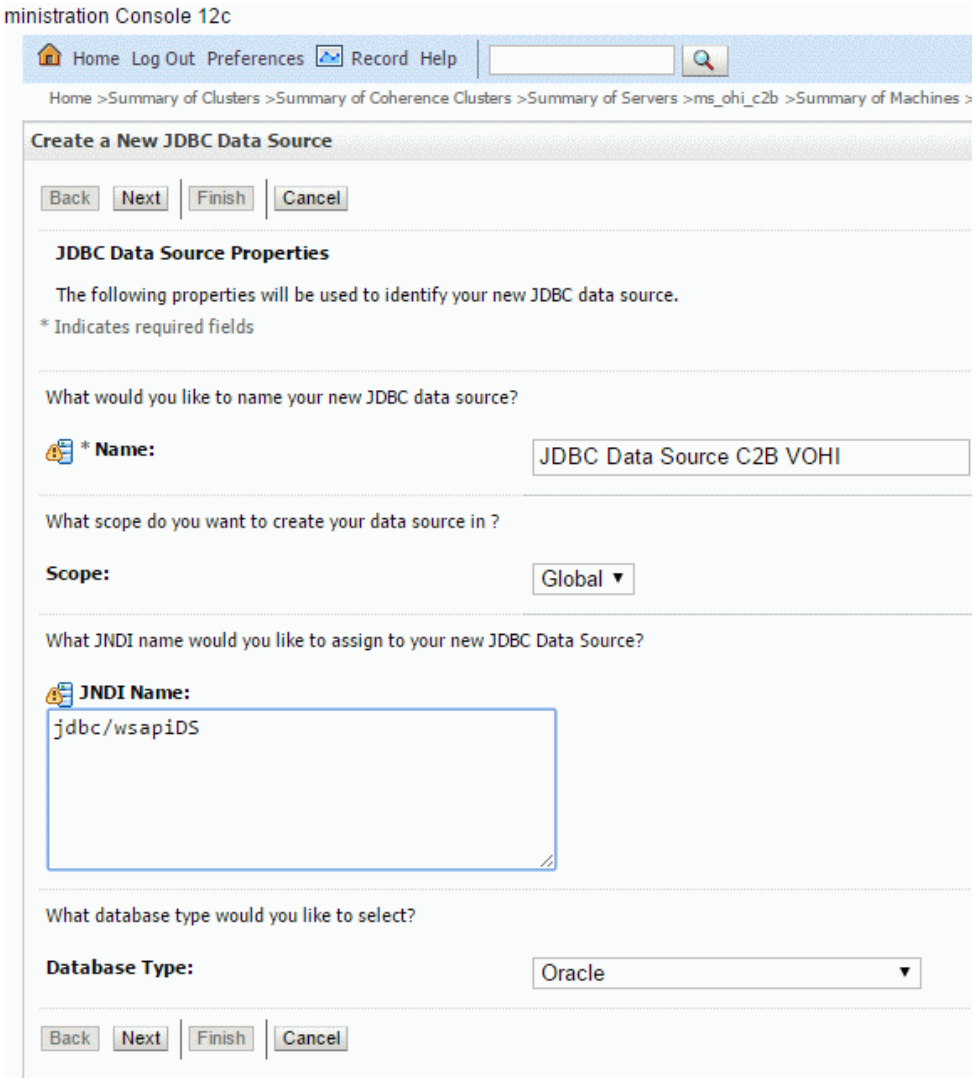
In WebLogic Server, you can configure database connectivity in two steps:

- Define a JDBC data source
- 'Target' the JDBC data source to the domain or to a managed server.

Name	Value
JNDI Name	jdbc/wsapiDS
Database	Oracle
Database Driver	Oracle's Driver (Thin) for Service connections; Versions:Any

...3.4.1.1. Create data source using JNDI name

Create a generic data source.



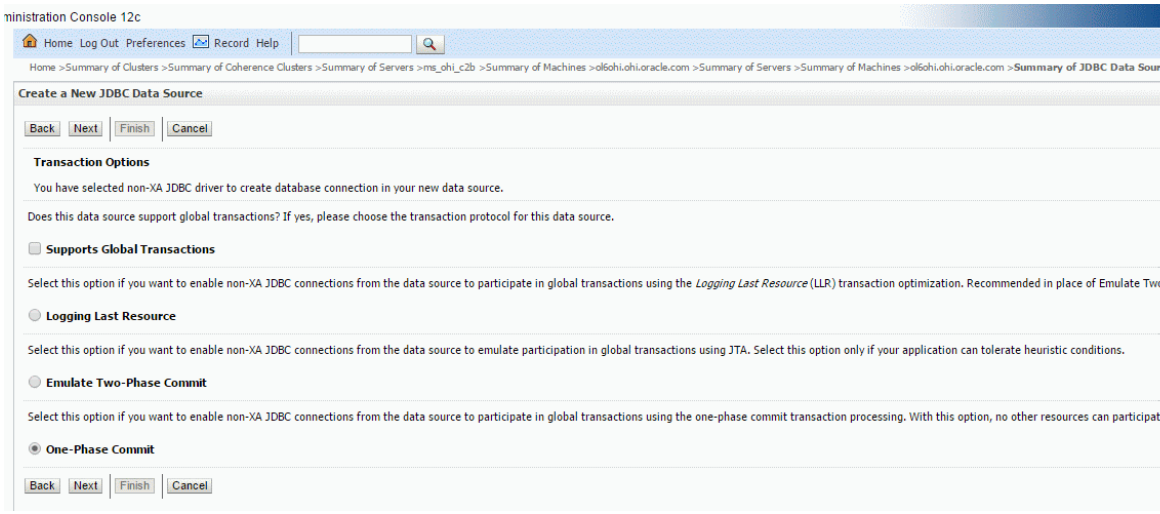
...3.4.1.2. *Select database driver*

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.

...3.4.1.3. *Select the transaction options*

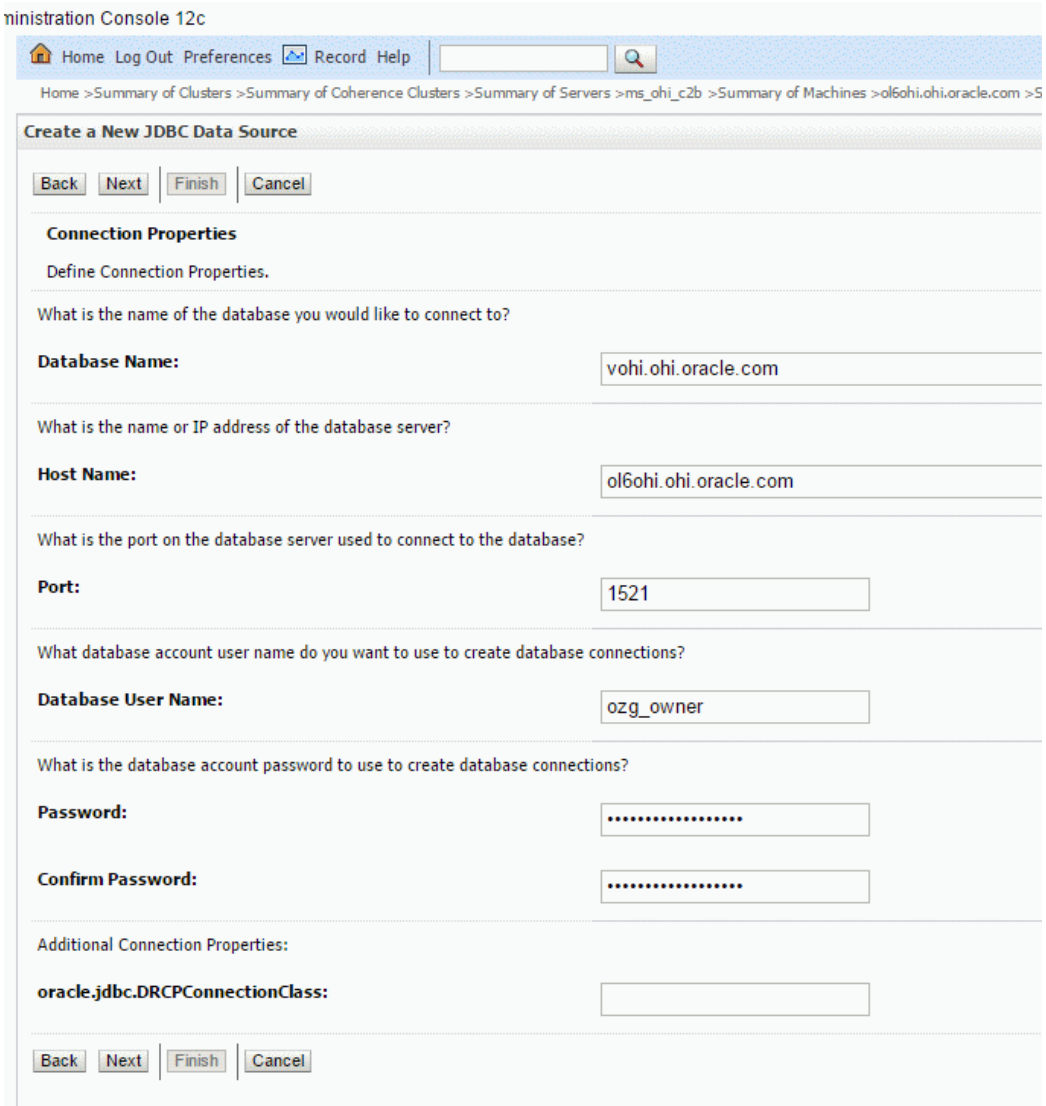
Deselect "Support Global Transactions"

Select "One-Phase Commit"



...3.4.1.4. *Set up connection properties*

Define the Connection Properties using the values described in 3.2.2.



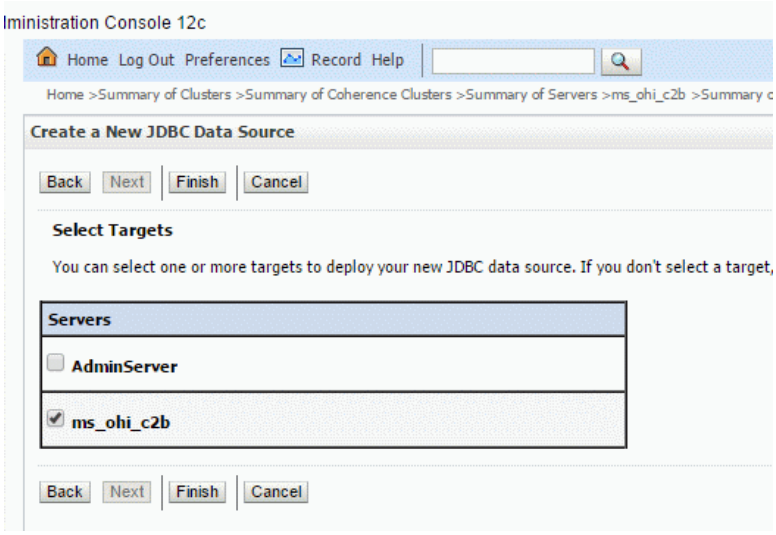
...3.4.1.5. *Test database connection*

Repeat the previous step until the database connection test is successful.

...3.4.1.6. *Target the JDBC data source*

By 'targetting' the JDBC data source you make it available for use.

Select a Managed Server as target.



...3.5. Start the managed server

Start the created Managed Server in the way you prefer it. This may be on the command line or through the (domain specific?) node manager using the console.

.4 Installation of SIC_OOZWEBSERVICEJ

This chapter describes how to install the SIC_OOZWEBSERVICEJ service. The actual deployment of the EAR file is a minor step compared to configuring the JMS queues. For this procedure we tried to simplify the configuration.

The main characteristics of the installation as described here are:

- Non-clustered installation
- Native JMS, using in-memory queues.

The installation procedure contains the following steps:

- Create JMS Server
  This process will implement the queuing mechanism and persist queue data.
- Create JMS Module
  This container will hold the resources (queue connection factory and queues)
- Create queue connection factory
  This is used to access the queues handled by the JMS Server
- Create subdeployment
  This is a mechanism to link queues to the JMS server
- Create JMS queues
- Deploy application (EAR file).

...4.1. Before you start

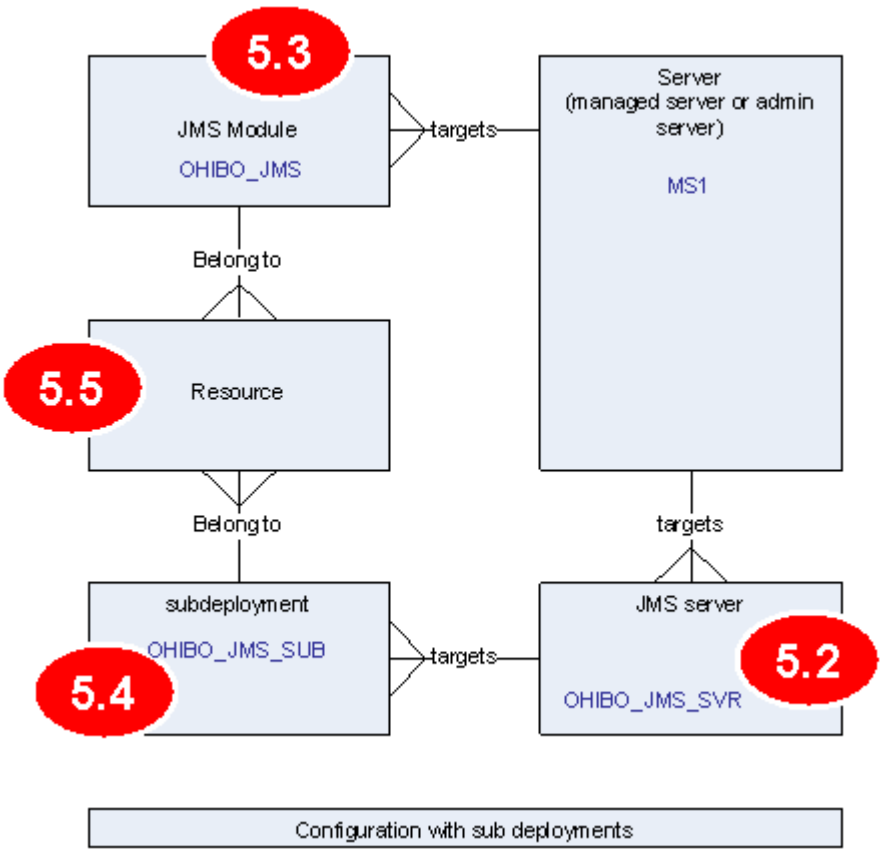
The following JMS queues must be created:

- jms/OOZWebServiceQueue
  (for the English version: jms/C2BWebServiceQueue)
- jms/OOZWebServiceResponseQueue
  (for the English version: jms/C2BWebServiceResponseQueue)
- jms/oozErrorQueue

The queues are accessed through a connection factory called jms/oozQueueConnectionFactory.

