

# **Oracle Insurance for Health**

## **Oracle Insurance Web Services for Health Installation Guide for WLS**

version 1.2

Part number: E39878\_01

March 2013

Copyright © 2011, 2013, Oracle and/or its affiliates. All rights reserved.

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

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

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

#### **U.S. GOVERNMENT RIGHTS**

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

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

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

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

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

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

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

## CHANGE HISTORY

Release	Version	Changes
10.12.2.0.0	1.1	<ul style="list-style-type: none"><li>• Added Appendix B for functional testing.</li></ul>
10.13.1.0.0	1.2	<ul style="list-style-type: none"><li>• 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).</li></ul>

## RELATED DOCUMENTS

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

---

# Contents

1.	Introduction.....	1
1.1.	Purpose .....	1
1.2.	Audience.....	1
1.3.	Document structure.....	1
1.4.	Release.....	2
1.5.	Software versions .....	2
1.5.1.	OHI Back Office .....	2
1.5.2.	Oracle WebLogic Server.....	2
2.	Overview of the OHI Back Office web services.....	3
2.1.	SOAP/JMS web service.....	3
2.2.	SOAP/HTTP web service.....	4
3.	Installation preparation .....	5
3.1.	Verification of required files .....	5
3.1.1.	Required Files.....	5
3.2.	Database preparation .....	5
3.2.1.	Database objects .....	5
3.2.2.	Oracle account to be used .....	5
3.3.	WebLogic Preparation .....	6
3.3.1.	Create domain, managed server, machine.....	6
3.4.	Creating JDBC Data Source.....	7
3.5.	Start the managed server .....	9
4.	Installation of SIC_OOZCLAIMSDATAS .....	10
5.	Installation of SIC_OOZWEBSERVICEJ .....	11
5.1.	Before you start.....	11
5.1.1.	Alternative implementations.....	12
5.2.	Create JMS Server.....	13
5.2.1.	Target JMS Server .....	13
5.3.	Configure JMS Module.....	13
5.3.1.	Create System Module .....	14
5.3.2.	Target the JMS module.....	14
5.3.3.	Finish JMS Module creation .....	14
5.4.	Create subdeployment.....	14
5.4.1.	Target subdeployment to JMS server.....	15
5.5.	Create Connection Factory .....	15
5.6.	Create Queues.....	16
5.7.	Create foreign server (only when using OHI Self Service) .....	18
5.8.	Deploy the application EAR file .....	20
6.	Administration.....	22
6.1.	Logging .....	22
6.1.1.	Set up directory structure .....	22
6.1.2.	Create log4j configuration file.....	22
6.1.3.	Configure WebLogic to use log4j.....	23
6.2.	Start WLS Node manager.....	24
6.3.	Start WLS Configuration Wizard .....	24

7.	Appendix A - Installation of SIC_OOZWEBSERVICES.....	25
8.	Appendix B - Functional Testing.....	26
8.1.	Test SIC_OOZWEBSERVICEJ.....	26
8.1.1.	Install HermesJMS .....	26
8.1.2.	Create wfullclient.jar for interacting with Weblogic.....	26
8.1.3.	Configure a provider in HermesJMS .....	26
8.1.4.	Configure a context in HermesJMS .....	27
8.1.5.	Discover the JMS queues for SIC_OOZWEBSERVICEJ.....	28
8.1.6.	Send test message to jms/OOZWebserviceQueue.....	28
8.2.	Test SIC_OOZWEBSERVICES.....	29
8.2.1.	Generate WSDL using WebLogic administration console .....	29
8.2.2.	Install SoapUI tool .....	30
8.2.3.	Create soapUI project.....	30
8.2.4.	Create SOAP request.....	31
8.2.5.	Run SOAP request .....	32
8.3.	Sample message.....	32

---

## 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 11G environment (part of Oracle Fusion Middleware 11g).

### 1.2. Audience

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

- Working knowledge of Oracle WebLogic Server 11g version 10.3.3 or higher.
- The creation of Domain and Managed Server 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 for EAR files in WebLogic Server.

Administrators of Oracle10g AS may find the following comparison between Oracle10g AS and WebLogic Server useful:

[http://download.oracle.com/docs/cd/B31017\\_01/migrate.1013/b31269/compare\\_weblogic.htm](http://download.oracle.com/docs/cd/B31017_01/migrate.1013/b31269/compare_weblogic.htm)

### 1.3. Document structure

This document is organized as follows:

- Overview of the OHI Back Office web services  
Describes the SOAP/JMS and SOAP/HTTP web services on a conceptual level.
- Installation preparation  
Applies to both SOAP/JMS and SOAP/HTTP implementations, unless stated otherwise.
- Installation of SIC\_OOZCLAIMSDATAS (SOAP/HTTP)
- Installation of SIC\_OOZWEBSERVICEJ (SOAP/JMS)  
Describes the minimum procedure for configuring the queues needed by SIC\_OOZWEBSERVICEJ.
- Administration

Finally, appendix A describes to deploy the synchronous version of SIC\_OOZWEBSERVICE.

## 1.4. Release

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

- SIC\_OOZClaimsData (synchronous)  
Retrieve claims information.
- SIC\_OOZWebservice (synchronous and asynchronous)  
Multiple services 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 taken by an OHI web service 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 some 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

So we have SIC\_OOZCLAIMSDATAS.ear for the synchronous web service to retrieve claims data and we have SIC\_OOZWEBSERVICES.ear and SIC\_OOZWEBSERVICEJ.ear as two variants of our general purpose web services.

## 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.

### 1.5.1. OHI Back Office

---

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

### 1.5.2. Oracle WebLogic Server

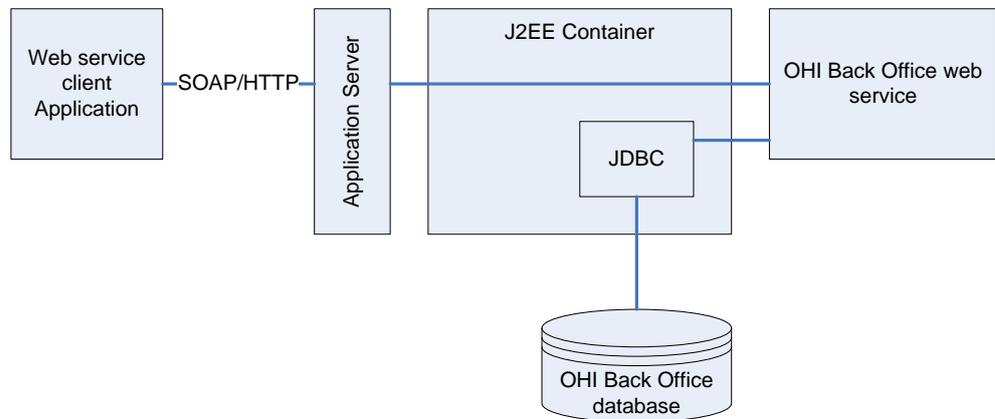
---

Starting with release 2011.01.0.0000 of OHI Back Office, the OHI Connect to Back Office must be deployed on Oracle WebLogic Server 11g. The version of the web services in the 2011.01.0.0000 release has been tested with Oracle WebLogic Server 11g version 10.3.3.

