

Oracle Health Insurance Back Office

Installation, Configuration and DBA Manual

Version 3.80

Part number: G24367-01

January 08, 2025

Copyright © 2011, 2025, 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	3.19	- Added change history paragraph. - Added links to CPAN download pages for required Perl Modules in Appendix D - Installing Required Perl Modules. - Changed references to configuration templates from Beehive Online - Removed Appendix G – Configuration Oracle Application Server 10G (OHI Back Office)
10.12.3.0.0	3.20	1. Added a paragraph about adjusting INITRANS and/or PCTFREE settings to prevent unnecessary lock messages.
10.13.1.0.0	3.21	2. Added note that WLS_FORMS managed server must be up to implement configuration settings.
10.13.1.0.0	3.22	- Updated description and references to Oracle Fusion Middleware software to 11g Release 2. - Changed forms settings description to support language dependent keyboard mapping file. - Added description of formParam1 and formParam2 to pass a startup form and keys on the URL for an OHI user interface session. - Added definition of workingDirectory property for formsweb.cfg configuration. Changed location of .env file from \$OZG_ADMIN to \$OZG_BASE in examples. - Added paragraph about customizing your toolbar icons and signing your own .jar file for toolbar icons.
10.13.1.1.0	3.23	Added an additional chmod command for Reports configuration as needed for 11gR2 release.
10.13.2.0.0	3.24	- Enhanced the description about the formParam1 and formParam2 parameters. - Added a paragraph about batch scheduler performance and parallelism for using parallel execution of the serial part of a batch request
10.13.2.1.0	3.25	- Added description of new Oracle Scheduler jobs that ‘assist’ the background processing as implemented by the OHI Back Office batch scheduler. - Added instructions about adding application server IP addresses. - Added environment cloning tips. - Web service consumers are now referenced using this correct term. - Made clearer which system parameter specifies maximum number of parallel batch processes.
10.13.3.0.0	3.26	- Added paragraphs about SQL Plan Management (SPM) as a first implementation of this database functionality is supported with this OHI Back Office release. - Also an Appendix that addresses setting up an OEM Metric for SPM space management is added - Added descriptions for the newly introduced background Oracle Scheduler Jobs in the paragraph about the Batch Scheduler - Specified maximum length of 10 for parameter p_mdi_window_label - Adapted references to database software from DB11G2 to DB11204 as for OHI Back Office 10.13.3 database version 11.2.0.4 is the required version
10.13.3.0.0	3.27	- The documentation has been updated regarding the support of SSO for the OHI Back Office user interface users. - The SSO appendix regarding this topic has been completely revised. - The Reports Server batch account is no longer described using its former default name but is named referencing where it is meant for.
10.14.1.0.0	3.27	- On a few locations information has been added regarding the new database account (typically OHI_DPS_USER) that is used for execution of dynamic pl/sql. This as a result of a security improvement. - The Appendix for installing the Perl modules has been adapted as for parallel forms compilation some additional modules are required. - The command for setting NLS_LENGTH_SEMANTICS in the server parameter file has slightly been adjusted (scope=spfile instead of scope=both, because of a limitation in the database).
10.14.2.0.0	3.28	- Only some minor textual corrections in the SQL Plan Management paragraphs. - Corrected the <Directory> entry in forms.conf to allow for a dynamic number of digits in the last number of the release version string (?* instead of ????) for viewing the release information. - Added notes to the Globalization paragraph and the NLS_LANG setting paragraph which describes generic character set settings and behaviour regarding this topic. The client character setting used for Forms is discussed over here. - An Appendix is added which describes how to migrate an OHI 10.14.2 database from a non Unicode character set (for OHI typically WE8ISO8859P15) to AL32UTF8.