In Weblogic, JMS queues and JMS connection factories are called JMS resources. Resources can only be created within the context of a JMS module. The JMS module itself must belong to a (managed) server, while its resources are handled by a JMS server.

The diagram below visualizes how the different components in this configuration procedure relate to each other (shown as an entity relationship diagram). Note that the numbering of the components corresponds with the paragraphs which describe the configuration of that particular component. The names used for the components are indicated in blue:



...4.1.1. Alternative implementations

The installation procedure assumes a memory based, non-clustered implementation of the JMS queues.

Generally speaking, all queues will be (nearly) empty during normal operation. Note however that if the application server is shut down by force or as a result of a crash, the contents of the queues are lost, and new requests can not be queued by the calling application.

To avoid this, you can do the following:

- Use persistent storage for the queue messages (either in the database or in a File Store).
- Create a multi-node, clustered configuration. This allows you to take an application server instance down without interrupting the service.

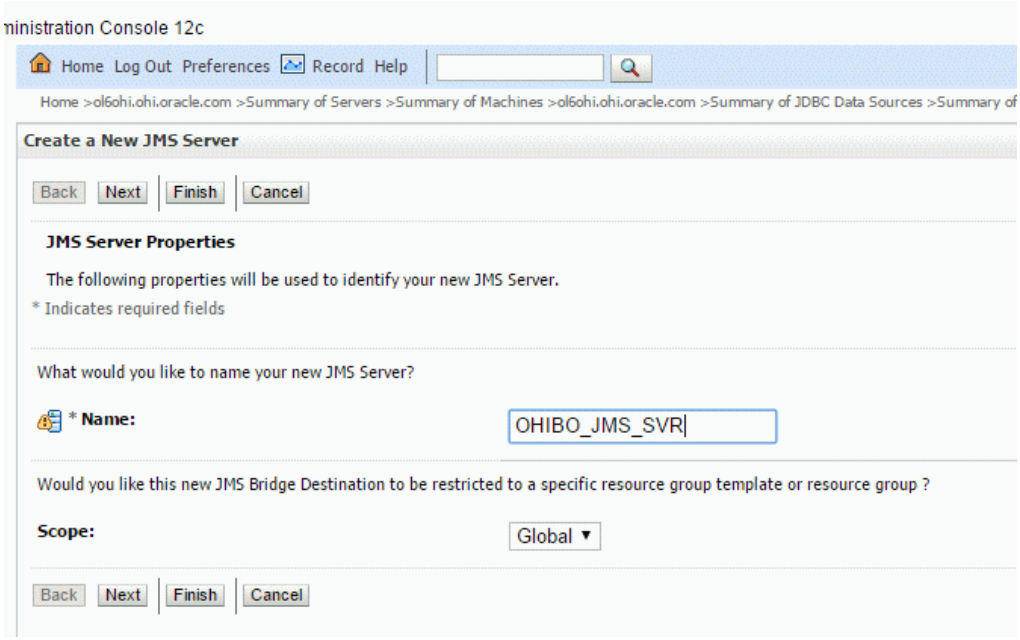
If you choose to take this approach, the following changes apply to the installation procedure:

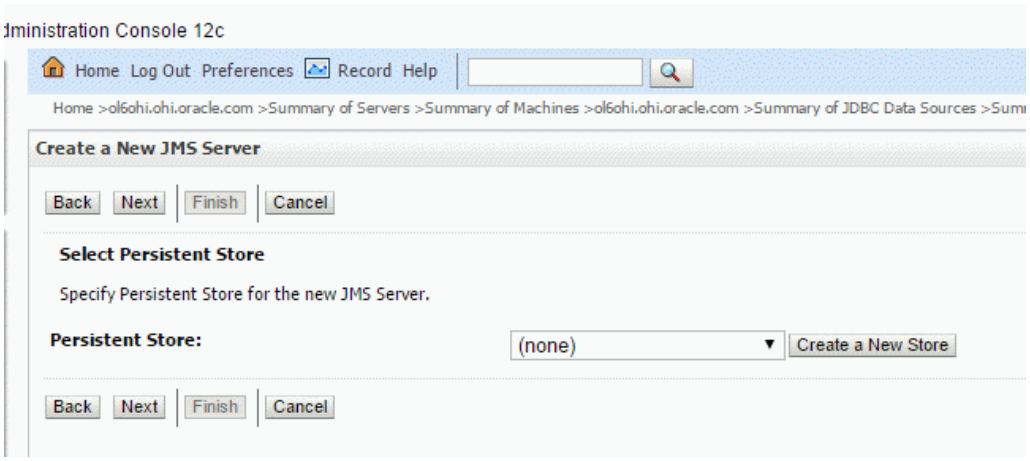
- Create Persistent Stores (eg. File Stores) for each managed server and associate the JMS server of each managed server with its Persistent Store.
- Create Distributed Queues instead of normal queues. Instead of linking the queues to a subdeployment, you can now use default targeting.

Refer to your Weblogic documentation for more detailed instructions.

...4.2. Create JMS Server

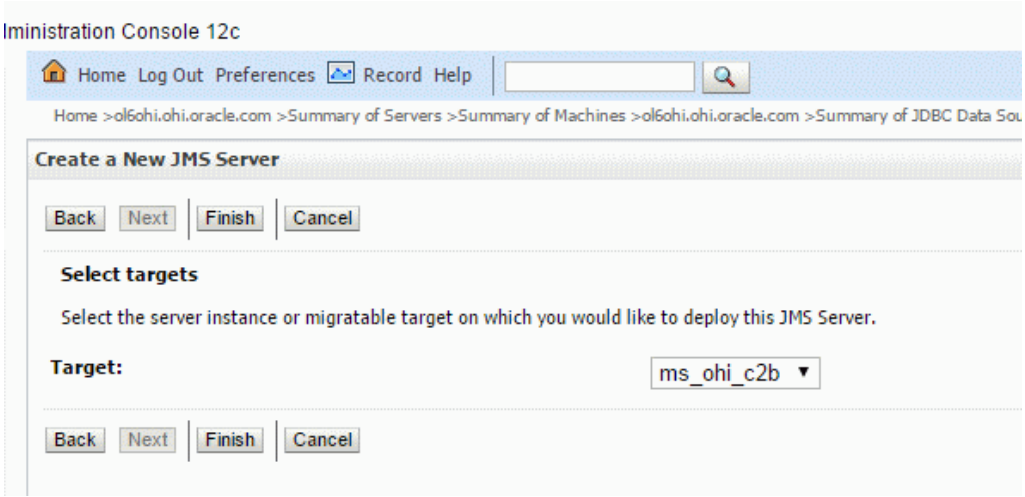
Choose in the Domain Structure pane Services > Messaging > JMS Servers and create a new JMS Server.





...4.2.1. Target JMS Server

The JMS server must be targeted to a managed server.



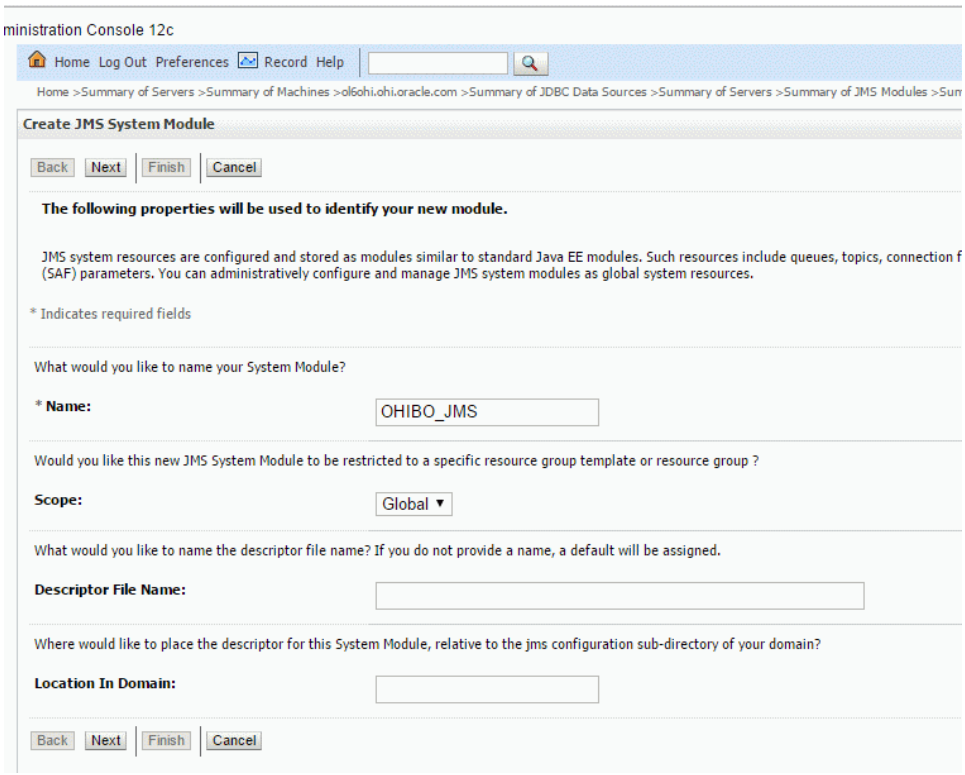
...4.3. Create and configure JMS Module

A JMS Module is simply a container to hold JMS resources. Apart from the JMS queues and connection factory, a JMS Module may also hold a 'Foreign Server'.

A Foreign Server represents a JMS provider that is outside the local WebLogic Server. It contains information that allows local server instance to reach a remote JNDI provider.

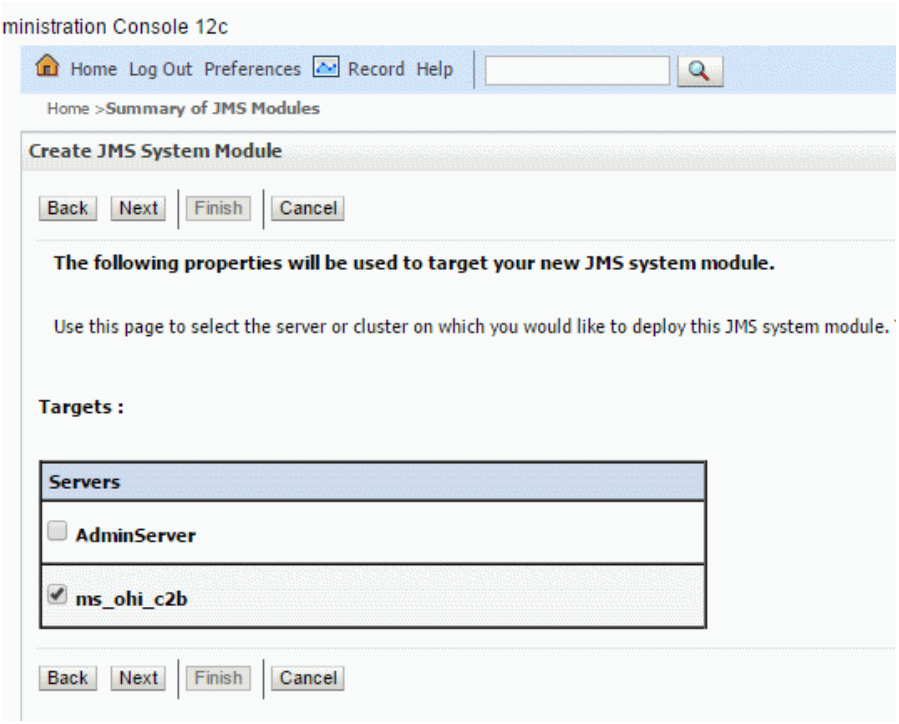
...4.3.1. Create System Module

Choose in the Domain Structure pane Services > Messaging > JMS Modules



...4.3.2. Target the JMS module

Next you must 'target' the new JMS module to a server. Choose the managed server of the ohi_c2b domain.



...4.3.3. Finish JMS Module creation

Finish the JMS Module creation. There is no need to continue to create the resources because we will create a subdeployment first.