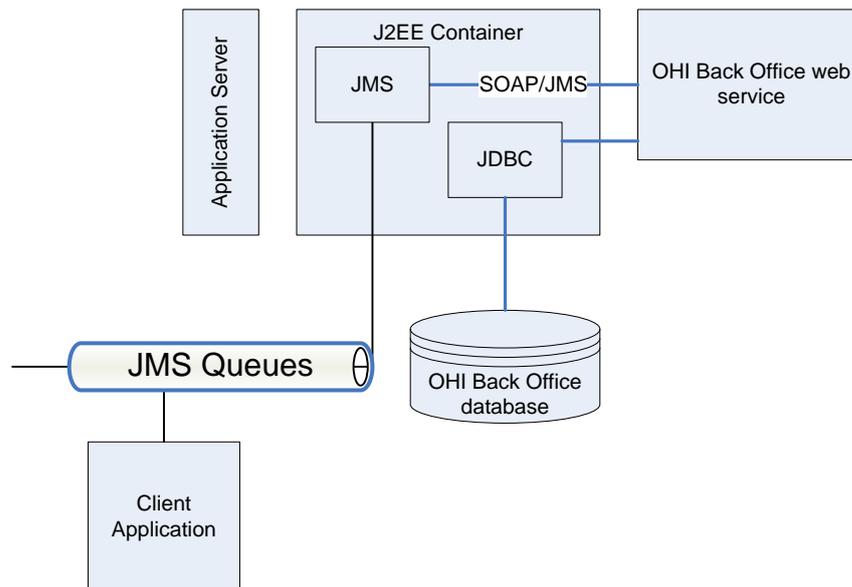
## 2. Overview of the OHI Back Office web services

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 BackOffice 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`.

## 2.2. SOAP/HTTP web service

The SOAP/HTTP web service (`OOZCLAIMSDATAS`) 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

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 by Oracle on Beehive Online (in future eDelivery) as part of a release delivery:

- SIC\_OOZWEBSERVICEJ.EAR  
(SOAP/JMS web service containing various services for Connect 2 Back Office)
- SIC\_OOZCLAIMSDATAS.EAR  
(SOAP/HTTP web service for claims enquiries)

### 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 runs. Normally this is the database server.
Port	Port number of the TNS listener Typically this is 1521 or 1526
Sid	The ORACLE SID 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 most strictly necessary rights have been granted (there is no standardized grant script available for this).

### 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

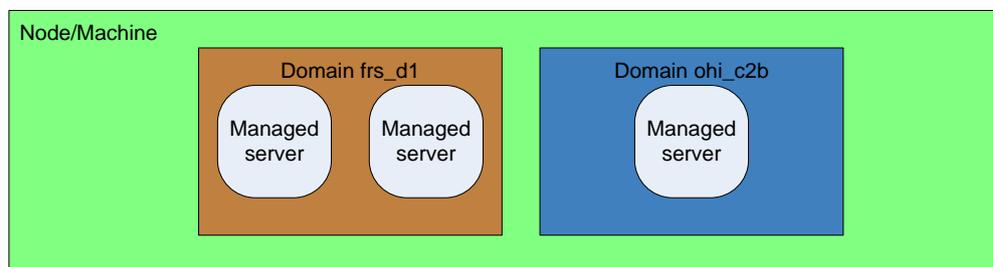
- Create domain, managed server, machine
- Create of data source

#### 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 and machine definitions for this new domain. Note in the diagram below how the 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

Start the configuration wizard for creating a new domain:

- Log in as the UNIX account running WebLogic (eg. oracle)
- `.ozg_init.env $OZG_ORATAB_FRS11G1`
- `$WL_HOME/common/bin/config.sh`

Generate a new domain for supporting web services with its own administration server as per paragraph 3.2.1 of **Doc[1]**.

Do not use port 7001 for the administration server but, for example, port 7070.