Release	Version	Changes
10.14.2.1.0	3.29	- Added OHI Data Mart instructions regarding migration to AL32UTF8. - Incorporated changes in AL32UTF8 migration Appendix based on latest experiences.
10.15.1.0	3.30	- Added query to SQL Plan management Appendix to monitor space used by the SPM repository. - Added additional stats gathering instructions in AL32UTF8 Appendix for virtual columns related to function based indexes. - Removed references to Reports and Discoverer - Revised resource management (12c) - Updated 'batch scheduler settings' - Changed database version to 12c and adjusted for the use of a pluggable database - Worked through impact of less files in \$OZG_ADMIN (\$OZG_BASE/utls, OHIPATCH, etc.) - Corrected spelling errors
10.15.3.0.0	3.31	Replace OHI Business Intelligence by OHI Data Marts.
10.16.1.0.0	3.32	Only some small typing errors were changed.
10.16.2.0.0	3.33	- Many changes to describe the configuration of WebLogic and Forms 12c (12.2.1.1.0) - On several locations information has been added regarding the new database account OHI_VIEW_OWNER that is used for the implementation of Virtual Private Database. Removed reference to WLS_REPORTS.
10.16.2.2.2	3.34	- Some minor textual formatting. - Added additional settings in forms.conf for better readability of .log and .out files in Internet Explorer. - Small textual formatting correction on wlst example - Added instance parameter _common_data_view_enabled setting - Added FORMS_LOV_... settings - Described starting user interface outside the browser using javaws - Added attention note to remove serverArgs from formsweb.cfg
10.17.1.0.0	3.35	- Added FORMS_DATETIME_LOCAL_TZ setting. - Added Oracle error when mkstore incompatibility problem occurs - Changed all references to WebLogic and Forms from 12.2.1.1 to 12.2.1.2 including relevant hyperlinks as this version is supported starting with this release. But the Forms 12.2.1.1 release can be used until OHI release 10.17.2.0.0 is installed.
10.17.2.0.0	3.36	- Forms 12.2.1.1 release desupported. - Some minor adjustments in FMW installation instructions. - Changed description of database configuration regarding the use of multiple pluggable databases in a container database.
10.18.1.0.0	3.37	- Adapted because of required database release 12.2.0.1.0 starting with this OHI release. Instructions for installing DBD::Oracle slightly changed. Text is adapted for changes related to the SPM Evolve job, instance parameter settings and AWR. Also the instruction regarding setting NLS_LENGTH_SEMANCTICS has changed. - Missing instructions for creating additional service definitions added.
10.18.1.2.0	3.38	Changed all references to WebLogic and Forms from 12.2.1.2 to 12.2.1.3 including relevant hyperlinks as this version is supported starting with this release. But the Forms 12.2.1.2 release can be used until OHI release 10.18.2.0.0 is installed.
10.18.1.3.0	3.39	UTF8 instead of AL32UTF8 for Forms in case of character conversion issues. - Added description of existing accessibility option
10.18.1.4.0	3.40	Corrected some contradicting information regarding terminal resource files (key mapping description files specified through the <code>term</code> parameter in formsweb.cfg as described in the formsweb.cfg configuration paragraph).
10.18.2.0.0	3.41	Removed C2B references when present in still current text.
10.18.2.1.0	3.42	- Added remark about Oracle Managed Java Client for Client Tier paragraph. - Changed the MEMORY_TARGET and MEMORY_MAX_TARGET advise.
10.18.2.2.0	3.43	- Added chapter regarding Database Vault installation & activation - Batch scheduler description was outdated, where necessary this has been updated to better reflect the current functionality
10.18.2.3.0	3.44	- Removed some deprecated files from \$OZG_BASE/conf - Specified which character set is used hardcoded in the installation menu - Small change in Database Vault activation description
10.19.1.0.0	3.45	No changes. Republished with different part nr.
10.19.1.2.0	3.46	Added instruction for creating a user for accessing standard OHI database queues with a JMS text payload.
10.19.1.4.0	3.47	- Removed references to Designer repository based information. - Added 'deprecated' to relevant paragraphs regarding deprecated service consumer supporting components.
10.19.2.0.0	3.48	Deprecated service consumer paragraphs are now 'expired'.
10.20.1.0.0	3.49	Added reference to the release installation manual for when seeking information regarding compression.