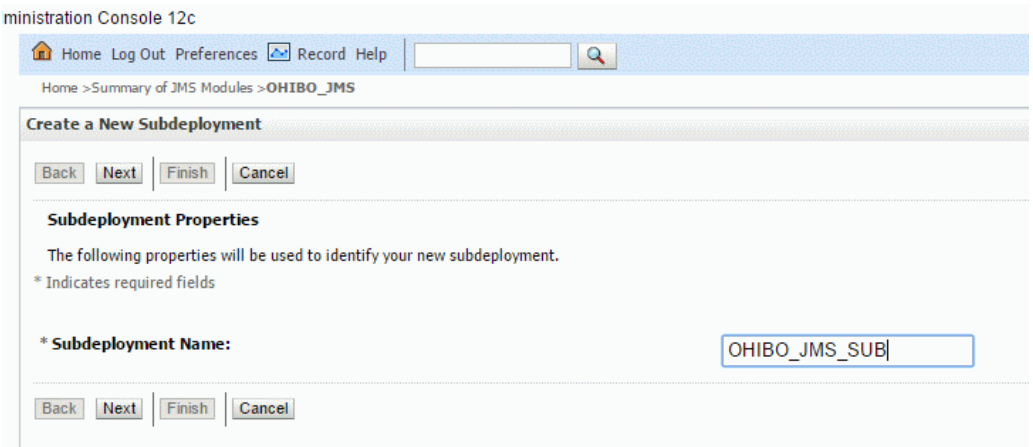
...4.4. Create subdeployment

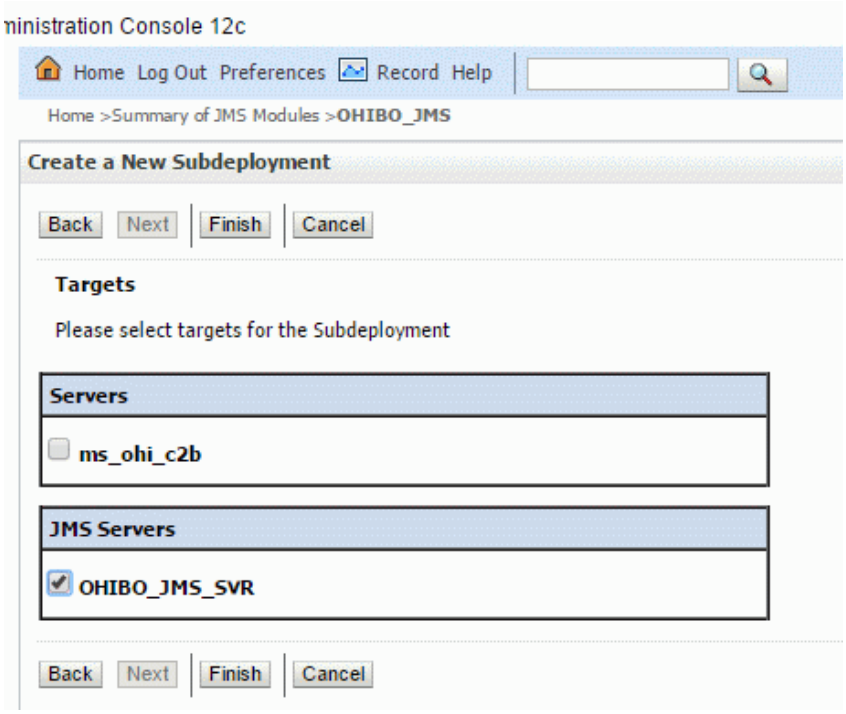
According to the Weblogic documentation, standalone queues must be targeted to JMS servers. Since this cannot be done directly, we use a subdeployment as a go-between.

More information can be found in:

Oracle® Fusion Middleware Administering JMS Resources for Oracle WebLogic Server

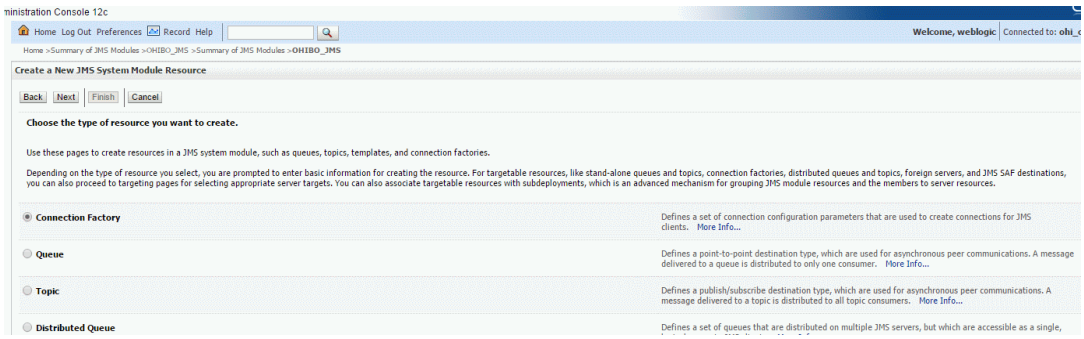
Navigate to the JMS Module you just created and choose the Subdeployments tab to create a Subdeployment.





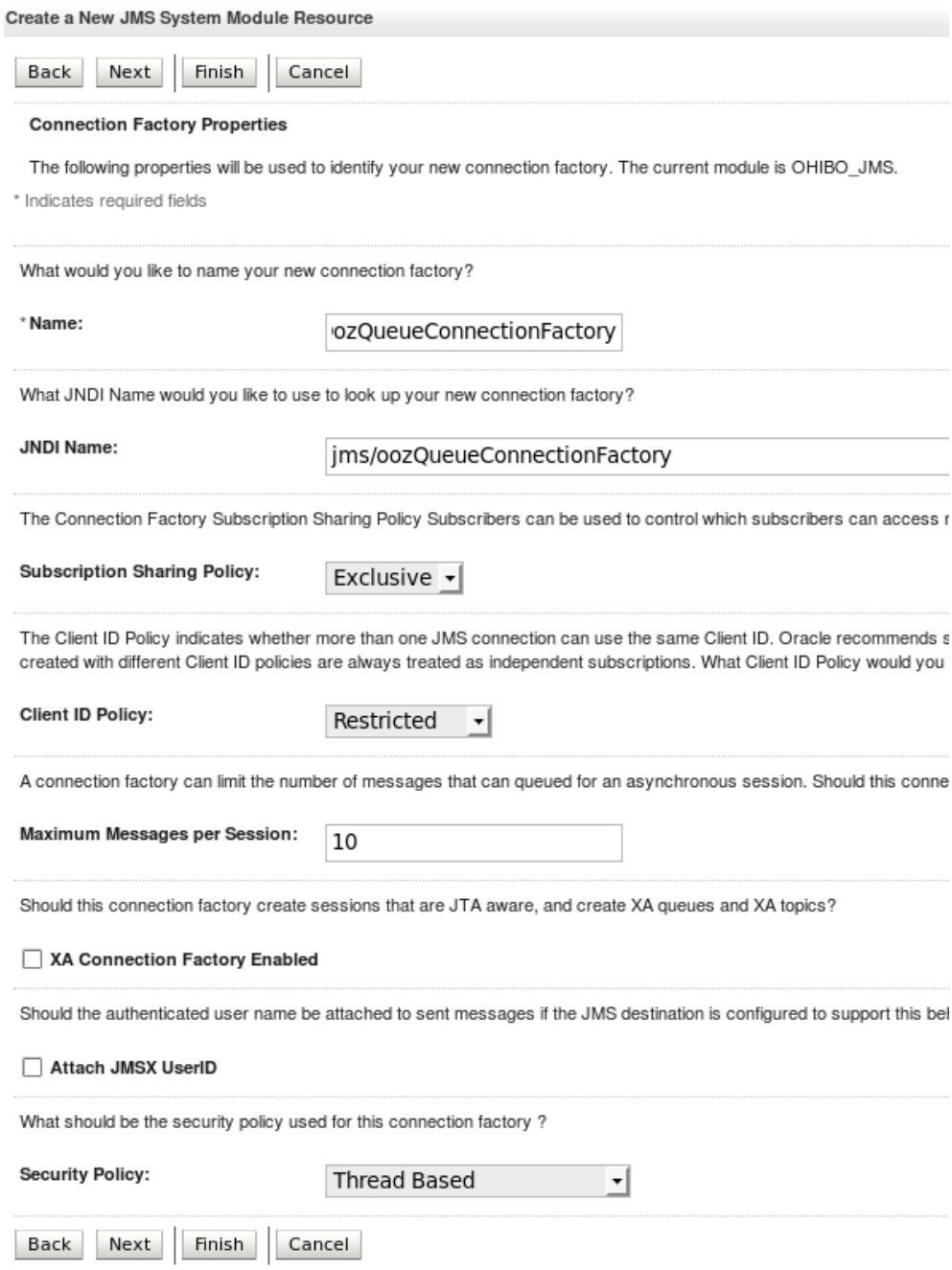
...4.5. Create Connection Factory

In the webservice program, the queue connection factory is used to access the JMS queues. Create a JMS System Module Resource of type 'Connection Factory'. For this select the JMS Module you created and use the New button on the Configuration tab to create a new resource.

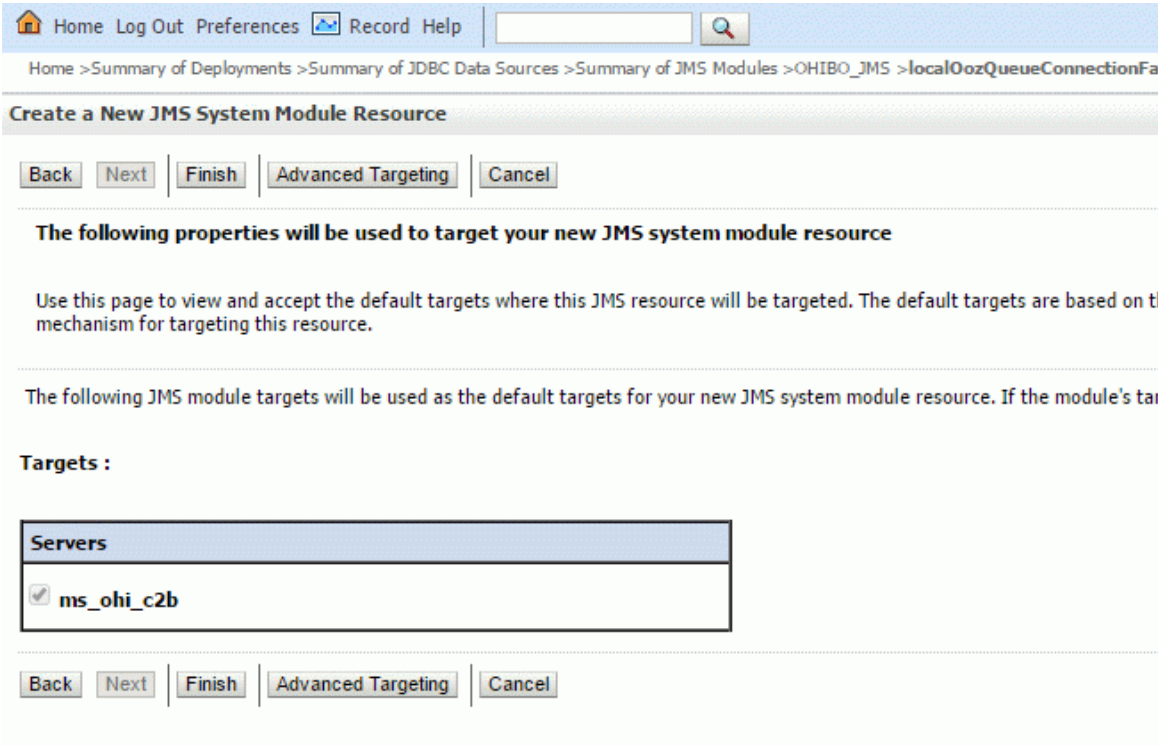


In the Connection Factory Properties page set the JNDI name to jms/oozQueueConnectionFactory. This JNDI name is used by the webservice program.

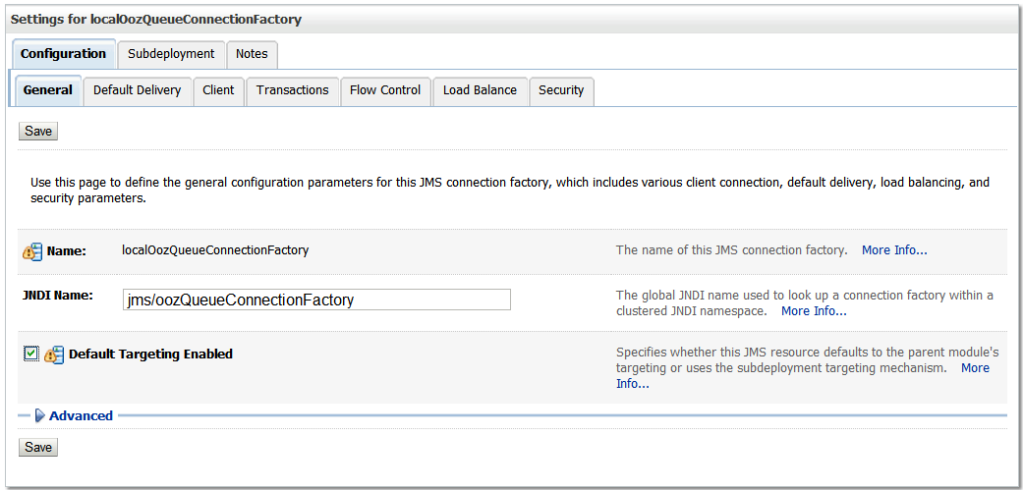
Leave other options default except for the 'XA Connection Factory Enabled' option which should be set to unchecked.


Create a New JMS System Module Resource
Back
Next
Finish
Cancel
Connection Factory Properties
The following properties will be used to identify your new connection factory. The current module is OHIBO_JMS.
* Indicates required fields
What would you like to name your new connection factory?
* Name:
ozQueueConnectionFactory
What JNDI Name would you like to use to look up your new connection factory?
JNDI Name:
jms/oozQueueConnectionFactory
The Connection Factory Subscription Sharing Policy Subscribers can be used to control which subscribers can access r
Subscription Sharing Policy:
Exclusive
The Client ID Policy indicates whether more than one JMS connection can use the same Client ID. Oracle recommends s
created with different Client ID policies are always treated as independent subscriptions. What Client ID Policy would you
Client ID Policy:
Restricted
A connection factory can limit the number of messages that can queued for an asynchronous session. Should this conne
Maximum Messages per Session:
10
Should this connection factory create sessions that are JTA aware, and create XA queues and XA topics?
XA Connection Factory Enabled
Should the authenticated user name be attached to sent messages if the JMS destination is configured to support this bel
Attach JMSX UserID
What should be the security policy used for this connection factory ?
Security Policy:
Thread Based
Back
Next
Finish
Cancel

Press Next.