Start the administration server for this domain:

```
$WL_HOME/../../user_projects/domains/<domain>/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 3.2 of **Doc[1]** to create a new domain with a single managed server. 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).

Suggested names:

- Domain: ohi\_c2b
- Machine: machine1
- Managed Server: MS1 (we used port 7071)

### 3.4. Creating 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 Instance connection; Version: 9.0.1,9,2,0,10,11

#### 3.4.1.1. Create data source using JNDI name

Create a generic data source.

Home Log Out Preferences Record Help Welcome, weblogic Connected to: OHI\_C2B

Home > Summary of Deployments > Summary of Log Files > Server Log > Summary of Deployments > SIC\_OOZCLAIMSDATAS > Summary of Servers > Summary of Deployments > Summary of JDBC Data Sources > Summary of Servers > Summary of JDBC Data Sources

### Create a New JDBC Data Source

Back Next Finish Cancel

#### JDBC Data Source Properties

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

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

Name: JDBC Data Source-ONTW

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

JNDI Name: jdbc/wsapiDS

What database type would you like to select?

Database Type: Oracle

Back Next Finish Cancel

### 3.4.1.2. Select database driver

The screenshot shows the 'Create a New JDBC Data Source' wizard in the Oracle WebLogic console. The current step is 'JDBC Data Source Properties'. The 'Database Type' is set to 'Oracle'. The 'Database Driver' dropdown menu is open, showing 'Oracle's Driver (Thin) for Instance connections; Versions: 9.0.1, 9.2.0.10, 11' selected. Navigation buttons 'Back', 'Next', 'Finish', and 'Cancel' are visible at the top and bottom of the form.

### 3.4.1.3. Set up connection properties

Define the Connection Properties using the values described in 3.2.2.

The screenshot shows the 'Create a New JDBC Data Source' wizard in the Oracle WebLogic console. The current step is 'Connection Properties'. The 'Database Name' is 'ontw', 'Host Name' is 'nl0303.nl.oracle.com', 'Port' is '1527', and 'Database User Name' is 'ozg\_owner'. The 'Password' and 'Confirm Password' fields are masked with dots. Navigation buttons 'Back', 'Next', 'Finish', and 'Cancel' are visible at the bottom of the form.

### 3.4.1.4. Test database connection

Repeat previous step until the database connection test is successful.

### 3.4.1.5. Target the JDBC data source

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

Select a Managed Server as target.

Home Log Out Preferences Record Help  Welcome, weblogic Connected to: ohj\_c2b

Home > ohj\_c2b > Summary of JDBC Data Sources > Summary of Servers > Summary of JDBC Data Sources > JDBC Data Source-VOZG > Summary of Machines > Summary of Servers > MS1 > Summary of JDBC Data Sources > **JDBC Data Source-VOZG**

### Settings for JDBC Data Source-VOZG

Configuration **Targets** Monitoring Control Security Notes

Save

This page allows you to select the servers or clusters on which you would like to deploy this JDBC data source.

Servers
<input type="checkbox"/> AdminServer
<input checked="" type="checkbox"/> MS1

Save

### 3.5. Start the managed server

---

## 4. Installation of SIC\_OOZCLAIMSDATAS

The installation of a SOAP/HTTP web service is relatively straightforward because the service only depends on the JDBC data source which is used to connect to the OHI Back Office database.

First, carry out the preparatory activities as described in Ch 3.

Now you can deploy the SIC\_OOZCLAIMSDATAS.EAR file.

---

## 5. 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 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).

### 5.1. Before you start

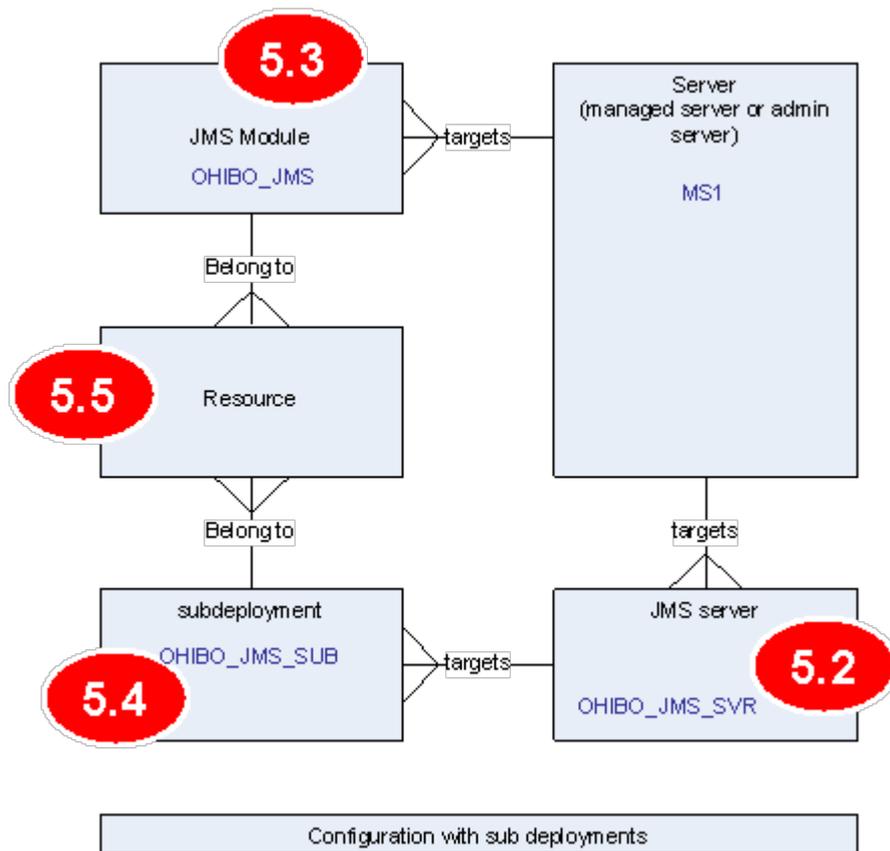
The following JMS queues must be created:

- jms/OOZWebServiceQueue
- jms/OOZWebServiceResponseQueue
- 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 relation 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:



### 5.1.1. 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.

## 5.2. Create JMS Server

The screenshot shows a web application interface for creating a new JMS server. At the top, there is a navigation bar with links for Home, Log Out, Preferences, Record, and Help. The user is logged in as 'weblogic' and connected to 'ohi\_c2b'. The breadcrumb trail indicates the current location: Home > Summary of Deployments > Summary of Diagnostics > Summary of Log Files > Server Log > JMS Modules > OHIBO\_JMS > JMS Modules > OHIBO\_JMS > localoozErrorQueue > Summary of JMS Servers.

The main form is titled 'Create a New JMS Server' and contains the following sections:

- JMS Server Properties:** A section with instructions: 'The following properties will be used to identify your new JMS Server.' and a note: '\* Indicates required fields'.
- Name:** A text input field labeled '\* Name:' containing the value 'OHIBO\_JMS\_SVR'.
- Persistent Store:** A dropdown menu labeled 'Persistent Store:' with '(none)' selected and a 'Create a New Store' button next to it.

Navigation buttons 'Back', 'Next', 'Finish', and 'Cancel' are present at the top and bottom of the form.

### 5.2.1. Target JMS Server

The JMS server must be targeted to a managed server.

This screenshot shows the same 'Create a New JMS Server' form, but at a different step. The breadcrumb trail is: Home > Summary of Deployments > Summary of Diagnostics > Summary of Log Files > Server Log > JMS Modules > OHIBO\_JMS > JMS Modules > OHIBO\_JMS > localoozErrorQueue > Summary of JMS Servers.

The form is titled 'Create a New JMS Server' and contains the following sections:

- Select targets:** A section with instructions: 'Select the server instance or migratable target on which you would like to deploy this JMS Server.'
- Target:** A dropdown menu labeled 'Target:' with 'MS1' selected.

Navigation buttons 'Back', 'Next', 'Finish', and 'Cancel' are present at the top and bottom of the form.

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

### 5.3.1. Create System Module

Home Log Out Preferences Record Help Welcome, weblogic Connected to: OHI\_C2B

Home > Summary of Servers > Summary of Deployments > Summary of JDBC Data Sources > Summary of Deployments > SIC\_OOZCLAIMSDATAS > Summary of Servers > Summary of Deployments > Summary of Servers > Summary of JMS Servers > JMS Modules

**Create JMS System Module**

Back Next Finish Cancel

The following properties will be used to identify your new module.

JMS system resources are configured and stored as modules similar to standard J2EE modules. Such resources include queues, topics, connection factories, templates, destination keys, quota, distributed queues, distributed topics, foreign servers, and JMS store-and-forward (SAF) parameters. You can administratively configure and manage JMS system modules as global system resources.

\* Indicates required fields

What would you like to name your System Module?

\* Name: OHIBO\_JMS

What would you like to name the descriptor file name? If you do not provide a name, a default will be assigned.

Descriptor File Name:

Where would like to place the descriptor for this System Module, relative to the jms configuration sub-directory of your domain?

Location In Domain:

Back Next Finish Cancel

### 5.3.2. Target the JMS module

Next you must 'target' the new JMS module to a server. Choose the managed server of the ohi\_c2b domain.

Home Log Out Preferences Record Help Welcome, weblogic Connected to: ohi\_c2b

Home > Summary of JDBC Data Sources > Summary of Servers > Summary of JDBC Data Sources > JDBC Data Source-VOZG > Summary of Machines > Summary of Servers > MS1 > Summary of JDBC Data Sources > JDBC Data Source-VOZG > JMS Modules

**Create JMS System Module**

Back Next Finish Cancel

The following properties will be used to target your new JMS system module.

Use this page to select the server or cluster on which you would like to deploy this JMS system module. You can reconfigure targets later if you wish.

Targets :

Servers
<input type="checkbox"/> AdminServer
<input checked="" type="checkbox"/> MS1

Back Next Finish Cancel

### 5.3.3. Finish JMS Module creation

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

## 5.4. 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:

[http://download.oracle.com/docs/cd/E12839\\_01/web.1111/e13738/basic\\_config.htm#i1151524](http://download.oracle.com/docs/cd/E12839_01/web.1111/e13738/basic_config.htm#i1151524)

The screenshot shows the 'Create a New Subdeployment' dialog box. At the top, there is a navigation bar with 'Home', 'Log Out', 'Preferences', 'Record', and 'Help'. The user is logged in as 'weblogic' and connected to 'ohi\_c2b'. The breadcrumb trail is: Home > JMS Log > Summary of JMS Servers > OHIBO\_JMS\_SVR > JMS Modules > OHIBO\_JMS > Roles > OHIBO\_JMS > localOOZWebServiceResponseQueue > JMS Modules > OHIBO\_JMS.

The dialog box has a title bar 'Create a New Subdeployment' and navigation buttons: Back, Next, Finish, and Cancel. Below the title bar, there is a section 'Subdeployment Properties' with the text: 'The following properties will be used to identify your new subdeployment. \* Indicates required fields'. A required field 'Subdeployment Name:' is present with the value 'OHIBO\_JMS\_SUB' entered in the text box. At the bottom, there are navigation buttons: Back, Next, Finish, and Cancel.

#### 5.4.1. Target subdeployment to JMS server

This screenshot shows the 'Create a New Subdeployment' dialog box at the 'Targets' step. The title bar is 'Create a New Subdeployment' with navigation buttons: Back, Next, Finish, and Cancel. The section 'Targets' contains the text: 'Please select targets for the Subdeployment'. There are two target selection boxes: 'Servers' with an unchecked checkbox for 'MS1', and 'JMS Servers' with a checked checkbox for 'OHIBO\_JMS\_SVR'. At the bottom, there are navigation buttons: Back, Next, Finish, and Cancel.

#### 5.5. 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':

The screenshot shows the 'Create a New JMS System Module Resource' dialog box. The navigation bar at the top shows the user is logged in as 'weblogic' and connected to 'OHI\_C2B'. The breadcrumb trail is: Home > Summary of Deployments > Summary of JDBC Data Sources > Summary of Deployments > SIC\_OOZCLAIMSDATAS > Summary of Servers > Summary of Deployments > Summary of Servers > Summary of JMS Servers > JMS Modules > OHIBO\_JMS.

The dialog box has a title bar 'Create a New JMS System Module Resource' and navigation buttons: Back, Next, Finish, and Cancel. Below the title bar, there is a section 'Choose the type of resource you want to create.' with the text: 'Use these pages to create resources in a JMS system module, such as queues, topics, templates, and connection factories. Depending on the type of resource you select, you are prompted to enter basic information for creating the resource. For targetable resources, like stand-alone queues and topics, connection factories, distributed queues and topics, foreign servers, and JMS SAF destinations, you can also proceed to targeting pages for selecting appropriate server targets. You can also associate targetable resources with subdeployments, which is an advanced mechanism for grouping JMS module resources and the members to server resources.'

There are four radio button options:

- Connection Factory**: Defines a set of connection configuration parameters that are used to create connections for JMS clients. [More Info...](#)
- Queue**: Defines a point-to-point destination type, which are used for asynchronous peer communications. A message delivered to a queue is distributed to only one consumer. [More Info...](#)
- Topic**: Defines a publish/subscribe destination type, which are used for asynchronous peer communications. A message delivered to a topic is distributed to all topic consumers. [More Info...](#)
- Distributed Queue**: Defines a set of queues that are distributed on multiple JMS servers, but which are accessible as a single, logical queue to JMS clients. [More Info...](#)

At the bottom, there are navigation buttons: Back, Next, Finish, and Cancel.

Set the JNDI name to `jms/oozQueueConnectionFactory`. This JNDI name is used by the webservice program.

Ensure that default targeting is enabled.

## 5.6. Create Queues

Create JMS queues with the following JNDI names:

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

Create a JMS Module Resource of type 'queue':

Home Log Out Preferences Record Help Welcome, weblogic Connected to: OH1\_C2B

Home > localOozQueueConnectionFactory > OHIBO\_JMS > placeholder > Summary of Deployments > JMS Modules > OHIBO\_JMS > JMS Modules > OHIBO\_JMS > JMS Modules > OHIBO\_JMS

### Create a New JMS System Module Resource

Back Next Finish Cancel

**Choose the type of resource you want to create.**

Use these pages to create resources in a JMS system module, such as queues, topics, templates, and connection factories.

Depending on the type of resource you select, you are prompted to enter basic information for creating the resource. For targetable resources, like stand-alone queues and topics, connection factories, distributed queues and topics, foreign servers, and JMS SAF destinations, you can also proceed to targeting pages for selecting appropriate server targets. You can also associate targetable resources with subdeployments, which is an advanced mechanism for grouping JMS module resources and the members to server resources.

<input type="radio"/> Connection Factory	Defines a set of connection configuration parameters that are used to create connections for JMS clients. <a href="#">More Info...</a>
<input checked="" type="radio"/> Queue	Defines a point-to-point destination type, which are used for asynchronous peer communications. A message delivered to a queue is distributed to only one consumer. <a href="#">More Info...</a>
<input type="radio"/> Topic	Defines a publish/subscribe destination type, which are used for asynchronous peer communications. A message delivered to a topic is distributed to all topic consumers. <a href="#">More Info...</a>
<input type="radio"/> Distributed Queue	Defines a set of queues that are distributed on multiple JMS servers, but which are accessible as a single, logical queue to JMS clients. <a href="#">More Info...</a>

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

Home Log Out Preferences Record Help Welcome, weblogic Connected to: OH1\_C2B

Home > localOozQueueConnectionFactory > OHIBO\_JMS > placeholder > Summary of Deployments > JMS Modules > OHIBO\_JMS > JMS Modules > OHIBO\_JMS > JMS Modules > OHIBO\_JMS

### Create a New JMS System Module Resource

Back Next Finish Cancel

**JMS Destination Properties**

The following properties will be used to identify your new Queue. The current module is OHIBO\_JMS.

\* Indicates required fields

\* Name: localOozWebServiceQueue

JNDI Name: jms/OozWebServiceQueue

Template: None

Back Next Finish Cancel

Finally, link the queue to the subdeployment:

Settings for localOozWebServiceQueue

Configuration Monitoring Control Security Subdeployment Notes

Save

Use this page to select a subdeployment to assign this destination resource. A subdeployment is a mechanism by which JMS destinations are grouped and targeted to a single JMS server instance. To reconfigure a subdeployment's targets, use the parent module's subdeployment management page.

Subdeployment: OHIBO\_JMS\_SUB

The subdeployment groups within this JMS module. Subdeployments enable you to deploy some resources in a JMS module to a JMS server and other JMS resources to a server instance or cluster. [More Info...](#)

Save

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

Finally, check that all queues are created:

Settings for OHIBO\_JMS

Configuration Subdeployments Targets Security Notes

This page displays general information about a JMS system module and its resources. It also allows you to configure new resources and access existing resources.

**Name:** OHIBO\_JMS The name of this JMS system module. [More Info...](#)

**Descriptor File Name:** jms/ohibo\_jms-jms.xml The name of the JMS module descriptor file. [More Info...](#)

This page summarizes the JMS resources that have been created for this JMS system module, including queue and topic destinations, connection factories, JMS templates, destination sort keys, destination quota, distributed destinations, foreign servers, and store-and-forward parameters.

[Customize this table](#)

**Summary of Resources**

New Delete Showing 1 to 4 of 4 Previous | Next

<input type="checkbox"/>	Name ↕	Type	JNDI Name	Subdeployment	Targets
<input type="checkbox"/>	localoozErrorQueue	Queue	.jms/oozErrorQueue	OHIBO_JMS_SUB	OHIBO_JMS_SVR
<input type="checkbox"/>	localOozQueueConnectionFactory	Connection Factory	.jms/oozQueueConnectionFactory	Default Targetting	MS1
<input type="checkbox"/>	localOOZWebServiceQueue	Queue	.jms/OOZWebServiceQueue	OHIBO_JMS_SUB	OHIBO_JMS_SVR
<input type="checkbox"/>	localOOZWebServiceResponseQueue	Queue	.jms/OOZWebServiceResponseQueue	OHIBO_JMS_SUB	OHIBO_JMS_SVR