Release	Version	Changes
10.20.2.0.0	3.50	Corrected typing error
10.20.3.0.0	3.51	- Adapted for supporting database 19c - Adapted for supporting FMW 12.2.1.4 (optionally used in this release) - Multiple textual improvements and correction of typing errors - Removal of some outdated paragraphs
10.20.4.0.0	3.52	- Instructions for DB Vault activation adapted. - Allowed schema maintenance described
10.20.5.0.0	3.53	Appendix for Database Vault activation has been revised and extended
10.20.6.0.0	3.54	Appendix for Database Vault activation has been adapted and extended.
10.20.7.0.0	3.55	Reorganisation of Database Vault chapter into two chapters, a description followed by instructions. Also text is adapted to optional use specific functionality.
10.20.8.0.0	3.56	- Some spelling mistakes have been corrected. - Advise for using OPTIMIZER_USE_SQL_PLAN_BASELINES setting has been made more explicitly an advise. - Updated Database Vault paragraph regarding OHI owner access - Updated instructions for setting up JMS queue in WebLogic, Appendix G
10.21.1.0.0	3.57	No changes, republished with new part number.
10.21.2.0.0	3.58	- Additional command dbms_macadm.add_auth_to_realm is needed when Database Vault is enabled but OHI Realm is not created. - Altered hierarchicall setup with MANAGER_ROL - Added a paragraph about activating the PL/SQL hierarchical profiler for batch processes - Added a paragraph to use JFC and JMC for monitoring WebLogic JVM processes
10.21.6.0.0	3.59	Removed obsolete text about compression from par. 3.3.4
10.21.7.0.0	3.60	Rewrote Appendix C (SSO).
10.22.1.0.0	3.61	No changes; republished with new part number.
10.22.2.0.0	3.62	Added information regarding using a service for running OHI background jobs.
10.22.3.0.0	3.63	Updated Maintenance jobs in OHI background jobs.
10.22.4.0.0	3.64	Removed description of Java PlugIn configuration (JPI) because Microsoft support for Internet Explorer is ending.
10.22.6.0.0	3.65	Corrected some typos
10.22.7.0.0	3.66	- Corrected numbering in paragraph 4.7 - Removed support for WLS12213/FRS12213 - Certification for Linux 8: update of perl module versions and installation - Described issue with Pro*C compilation - Removed obsolete RAC descriptions and NFS instructions - Removed references to obsolete ozg_main.sh
10.22.8.0.0	3.67	Added Access to OHIJET Application in Appendix C.
10.23.1.0.0	3.68	No changes, republished with a new part number.
10.23.2.0.0	3.69	Described issue with Pro*C compilation on Linux 8.7.
10.23.3.0.0	3.70	Memory parameters in paragraph “Database Instance Parameters”
10.23.5.0.0	3.71	Restructured.localization support aspects into paragraph 3.1 Localization Support
10.23.6.0.0	3.72	- Adapted for supporting FMW 12.2.1.19 (optionally used in this release) - Added 4 additional optional settings for formsweb.cfg in paragraph 4.3.1
10.23.7.0.0	3.73	Log and out directories on shared storage are not required for RAC, only when using multiple Middle Tier servers. Creating directories for the new user and Configuring Directories. Added FORMS_LOGIN_DIALOG_BLANK for SSO.
10.23.8.0.0	3.74	Expanded paragraph on additional service definitions
10.24.1.0.0	3.75	- Republished with new part number. - Rewrote CustomizingToolbarIcons and Signing jar file - Desupport of SIMPLE certificates in INSTALLATION & CONFIGURATION OF SSO IN OHI BACK OFFICE
10.24.2.0.0	3.76	Rewrote Creating OS Accounts & Authorization for separate wallets.
10.24.3.0.0	3.77	- Support for SSO without database accounts for end users in INSTALLATION & CONFIGURATION OF SSO IN OHI BACK OFFICE Removed details about individual RADs - Clarified creation of <environment>.env files in Configure Environment variables in *.env files - Removed support for FRS 12.2.1.4.0
10.24.4.0.0	3.78	New background job purge_uoc
10.24.6.0.0	3.79	Added paragraph Getting information after “ORA-12801”
10.25.1.0.0	3.80	- Republished with new part number. - Documented which OHI Back Office parameters are used to determine the retention period for a number of purge jobs. - Increased PGA_AGGREGATE_LIMIT to 12 GB