After creating the Connection Factory, ensure that default targeting is enabled.

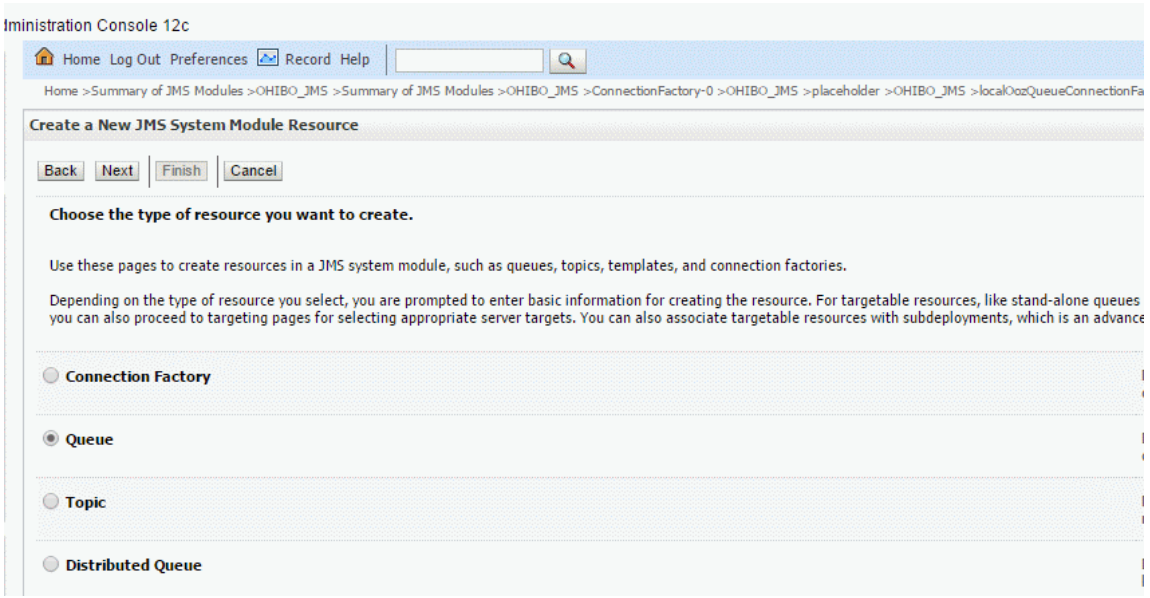


...4.6. Create Queues

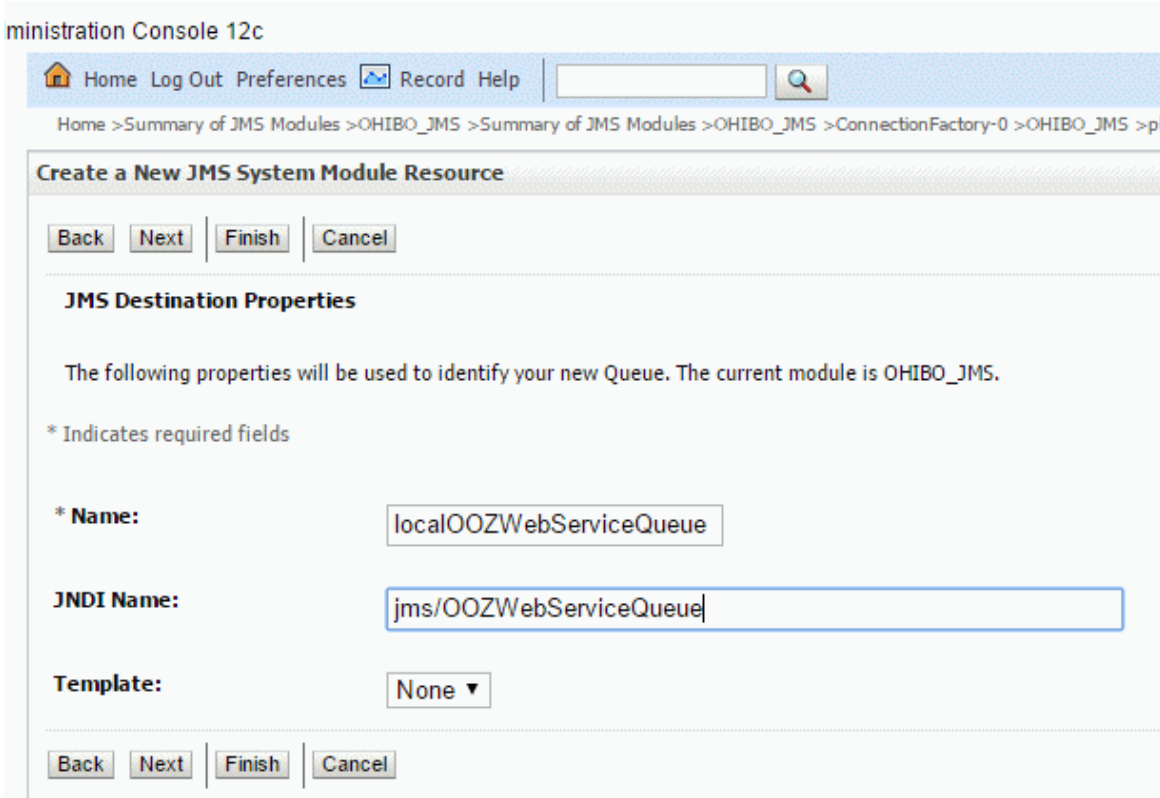
Create JMS queues with the following JNDI names (replace OOZ by C2B if you use the English version!):

Queue	JNDI Name
Request queue	jms/OOZWebServiceQueue
Response queue	jms/OOZWebServiceResponseQueue
Error queue	jms/oozErrorQueue

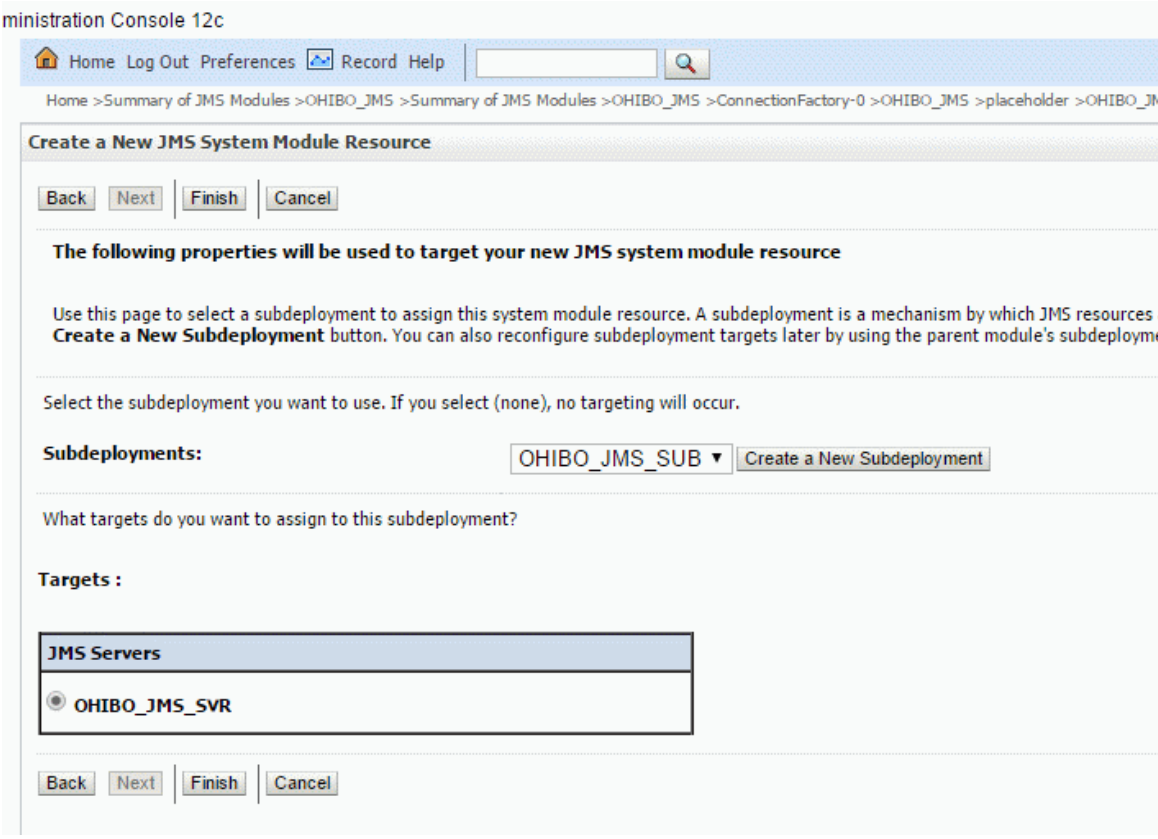
Create another JMS Module Resource within the earlier created JMS Module but this time of type ‘queue’:



Enter name and JNDI name. The JNDI name of each queue must correspond with an entry in the list above.

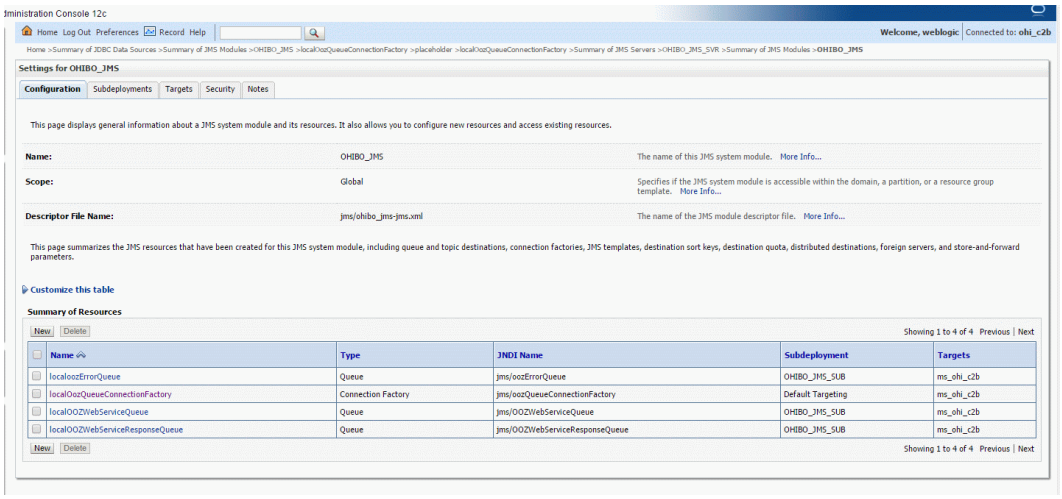


Finally, link the queue to the subdeployment:



The process for each queue is identical. The procedure above was used to create one of the queues, repeat this process for the remaining queues.

Finally, check that all queues are created:

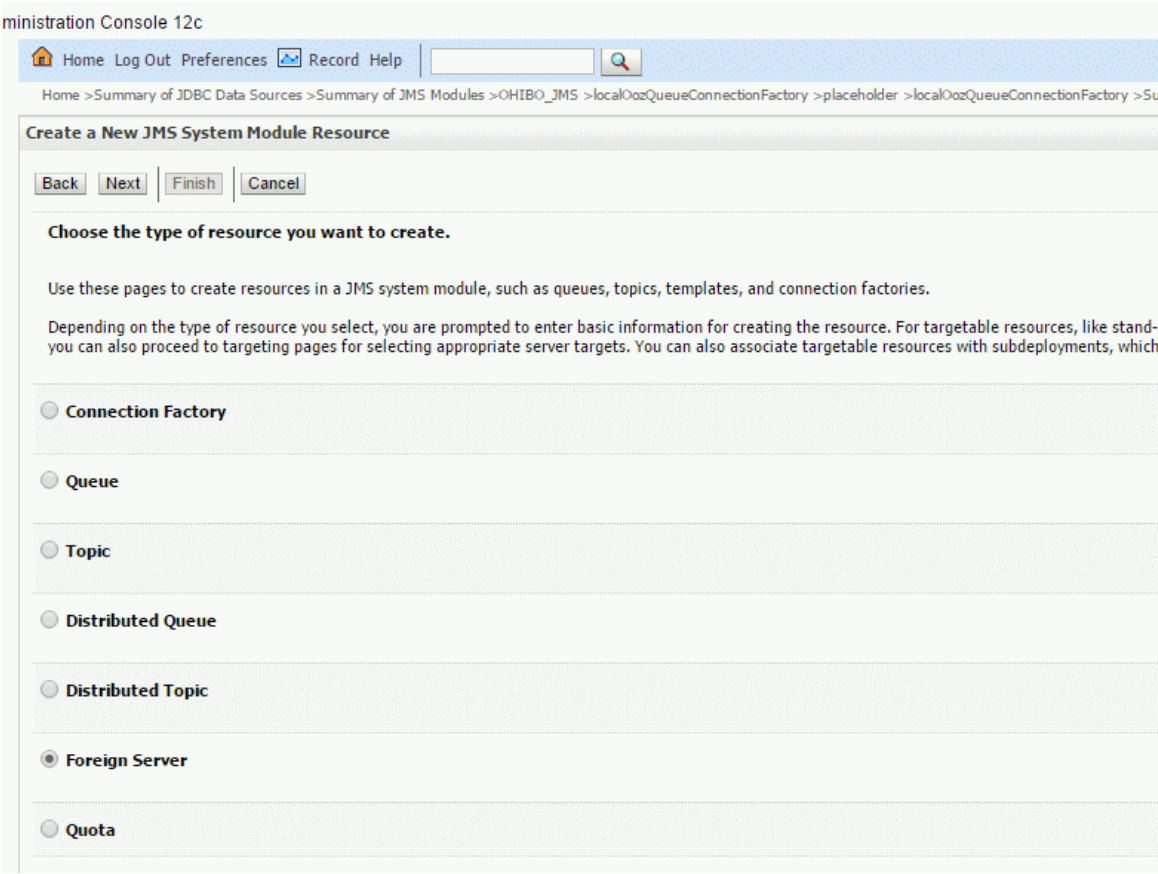


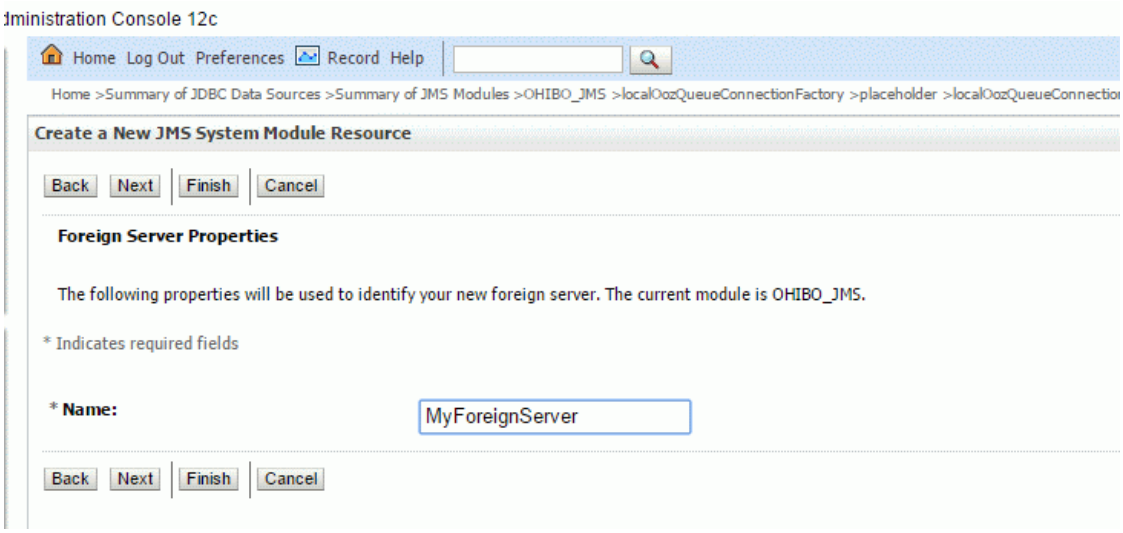
...4.7. Create foreign server (only when using OHI Self Service)

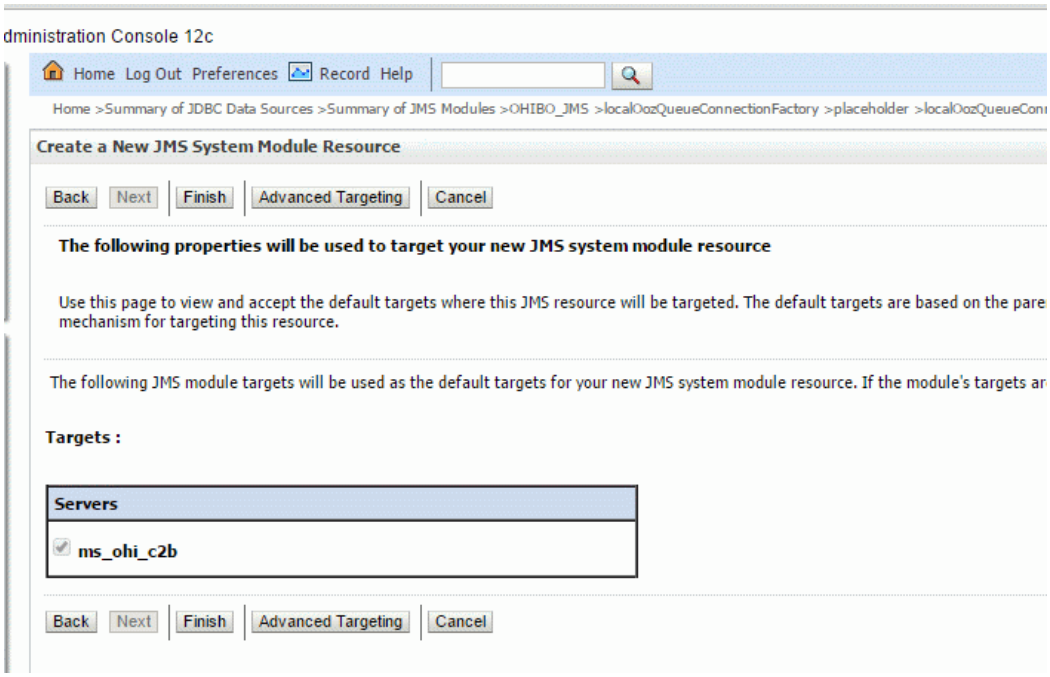
If you deploy OHI Connect to BackOffice for OHI Self Service you need to configure a Foreign Server, as a resource for the JMS System Module. A Foreign Server represents a JNDI provider that is outside the local WebLogic Server. It contains information that allows the local WebLogic Server instance to reach a remote JNDI provider.

Create a Foreign Server with the following property values:

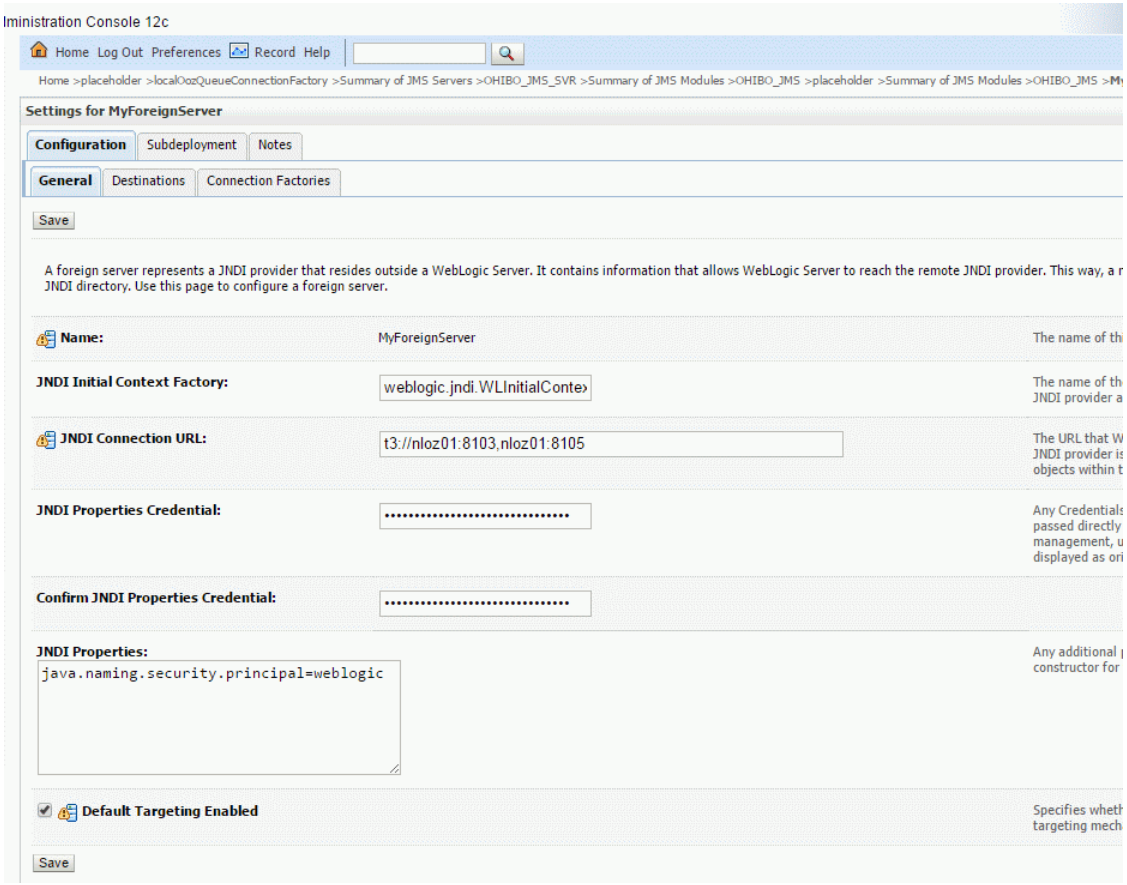
Name	Value
JNDI Initial Context Factory	**weblogic.jndi.WLInitialContextFactory** This is the default.
JNDI Connection URL	**t3://<host>:<port>,<host>:<port>** The URL that WebLogic Server will use to contact the JNDI provider. <host>: DNS name or IP address of the server that hosts the OHI Self Service. <port>: the port number from which you want to access the OHI Self Service server instance. e.g.: t3://nloz01:8103,nloz01:8105
JNDI Properties Credential	The credentials that must be set for the JNDI provider. Specify the secure password for the domain where OHI Self Service runs. These credentials must be specified along with the domain username in the JNDI Properties field.
JNDI Properties	**java.naming.security.principal=<*weblogic account*>** <*weblogic account*> is a secure username for the domain where the OHI Self Service runs, e.g. weblogic







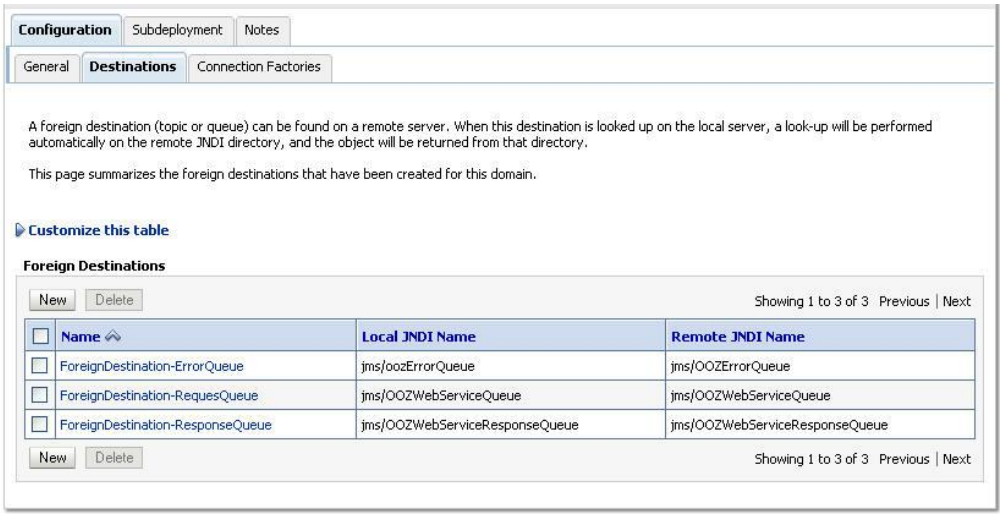
After creating the Foreign Server, fill in it's properties:



After creating a foreign server, you need to define a foreign connection factory and foreign destinations.

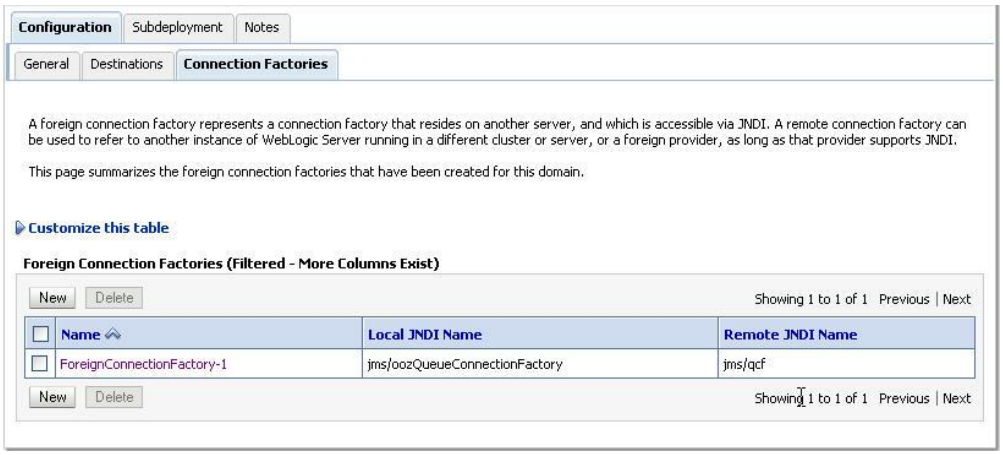
A Foreign Destination represents a queue that can be found on the remote server. Create foreign destinations with the following Local JNDI Names (replace OOZ by C2B if you use the English version!):

Local JNDI Name	Remote JNDI Name
jms/OOZWebServiceQueue	The JNDI name of the request queue at the OHI Self Service server.
jms/OOZWebServiceResponseQueue	The JNDI name of the response queue at the OHI Self Service server.
jms/oozErrorQueue	The JNDI name of the error queue at the OHI Self Service server.



You also need to define a foreign connection factory. Create a foreign connection factory with the following Local JNDI Name:

Local JNDI Name	Remote JNDI Name
jms/oozQueueConnectionFactory	The JNDI name of the connection factory in the remote JNDI provider, in this case the WebLogic instance where OHI Self Service running.