New Delete Showing 1 to 4 of 4 Previous | Next

## 5.7. 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. A Foreign Server represents a JNDI provider that is outside local WebLogic Server. It contains information that allows a 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	<b>weblogic.jndi.WLInitialContextFactory</b> This is the default.
JNDI Connection URL	<b>t3://&lt;host&gt;:&lt;port&gt;,&lt;host&gt;:&lt;port&gt;</b> 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	<b>java.naming.security.principal=&lt;weblogic account&gt;</b>  <weblogic account> is a secure username for the domain where the OHI Self Service runs, e.g. weblogic

**Configuration** Subdeployment Notes

**General** Destinations Connection Factories

Save

A foreign server represents a JNDI provider that resides outside a WebLogic Server. It contains information that allows WebLogic Server to reach the remote JNDI provider. This way, a number of connection factory and destination objects (queues or topics) can be defined on one JNDI directory. Use this page to configure a foreign server.

**Name:** nloz01 The name of this foreign server. [More Info...](#)

**JNDI Initial Context Factory:** weblogic.jndi.WLInitialC The name of the class that must be instantiated to access the JNDI provider. This class name depends on the JNDI provider and the vendor that are being used. [More Info...](#)

**JNDI Connection URL:** t3://nloz01:8103,nloz01:8105 The URL that WebLogic Server will use to contact the JNDI provider. The syntax of this URL depends on which JNDI provider is being used. For WebLogic JMS, leave this field blank if you are referencing WebLogic JMS objects within the same cluster. [More Info...](#)

**JNDI Properties Credential:** ..... Any Credentials that must be set for the JNDI provider. These Credentials will be part of the properties will be passed directly to the constructor for the JNDI provider's InitialContext class. Note: For secure credential management, use the Credential field. Using the Properties field results in the credential being stored and displayed as originally entered [More Info...](#)

**Confirm JNDI Properties Credential:** .....

**JNDI Properties:**  
 Any additional properties that must be set for the JNDI provider. These properties will be passed directly to the constructor for the JNDI provider's InitialContext class. [More Info...](#)

**Default Targeting Enabled** Specifies whether this JMS resource defaults to the parent module's targeting or uses the subdeployment targeting mechanism. [More Info...](#)

Save

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:

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.

**Configuration** Subdeployment Notes

General **Destinations** Connection Factories

A foreign destination (topic or queue) can be found on a remote server. When this destination is looked up on the local server, a look-up will be performed automatically on the remote JNDI directory, and the object will be returned from that directory.

This page summarizes the foreign destinations that have been created for this domain.

[Customize this table](#)

**Foreign Destinations**

New Delete Showing 1 to 3 of 3 Previous Next

<input type="checkbox"/>	Name ↕	Local JNDI Name	Remote JNDI Name
<input type="checkbox"/>	ForeignDestination-ErrorQueue	jms/oozErrorQueue	jms/OOZErrorQueue
<input type="checkbox"/>	ForeignDestination-RequestQueue	jms/OOZWebServiceQueue	jms/OOZWebServiceQueue
<input type="checkbox"/>	ForeignDestination-ResponseQueue	jms/OOZWebServiceResponseQueue	jms/OOZWebServiceResponseQueue

New Delete Showing 1 to 3 of 3 Previous Next

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.

**Configuration** Subdeployment Notes

General Destinations **Connection Factories**

A foreign connection factory represents a connection factory that resides on another server, and which is accessible via JNDI. A remote connection factory can be used to refer to another instance of WebLogic Server running in a different cluster or server, or a foreign provider, as long as that provider supports JNDI.

This page summarizes the foreign connection factories that have been created for this domain.

[Customize this table](#)

**Foreign Connection Factories (Filtered - More Columns Exist)**

New Delete Showing 1 to 1 of 1 Previous Next

<input type="checkbox"/>	Name ↕	Local JNDI Name	Remote JNDI Name
<input type="checkbox"/>	ForeignConnectionFactory-1	jms/oozQueueConnectionFactory	jms/qcf

New Delete Showing 1 to 1 of 1 Previous Next

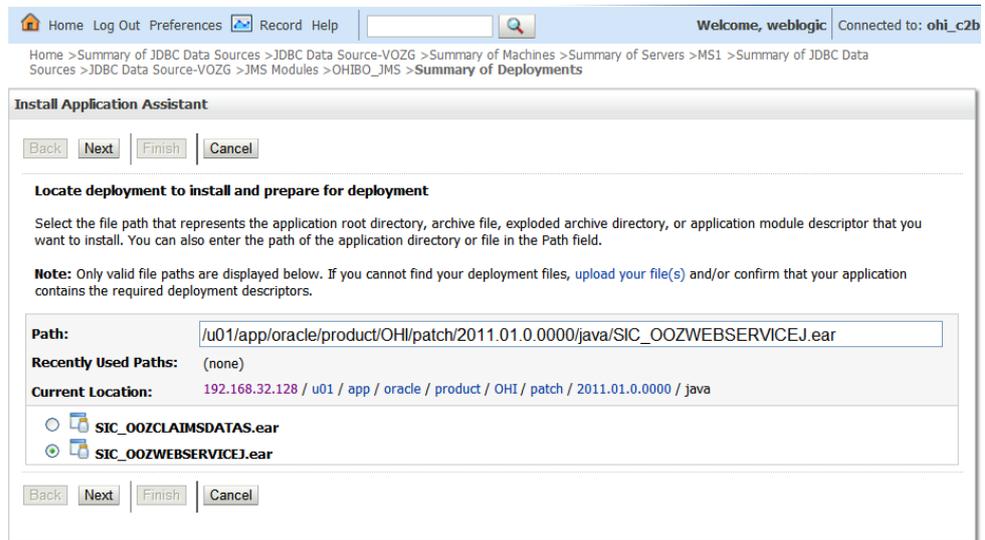
## 5.8. Deploy the application EAR file

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

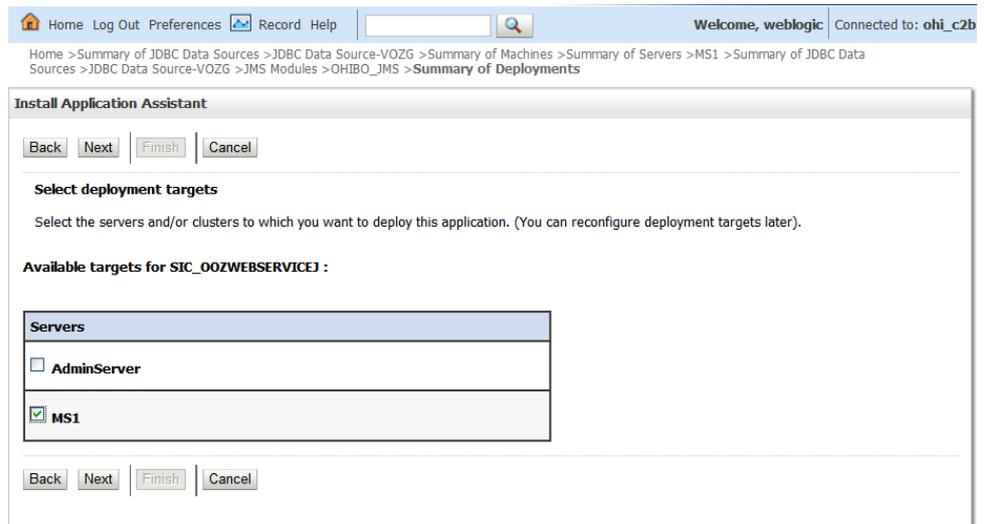
Now you can deploy the EAR file. Weblogic assumes that the EAR file can be found through the file system of the Admin Server host.

In our example, we will deploy the SIC\_OOZWEBSERVICEJ.EAR through the GUI.

Go to 'deployments' and 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?

---

## 6. Administration

### 6.1. Logging

The Connect to Back Office web services are set up for logging through log4j. The use of log4j is a widespread open source solution. More information can be found on

- <http://logging.apache.org/log4j>
- <http://en.wikipedia.org/wiki/Log4j>

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

- Set up directory structure
- Create log4j configuration file
- Configure WebLogic to use log4j

#### 6.1.1. 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: log4j.properties, output files for web services 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 log4j configuration and output files.

#### 6.1.2. Create log4j configuration file

---

Below is an example of a configuration file.