...4.8. Deploy the application EAR file

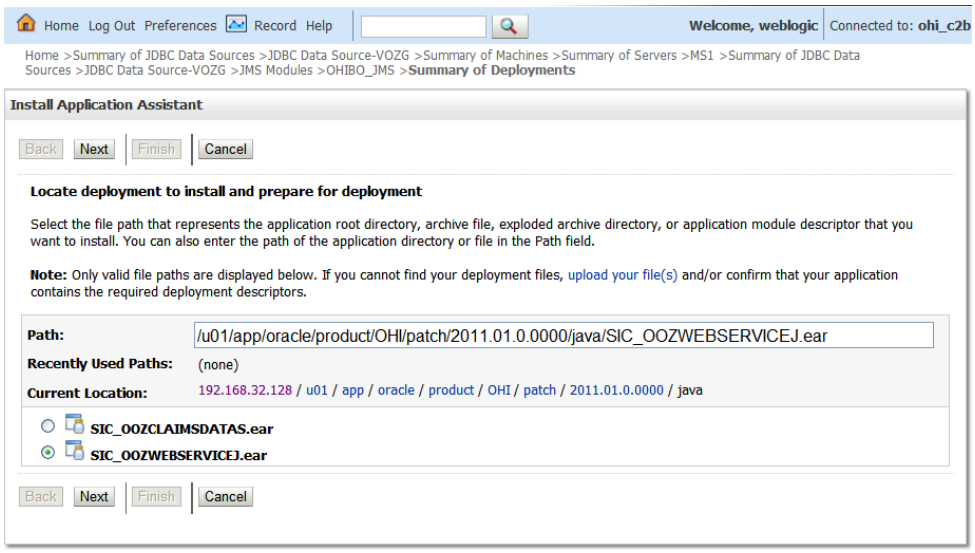
Ensure that the managed server is running and that the JMS Server is healthy.

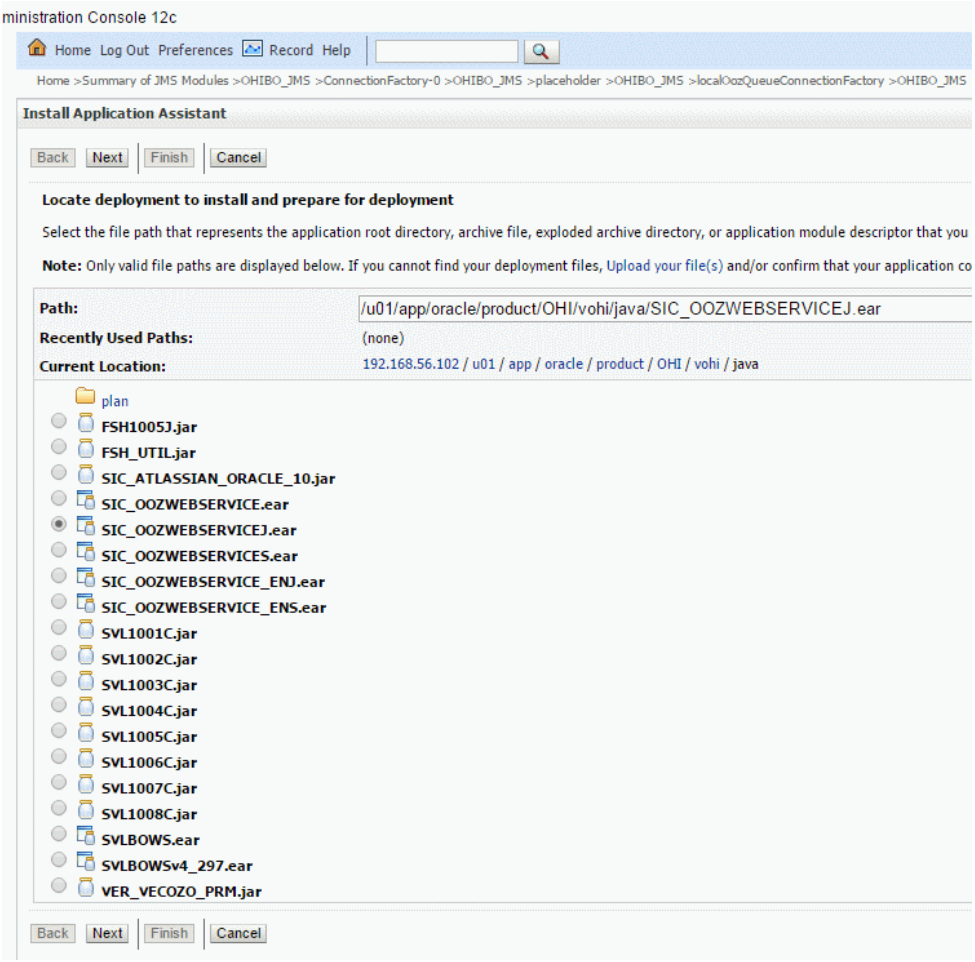
Now you can deploy the EAR file. Weblogic assumes that the EAR file can be found through the file system of the Admin Server host.

In our example, we will deploy the SIC_OOZWEBSERVICEJ.EAR through the GUI.

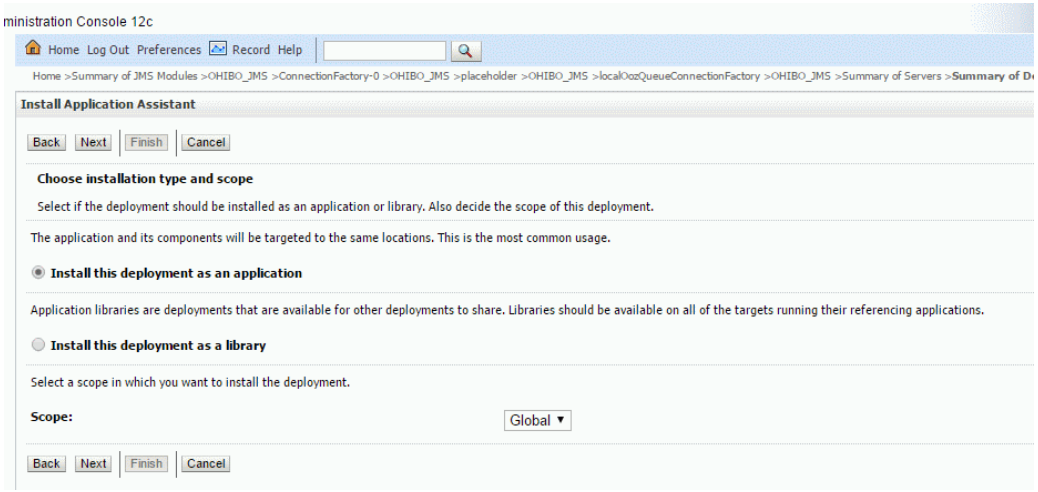
Go to 'deployments' page and start a new deployment by pressing the "Install"button (in Production Mode, do not Forget Lock & Edit to enable the Install option).

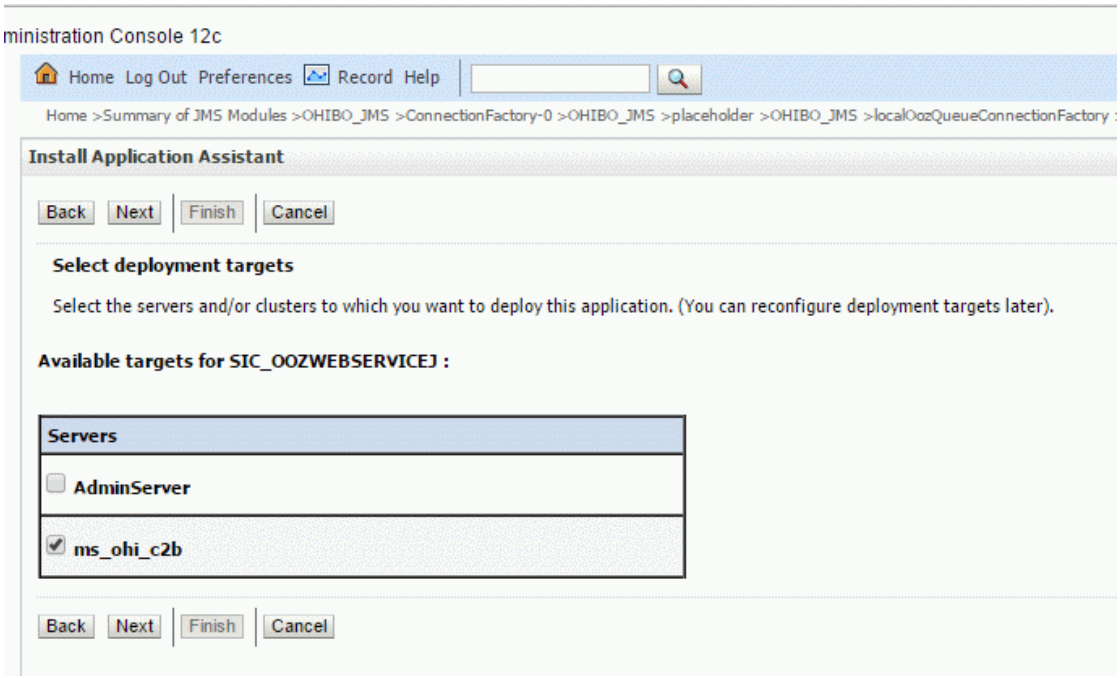
Select the EAR file to be deployed:





Install the EAR file as an application and target the EAR file to a managed server:





Do not change the 'optional settings'.

Finally start the newly deployed EAR file (start the server process).

If you receive a warning during start up, check the following:

- Is the JMS Server OK?
- Do you have a subdeployment targeted at the JMS Server?
- Have you used the correct JNDI names?
- Are the queues linked to the subdeployment?
- Has the connection factory been configured for default targeting?

.5 Administration

...5.1. Logging

The Connect to Back Office web services are set up for logging through Java Util Logging, in short JUL.

To set up logging, the following steps must be performed:

- Set up directory structure
- Configure C2B properties
- Configure WebLogic to use C2B properties

...5.1.1. Set up directory structure

It may be convenient to retain the $OZG_BASE/$OZG_ADMIN structure to support the Connect to Back Office web services:

- $OZG_BASE/java: used to locate EAR files
- $OZG_ADMIN: c2b.properties, output files for web service logging.

In the remainder of this chapter we will assume that

- $OZG_ADMIN will be used for locating log configuration and log output files.
- $OZG_ADMIN is located at /u01/app/oracle/product/OHI/admin
- $OZG_ADMIN does not include symbolic links

Beware that in an environment with multiple OHI Back Office installations sharing the same $OZG_ADMIN folder it may be convenient to use $OZG_BASE instead of $OZG_ADMIN for the log configuration and output files.

...5.1.2. Configure C2B properties

The Connect to Back Office (in short C2B) properties fie is specified in the startup options of the Managed Server with:

```
-Dc2b.properties=<path to properties file>
```

The file (suggested name 'c2b.properties') contains a number of properties to configure logging functionality. These values are generic for all webservices and specified only once.

The following properties are available:

Property key	Meaning
common.loging.filename	Specifies the file where the logging should be written towards.

common.logging.loglevel	The severity level of which logging should be written. This can be SEVERE, WARNING, INFO, CONFIG, FINE, FINER or FINEST.
common.logging.loglimit	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 approx. 1MB). Logs larger than the loglimit roll over to the next log file.
common.logging.logcount	The number of logfiles 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.
common.logging.logappend	A value to specify to append to the logfile or not. This should be filled out with either True or False (the default is True). A new logfile will be opened when append is False, when True the existing logfile will be appended.

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.

An example c2b.properties configuration file looks like this:

```
common.logging.filename=/ohi/logBase/c2b/c2b.log
common.logging.loglevel=SEVERE
common.logging.loglimit=10000000
common.logging.logcount=10
common.logging.logappend=TRUE
```

Finally, ensure that the file permissions of c2b.properties restrict access to authorised users only. In our test set up, we applied 'chmod 640' on c2b.properties.

...5.1.3. Configure WebLogic to use C2B properties

Start the WebLogic administration console.

- Select the managed server for which you want to set up logging.
- Select Server Start/Arguments and add
  -Dc2b.properties=/u01/app/oracle/product/OHI/admin/c2b.properties
- If you start the Managed Server using the scripts, also add these options to $DOMAIN_HOME/bin/startManagedWebLogic.sh:

```
JAVA_OPTIONS="-Dc2b.properties=
"/u01/app/oracle/product/OHI/vohi/c2b.properties" ${JAVA_OPTIONS}"
```

- Stop and start the managed server

Verify the output and log file which where created during startup of the managed server.

...5.2. Start WLS Node Manager

How to start the Node Manager depends on whether you configure a domain based Node Manager or a central Node Manager.

Activate environment settings:

```
. ozg_init.env $OZG_ORATAB_FRS12212
```

Central Node Manager:

```
cd $WL_HOME/server/bin

. ./setWLSEnv.sh

./startNodeManager.sh
```

Domain based Node Manager (assuming ohi_c2b as domain name):

```
cd $MW_HOME/user_projects/domains/ohi_c2b/bin

. ./setDomainEnv.sh

./startNodeManager.sh
```

...5.3. Start WLS Configuration Wizard

```
.ozg_init.env $OZG_ORATAB_FRS12212
```

Start `$WL_HOME/common/bin/config.sh`

You might run into the errors below:

Could not reserve enough space for object heap

Could not create the Java virtual machine.

In that case run the same command after issuing the following command:

```
export _JAVA_OPTIONS=-XX:MaxPermSize=512m
```

...5.4. Update deployment

When a new version of the .ear file is delivered make sure you update the deployment:

- Make sure the Admin Server process as well as the Managed Server process are running.
- Start the WebLogic console for the domain.
- Navigate to the deployment you need to update, and when running in 'Production Mode', activate 'Lock & Edit'.

- Select the checkbox in front of the existing deployment and use the ‘Update’ button to update the deployment.

.6 Appendix A - Installation of SIC_OOZWEBSERVICES (SOAP/HTTP)

It is possible to use a synchronous version of the SIC_OOZWEBSERVICEJ web service.

The EAR file for the synchronous version is SIC_OOZWEBSERVICES.ear

The installation does NOT require setting up JMS queues.

Installation:

- Ensure you have completed the installation of the WebLogic domain as described in Chapter 3.
- Deploy the SIC_OOZWEBSERVICES.ear, target to a managed server and start the web service

.7 Appendix B - Functional Testing

...7.1. Test SIC_OOZWEBSERVICEJ

It is possible to functionally test the SOAP/JMS interface with the tool HermesJMS.

Steps:

- Install HermesJMS
- Configure a session with the WebLogic host (config file needed)
- Select the jms/OOZWebServiceQueue
- Select Messages>Send Text Message and send an XML file with a request for SIC_OOZWEBSERVICEJ to jms/OOZWebServiceQueue
- Verify that the web service is responding by looking into the jms/OOZWebServiceResponseQueue and jms/oozErrorQueue.

Beware for the instructions above: replace OOZ by C2B if you use the English version!

...7.1.1. Install HermesJMS

HermesJMS is a free tool to interact with JMS providers. It allows you to play back prepared SOAP messages to functionally test the SOAP/JMS interface.

HermesJMS can be installed with and started from the tool SoapUI that can be used to test the synchronous version of C2B.

For more information and to download the standalone version of the tool go to

https://sourceforge.net/projects/hermesjms

Note that prior to installing HermesJMS you must create an environment variable JAVA_HOME which points to a JVM (version 1.5 or higher), or set it in <Hermes>\bin\hermes.bat

...7.1.2. Create wlfullclient.jar for interacting with Weblogic

You will need client library files for testing the web service. The easiest solution is to create the client libraries on the web logic server.

- Go to $WL_HOME/server/lib on the weblogic server.
- Issue the following command to create a full client JAR file for interacting with WebLogic:

  java –jar wljarbuilder.jar

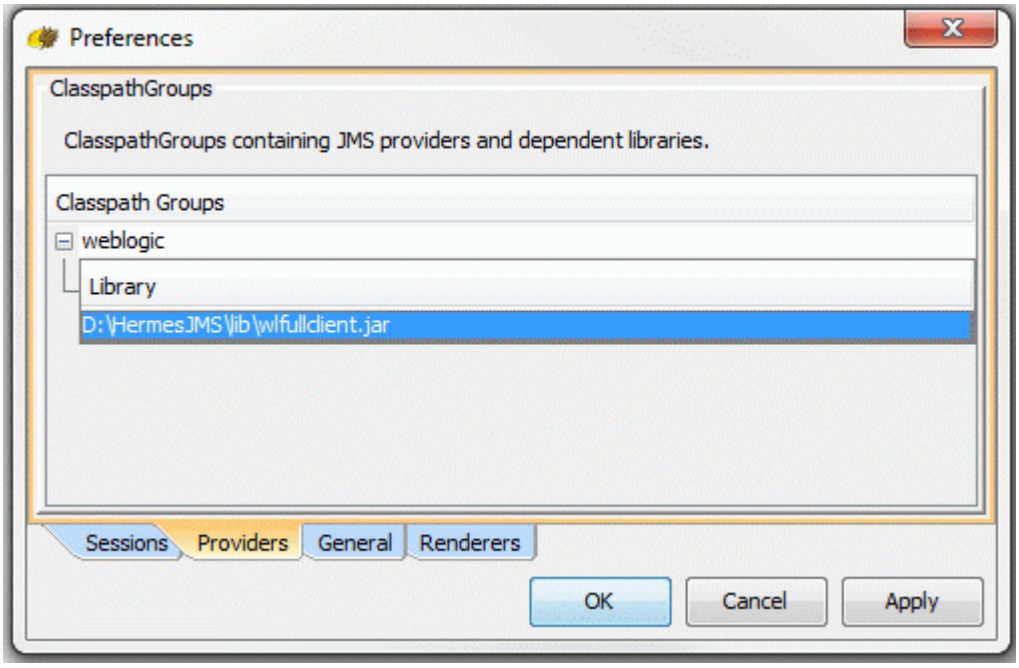
- Copy wlfullclient.jar to the computer running HermesJMS, to the <Hermes>\lib directory before starting HermesJMS.

...7.1.3. Configure a provider in HermesJMS

Start HermesJMS and open the dialog Options - Configuration.

Select the 'Providers' tab and add a classpath group named 'weblogic'.

Add the wlfullclient.jar file to the library. The file must be in the <Hermes>\lib directory.



...7.1.4. Configure a context in HermesJMS

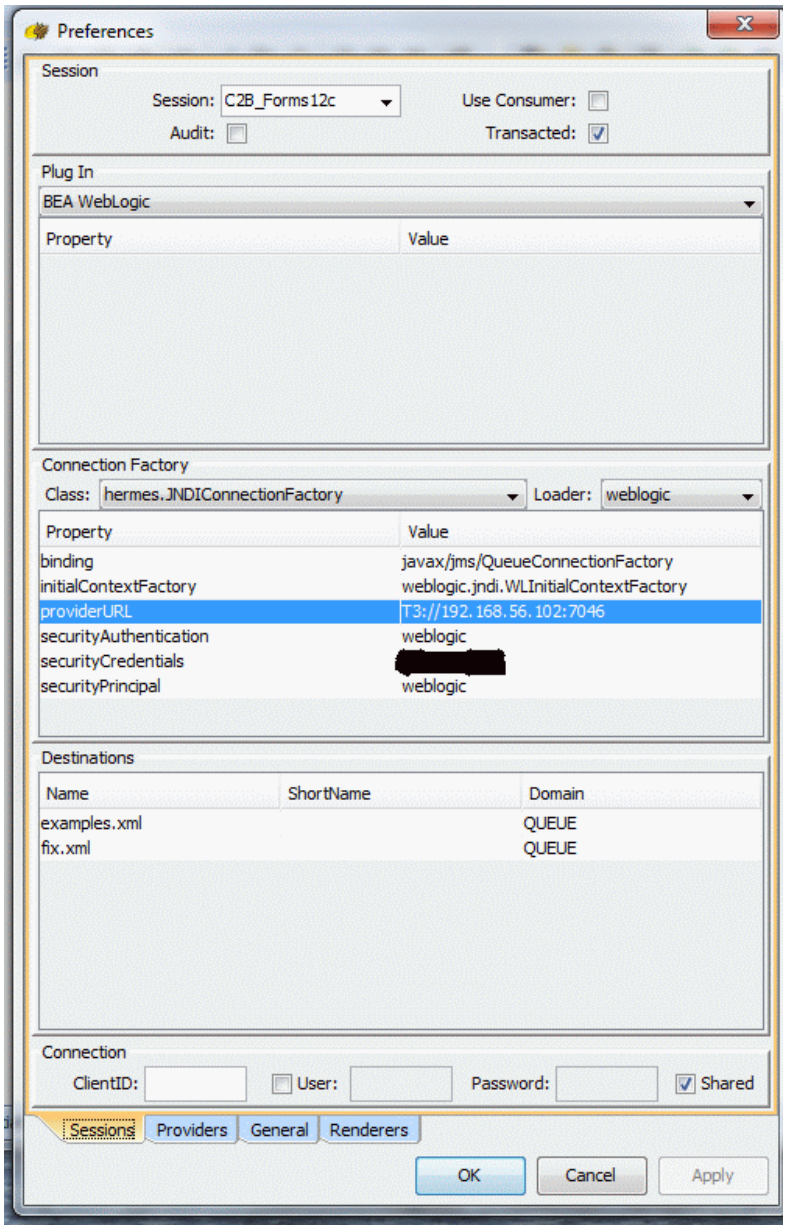
Open the 'Sessions' tab in the dialog Options - Configuration.

- Enter a session name, eg. 'C2B_Forms12c' or whatever has your fancy.
- Select the 'weblogic' loader.
- Select the BEA WebLogic plugin
- Select the hermes.JNDIConnectionFactory class
- Set the properties as indicated below:

Name	Value
providerURL	T3://<hostname>:<port> Hostname: managed server host Port number: managed server port number
initialContextFactory	weblogic.jndi.WLInitialContextFactory
Binding	javax/jms/QueueConnectionFactory
securityAuthentication	weblogic
securityPrincipal	<administrator>, e.g. weblogic

securityCredentials	<administrator_password>

At this point the screen should look like this:



Press OK to save the settings.

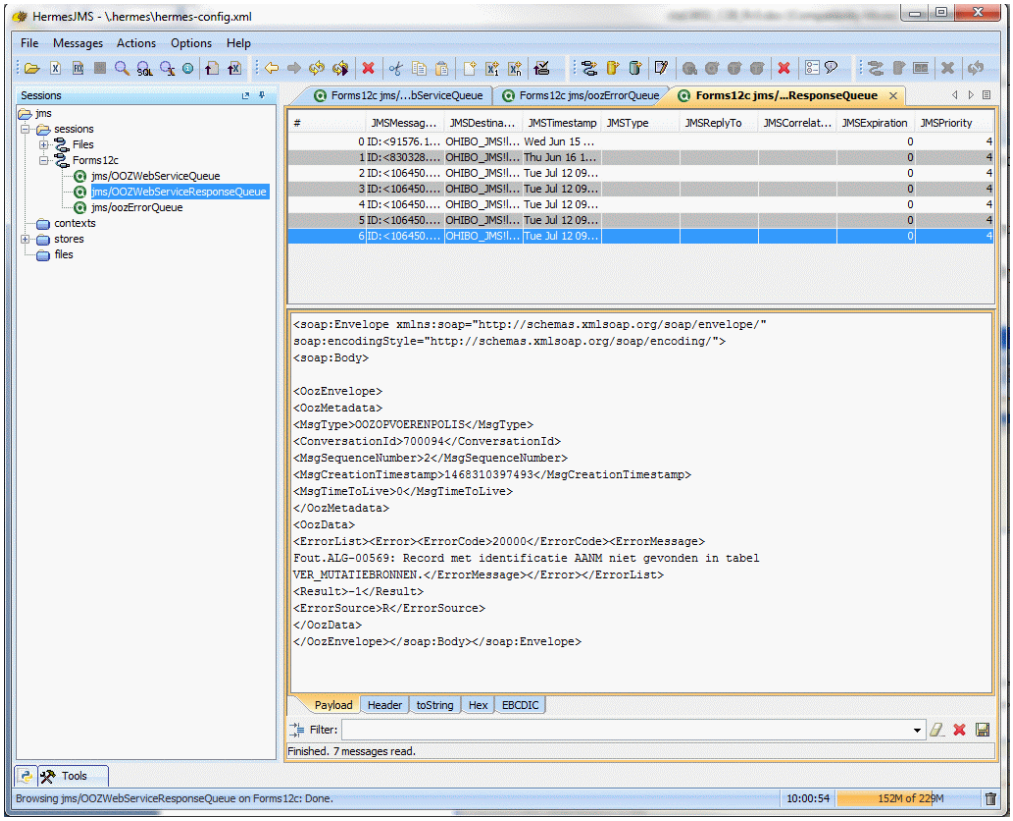
...7.1.5. Discover the JMS queues for SIC_OOZWEBSERVICEJ

Right-click on the session and select 'Discover'.

At this stage you will see the 3 queues which were set up for SIC_OOZWEBSERVICEJ.

- Create an XML file containing a test message. You may use the sample message for this. See below.
- Choose 'Send Text Message' in HermesJMS and select the file containing the test message. Once you click the 'Send' button, the message is sent to the SOAP/JMS service.
- Now double-click on the OOZWebServiceResponseQueue.
- Refresh the contents of this queue, until a message arrives from the web service.
- If no message arrives you may need to check the oozErrorQueue.

The screendump below shows a functional error message returned by SIC_OOZWEBSERVICEJ after processing the sample message:



...7.2. Test SIC_OOZWEBSERVICES

You can functionally test the SOAP/HTTP variant of the SIC_OOZWEBSERVICEJ web service by taking the OozEnvelope part of an existing SOAP/JMS message and inserting it in a SOAP/HTTP request.