```
#
# 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 too 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/admin/SIC_OOZ_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.ohi.c2b.sicoozwebservicej=INFO,F

```

Note that for each web service you need to specify a logger entry like:

```
log4j.logger.com.oracle.ohi.c2b.sicoozwebservicej=INFO,F
```

In the above example, the logger name is the package associated with the web service. In this case *com.oracle.ohi.c2b.sicoozwebservicej* was used for the SIC\_OOZ\_WEBSERVICEJ web service

Finally, ensure that the file permissions of `log4j.properties` restrict access to authorised users only. In our test set up, we applied 'chmod 640' on `log4j.properties`.

### 6.1.3. Configure WebLogic to use log4j

---

Start the WebLogic administration console.

- Select the managed server for which you want to set up logging.
- Select Logging/Advanced
- Change the Logging Implementation from JDK to Log4j
- Select Server Start/ Arguments and add  
-Dlog4j.configuration=file:/u01/app/oracle/product/OHI/admin/log4j.properties
- Copy `$WL_HOME/server/lib/wllog4.jar` to `$DOMAIN/lib`  
(`$WL_HOME/./user_projects/domains/<domain>/lib`)
- Download `log4j.jar` (for example from <http://logging.apache.org/log4j/>) to `$DOMAIN/lib`
- Stop and start the managed server
- Check the managed server log to verify that log4j is being used.

Verify the output and log file which were created during startup of the managed server. Warnings like this:

```
log4j: WARN no appenders could be found for logger XYZ
```

indicate that `log4j.properties` is not properly configured. Possible errors are:

- `log4j.properties` cannot be located. Verify that the location specified in `log4j.configuration` matches with the location of the actual file.
- The logger for the service XYZ has not been configured.

Once `log4j.properties` is configured, these messages will go away and you will see messages in the designated log file.

## 6.2. Start WLS Node manager

```
. ozg_init.env $OZG_ORATAB_FRS11G1  
cd $WL_HOME/server/bin  
.  
. setWLSEnv.sh  
./startNodeManager.sh
```

## 6.3. Start WLS Configuration Wizard

```
. ozg_init.env $OZG_ORATAB_FRS11G1  
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
```

---

## 7. Appendix A - Installation of SIC\_OOZWEBSERVICES

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 Ch 3.
- Deploy the SIC\_OOZWEBSERVICES.ear, target to a managed server and start the web service

---

## 8. Appendix B - Functional Testing

### 8.1. Test SIC\_OOZWEBSERVICEJ

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

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.

#### 8.1.1. 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.

For more information and to download the tool go to <http://www.hermesjms.com/confluence/display/HJMS/Home>

Note that prior to installing HermesJMS you must create an environment variable JAVA\_HOME which points to a JVM (version 1.5 or higher).

#### 8.1.2. Create wfullclient.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 wfullclient.jar to the computer running HermesJMS.

#### 8.1.3. Configure a provider in HermesJMS

---

Start HermesJMS and open the preferences dialog.

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

Add the wfullclient.jar file to the library.

#### 8.1.4. Configure a context in HermesJMS

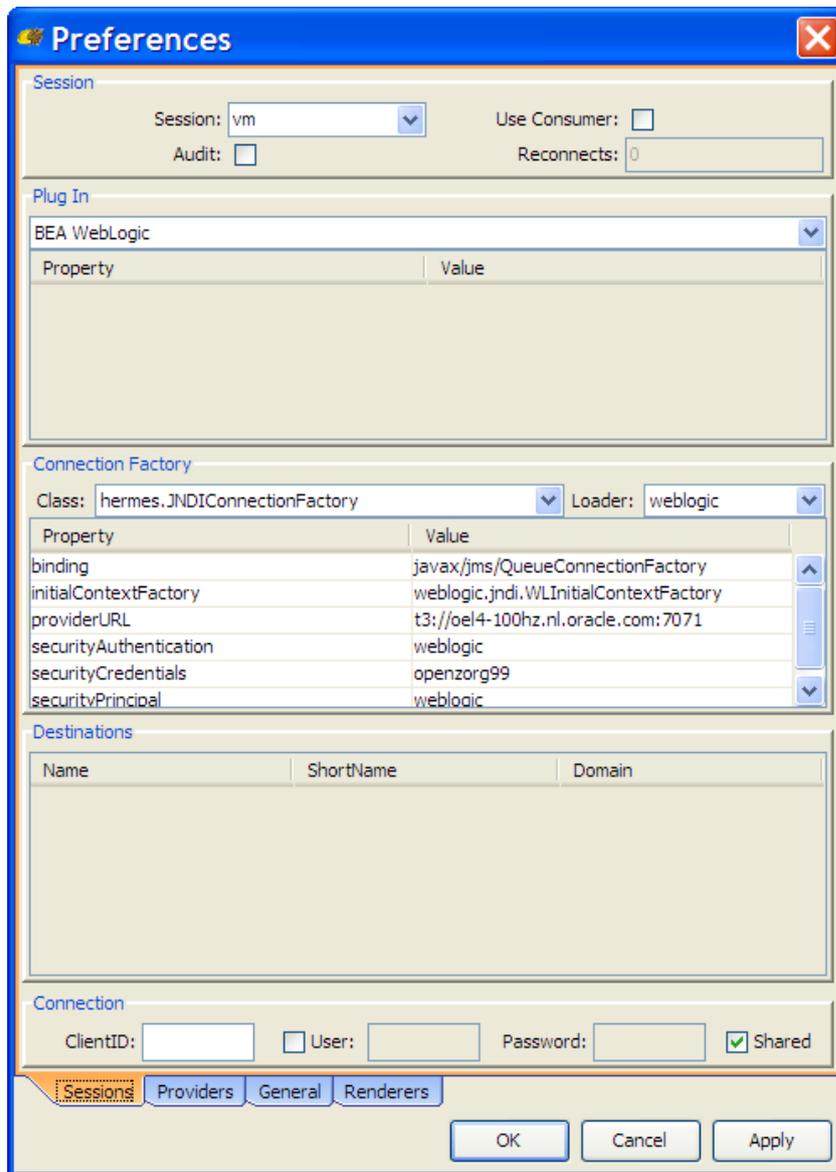
---

Open the 'Sessions' tab in the preferences dialog.

- Enter a session name, eg. 'vm' 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>, eg. weblogic
securityCredentials	<administrator_password>, eg. openzorg99

At this point the screen should look like this:



### 8.1.5. Discover the JMS queues for SIC\_OOZWEBSERVICEJ

---

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

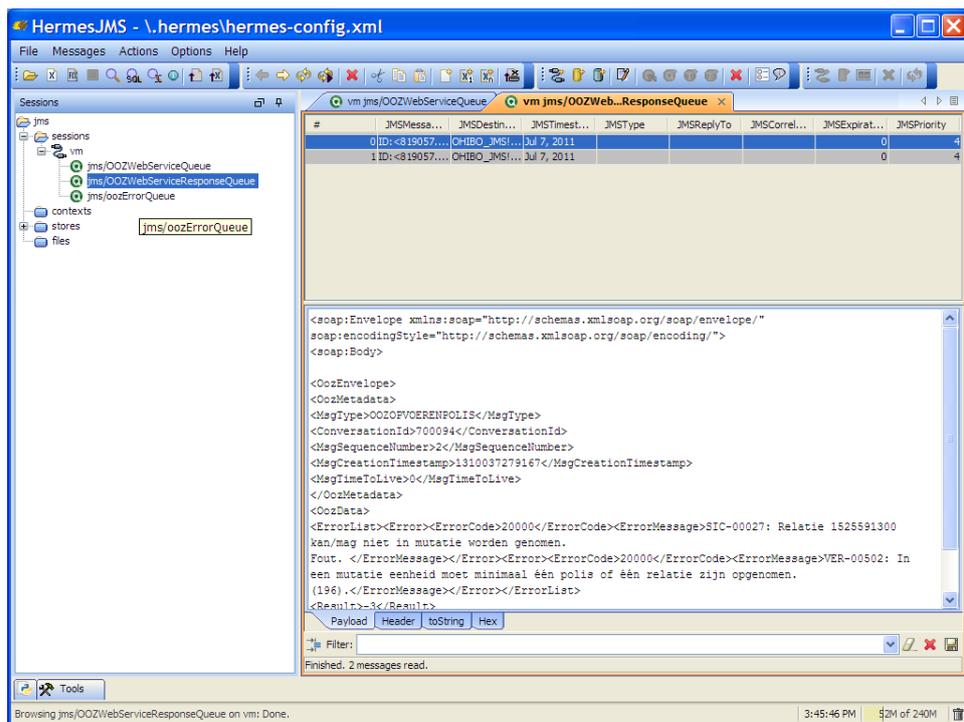
At this stage you will see the 3 queues which were set up for SIC\_OOZWEBSERVICEJ.

### 8.1.6. Send test message to jms/OOZWebServiceQueue

---

- Create an XML file containing a test message. You may use the sample message for this.
- 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:



## 8.2. 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

### 8.2.1. 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:

Deployments

Install Update Delete Start Stop Showing 1 to 3 of 3 Previous Next

Name	State	Health	Type	Deployment Order
SIC_OOZCLAIMSDATAS	Active	OK	Enterprise Application	100
SIC_OOZWEBSERVICEJ	Active	OK	Enterprise Application	100
SIC_OOZWEBSERVICES	Active	OK	Enterprise Application	100
Modules				
SIC_OOZWEBSERVICES			Web Application	
EJBs				
None to display				
Web Services				
SicOozwebserviceS			Web Service	

Install Update Delete Start Stop Showing 1 to 3 of 3 Previous Next

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

Home > Summary of Deployments > SicOozwebserviceS

Settings for SicOozwebserviceS

Overview Configuration Security **Testing** Monitoring

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

Deployment Tests

Showing 1 to 1 of 1 Previous Next

Name	Test Point	Comments
SicOozwebserviceS		Test points for this WebService module.
/SIC_OOZWEBSERVICES/SicOozwebserviceS	?WSDL	WSDL page on server MS1

Showing 1 to 1 of 1 Previous Next

Click on WSDL to generate the WSDL file.

Save as SicOozWebserviceS.wsdl.

## 8.2.2. Install SoapUI tool

soapUI is an open source tool for functional testing. You can download it from [www.soapui.org](http://www.soapui.org).

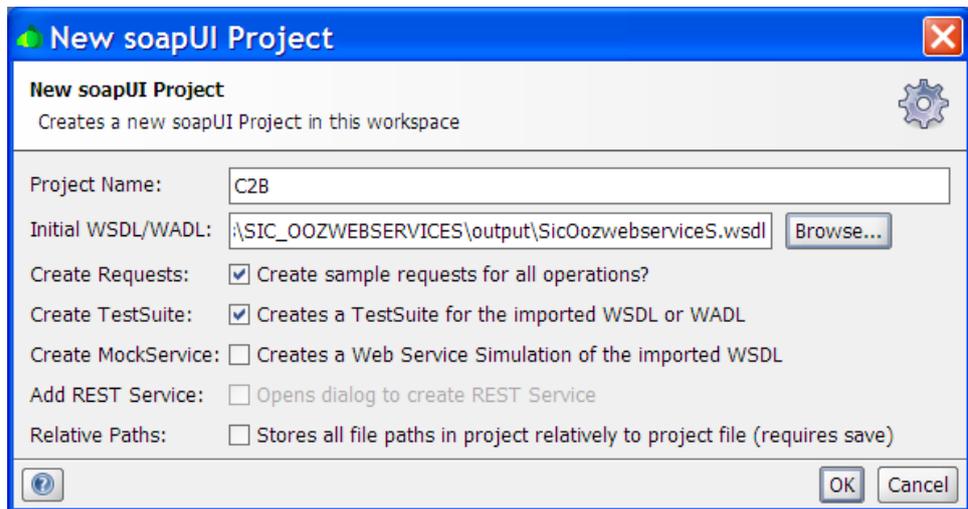
We have used version 3.5.1.

## 8.2.3. Create soapUI project

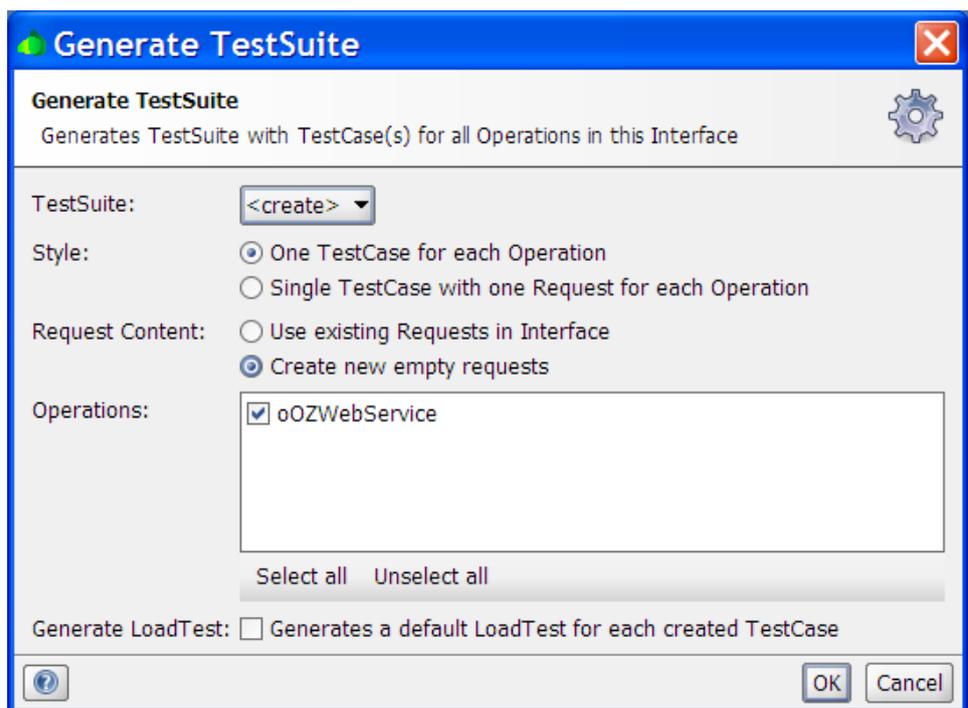
Start the soapUI tool, ignore timeout warnings.

Perform the following steps:

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



For the generation of the test suite, ensure to create empty requests:



Click OK to generate the project and test suite.

#### 8.2.4. 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 and right-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:

The request (in the left window of the request editor) will look like this:

```
<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>
```

### 8.2.5. 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.

### 8.3. Sample message

```
<?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>
```