The whole process consists of the following steps:

- Generate WSDL using WebLogic administration console
- Install SoapUI tool

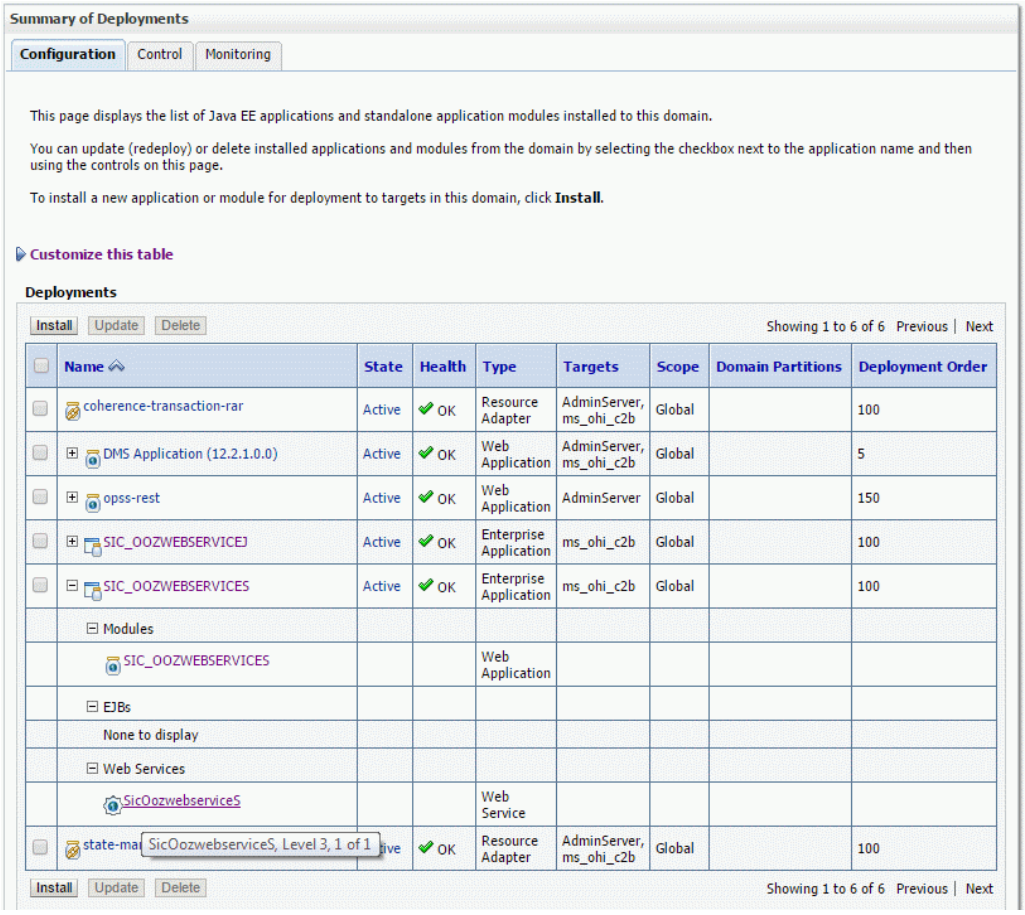
- Create SoapUI project
- Create SOAP request
- Run SOAP request

...7.2.1. Generate WSDL using WebLogic administration console

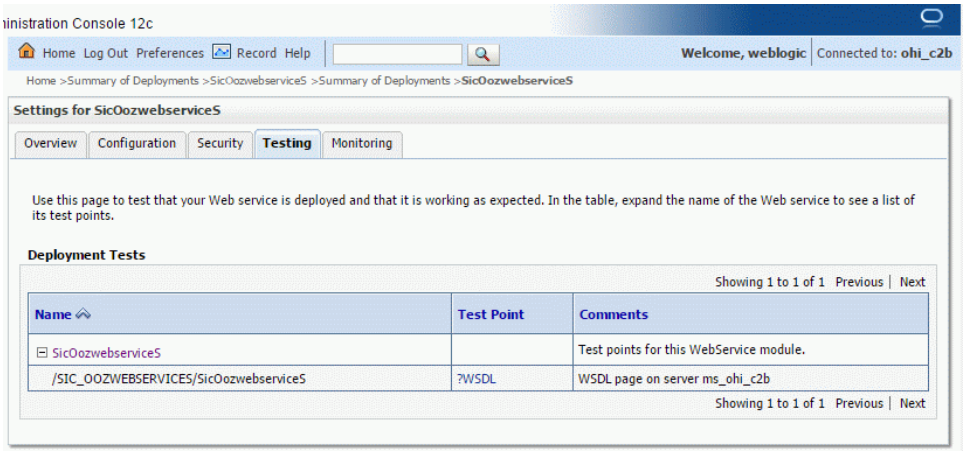
Start the WebLogic administration console.

Select the SIC_OOZWEBSERVICES web service in the deployments page.

Select the web service as shown below:



Click on the SicOozwebserviceS web service and choose the ‘Testing’ tab. Within this tab, expand SicOozwebserviceS like below:



Click on WSDL to generate the WSDL file.

Save as SicOozWebserviceS.wsdl.

...7.2.2. Install SoapUI tool

soapUI is an open source tool for functional testing. You can download it from www.soapui.org.

We have used version 5.2.1.

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

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

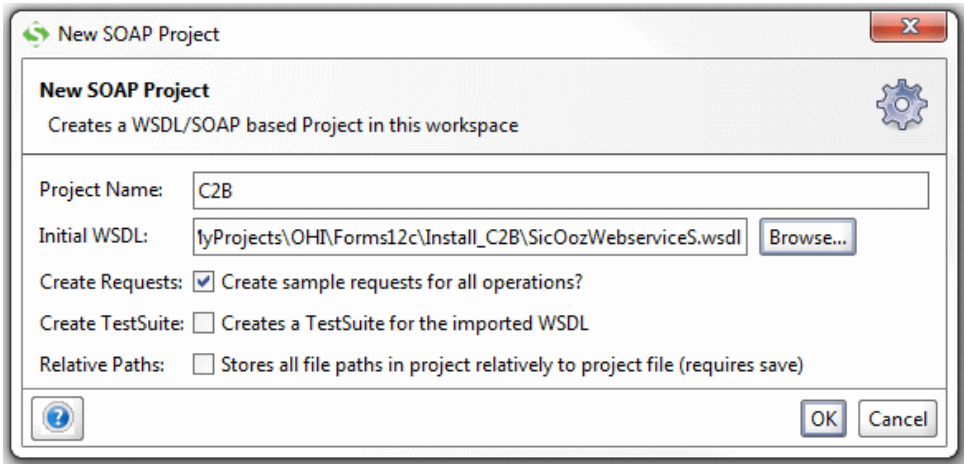
and make sure your shortcut uses that soapui.bat file.

...7.2.3. Create soapUI project

Start the soapUI tool, ignore timeout warnings.

Perform the following steps:

- Create a new SOAP project.
- Enter a project name (eg. C2B)
- Include the previously created WSDL file using 'Browse'
- Ensure that 'create request' is checked.
- Click OK to create the soapUI project.



...7.2.4. Create SOAP request

Use a text editor to view a SOAP/JMS message suitable for testing the SOAP/JMS variant of SIC_OOZWEBSERVICE. An example is given in the remainder of this chapter.

Select the contents of <OozEnvelope>, including the open and close tag for <OozEnvelope>:

<OozEnvelope>*Contents*</OozEnvelope>

Now wrap the selection in a CDATA fragment:

<![CDATA[<OozEnvelope>*Contents*</OozEnvelope>**]]>**

Open the C2B project in soapUI, expand the node oOZWebService and double-click on Request 1 to start the request editor.

Replace the ? in <arg0>?</arg0> with the CDATA fragment. The request will now look like this (in the left window of the request editor):

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:sic="http://sicoozwebservices.c2b.ohi.oracle.com/">
   <soapenv:Header/>
   <soapenv:Body>
      <sic:oOZWebService>
         <!--Optional:-->
         <arg0>
<![CDATA[<OozEnvelope>Contents</OozEnvelope>]]>
      </arg0>
      </sic:oOZWebService>
   </soapenv:Body>
</soapenv:Envelope>
```

...7.2.5. Run SOAP request

Click the PLAY button in the top left of the request editor window to submit the request. Once the response is received, it will be shown in the right window of the request editor.

The response is a SOAP message which may contain a :

1. Confirmation from Oracle Health Back Office.
   Action: no action required.

2. Functional error message from Oracle Health Back Office
   Action: no action required.

3. Technical error message from Oracle Health Back Office
   Action: verify that the web version is compatible with the version of Oracle Health Back Office.

4. Technical error message indicating that the web service could not be reached
   Action: ensure that the web service is up and running at the host and port number which were specified in the WSDL.

5. Technical error message regarding the contents of the request.
   Action: edit the message until it is accepted as a valid request.

...7.3. Sample message OOZWebService

The definition of the OOZWebService message payload itself that can be exchanged is documented in an XSD file.

The XSD files for the OHI Connect to Back Office web services can be found on the application server when the application server OHI software is installed.

In folder $OZG_BASE/xml the files SIC_OOZWEBSERVICE.xsd and SIC_OOZWEBSERVICE_EN.xsd define the structure of the messages for the Dutch respectively English version of the OOZWebService service.

...7.3.1. Dutch sample message

Below a sample Dutch message is given.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
 soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<soap:Body>
<OozEnvelope>
  <OozMetadata>
    <MsgType>OOZOPVOERENPOLIS</MsgType>
    <ConversationId>700094</ConversationId><!--wijzig-->
    <MsgSequenceNumber>1</MsgSequenceNumber>
    <MsgCreationTimestamp>21129975</MsgCreationTimestamp>
  </OozMetadata>
   <OozData>
     <WebService>
       <OOZOpvoerenPolis>
            <BerichtKenmerken>
               <DatumIngang>2011-04-01</DatumIngang>
               <Merk>PBSTD</Merk>
               <Functionaris>RSULING</Functionaris>
               <Brondocument>7094</Brondocument><!--wijzig-->
               <Mutatiebron>AANM</Mutatiebron>
               <IndGereedMelden>N</IndGereedMelden>
               <InMutatieToegestaan>N</InMutatieToegestaan>
               <IndMutatieWooneenheid>N</IndMutatieWooneenheid>
               <IndAanpassenRelAdres>J</IndAanpassenRelAdres>
            </BerichtKenmerken>
            <Relatie>
               <Relid>
                  <RelVolgNr>1</RelVolgNr>
                  <BSNNr>700000161</BSNNr><!--wijzig-->
                  <GebDatum>1994-05-14</GebDatum>
               </Relid>
               <VestAdres>
```

```
                  <Adres>
                     <Land>NL</Land>
                     <PCNr>1112</PCNr>
                     <PCLetter>XH</PCLetter>
                     <HuisNrPostb>4</HuisNrPostb>
                     <HuisNrToev/>
                  </Adres>
               </VestAdres>
               <Naam>Volumetest1</Naam>
               <Vrltrs>T</Vrltrs>
               <Geslacht>1</Geslacht>
               <GebDatum>1994-05-14</GebDatum>
               <BSNNr>700000161</BSNNr><!--wijzig-->
            </Relatie>
            <Polis>
               <VerzNmr>
                  <RelID>
                     <RelVolgNr>1</RelVolgNr>
                  </RelID>
               </VerzNmr>
               <FinInfo>
                  <OntvWijzePrem>2</OntvWijzePrem>
                  <OntvWijzeDecl>2</OntvWijzeDecl>
                  <IncFreq>1</IncFreq>
               </FinInfo>
               <OvereenkDatum>2011-04-01</OvereenkDatum>
               <Contract>
                <CollNrExt>274</CollNrExt>
               </Contract>
            </Polis>
            <Dekkingen>
               <Verzekerden>
                  <Relid>
                     <RelVolgNr>1</RelVolgNr>
                  </Relid>
                  <DatumIngang>2011-04-01</DatumIngang>
                  <DatumAanm>2011-03-02</DatumAanm>
               </Verzekerden>
               <MPEId>181</MPEId>
            </Dekkingen>
            <RedenToetr>005</RedenToetr>
         </OOZOpvoerenPolis>
         <Transactie>OOZOPVOERENPOLIS</Transactie>
      </WebService>
   </OozData>
</OozEnvelope>
</soap:Body></soap:Envelope>
```

...7.3.2. English sample message

Below a sample English message is given.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
 soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<soap:Body>
<OozEnvelope>
  <OozMetadata>
    <MsgType>C2BADDPOLICY</MsgType>
    <ConversationId>0-1611433041</ConversationId>
    <MsgSequenceNumber>1</MsgSequenceNumber>
    <MsgCreationTimestamp>1157707483046</MsgCreationTimestamp>
  </OozMetadata>
  <OozData>
    <WebService>
      <C2BAddPolicy>
        <MessageProperties>
```

```
      <StartDate>2012-01-01</StartDate>
      <Brand>PBSTD</Brand>
      <Officer>MSMALLEN</Officer>
      <IndApplyMod>N</IndApplyMod>
      <IndModHousehold>N</IndModHousehold>
    </MessageProperties>
    <Party>
      <PartyId>
        <PartySeqNo>1</PartySeqNo>
        <BSNNo>104949612</BSNNo>
        <DateOfBirth>1970-11-11</DateOfBirth>
      </PartyId>
      <Residence>
        <Address>
          <PCNo>2571</PCNo>
          <PCLetter>WG</PCLetter>
          <HouseNoPOBox>168</HouseNoPOBox>
        </Address>
      </Residence>
      <CodeType>
        <TypeCodeType>19</TypeCodeType>
        <CodeType>104949612</CodeType>
        <StartDate>2008-01-01</StartDate>
      </CodeType>
      <Name>M-1914 Verzekeringnemer 1</Name>
      <BSNNo>104949612</BSNNo>
      <Initials>B</Initials>
      <NameOrder>1</NameOrder>
      <Gender>1</Gender>
      <DoB>1970-11-11+01:00 </DoB>
      <MaritalStatus>2</MaritalStatus>
      <Origin>ZRG</Origin>
    </Party>
    <Policy>
      <PolicyHolder>
        <PartyId>
          <PartySeqNo>1</PartySeqNo>
        </PartyId>
      </PolicyHolder>
      <FinInfo>
        <PremiumAccount>
          <Account>
            <AccountNo>366718428</AccountNo>
          </Account>
        </PremiumAccount>
        <PremiumReceiptMethod>1</PremiumReceiptMethod>
        <ClaimsReceiptMethod>1</ClaimsReceiptMethod>
        <ColFreq>1</ColFreq>
      </FinInfo>
      <StartDate>2012-01-01</StartDate>
    </Policy>
    <Coverage>
      <Member>
        <PartyId>
          <PartySeqNo>1</PartySeqNo>
        </PartyId>
      </Member>
      <MPRStartDate>2006-01-01</MPRStartDate>
      <Package>PBSTD</Package>
      <PremStruc>STDPC</PremStruc>
      <CovStruc>STDDC</CovStruc>
      <YDStruc>STDER</YDStruc>
      <YDStep>ER0</YDStep>
      <ContrCare>PBNAT</ContrCare>
    </Coverage>
    <EntryReason>007</EntryReason>
    <PreviousInsurer>0701</PreviousInsurer>
  </C2BAddPolicy>
  <Transaction>C2BADDPOLICY</Transaction>
```

```
    </WebService>
  </OozData>
</OozEnvelope>
</soap:Body></soap:Envelope>
```