CONTENTS

1.	Introduction	9
2.	Components and Requirements	11
2.1.	Architecture	11
2.2.	Application Software Installation	13
2.3.	Related Publications	13
2.4.	System Requirements	14
2.5.	Disk Space and Memory Requirements	16
2.6.	Restrictions	17
2.7.	Application Software and Templates	18
3.	Setting up the Environment	19
3.1.	Localization Support	19
3.2.	Performing the Pre-Installation Tasks	23
3.3.	Setting up the Environment Variables	31
3.4.	Setting up the OHI Directory Structure	35
3.5.	Installation of OHI Back Office Application Logos	38
3.6.	Setting up Universal Oracle OS Scripts	38
3.7.	Install additional Perl modules	40
3.8.	Customizing Toolbar Icons	40
4.	Configuration Oracle Fusion Middleware Software for OHI Back Office	43
4.1.	Installation and Initial Configuration of Required Software	43
4.2.	Configure Oracle HTTP Server	53
4.3.	Configure Forms Server	56
4.4.	Configure Security and Manage Startup/ Shutdown	64
4.5.	Improving Startup Time of WebLogic	67
4.6.	Adding Managed Servers to 'cluster_forms'	68
4.7.	Understanding batch & background processing	70
5.	Installation OHI Back Office Application Software	75
5.1.	Creating OS Accounts & Authorization	75
5.2.	Installation of the Application	82
5.3.	Configure authorized Application Servers	82
5.4.	Compile and Check Application Software	83
5.5.	Configuring Directories	85
5.6.	Registering batch scheduler Account	85
6.	Completing the Installation	86
6.1.	Synonyms for Database links	86
6.2.	Starting OHI Back Office Batch Scheduler	86
6.3.	Checking the Installation	86

6.4.	Creating Backups	86
6.5.	Activating Jobs	87
7.	Oracle Administration Related to OHI Back Office	88
7.1.	Backup & Recovery	88
7.2.	Startup and Shutdown	89
7.3.	Performance Tuning and Monitoring	89
7.4.	Using SQL Plan Management	91
7.5.	Installation, Configuration, Upgrade, Migration and Version Control	96
7.6.	Access Control and Security Privileges	96
7.7.	Space and Storage Management	97
7.8.	License Control	97
7.9.	Networking	97
8.	OHI Back Office Management	99
8.1.	System Management/DBA	99
8.2.	Creating New Application Users	105
8.3.	Installation OHI (Patch) Releases	107
8.4.	Cloning an OHI Back Office environment	107
8.5.	Allowed Schema Maintenance	109
9.	References for Active Management of OHI Back Office	110
9.1.	Performance - General	110
9.2.	Performance & Management of the OHI Batch Scheduler	113
9.3.	Database Performance	115
9.4.	Performance Application Server	137
9.5.	Getting information after “ORA-12801”	138
9.6.	Prevent Unnecessary Lock Messages	139
9.7.	Anticipate Preventive Resource Messages	140
10.	OHI Database Vault - Description	142
10.1.	Database Vault for OHI – Organisational impact	142
10.2.	Realm composition	144
10.3.	Realm access	145
10.4.	Account management	147
10.5.	OHI technical application management	147
10.6.	Supported situations	147
10.7.	Effective Use of the OHI specific Realm	149
11.	Installing and Configuring Oracle Database Vault for OHI Back Office and/or OHI Data Marts	151
11.1.	Implement Database Vault	151
11.2.	Activate the OHI Realm	158
11.3.	Updating, Deactivating and removing the OHI specific realm	162
11.4.	Database Vault and DB Maintenance Tasks	165

12. Uninstalling OHI Back Office	166
Appendix A – Configuration of Multiple batch Schedulers	167
Starting / Stopping the Batch Scheduler	167
Process Load Balancing Mechanism	168
Appendix B - Installation & Configuration of OHI in a RAC Environment	170
Introduction	170
Architecture	170
Batch Processing	170
Failover	171
Management	171
Appendix C - Installation & Configuration of SSO in OHI Back Office	172
Introduction	172
Related Publications	173
Architecture	174
Security	175
Single Point of Failure	176
SSO System Administration	176
Software Requirements	177
Licenses	177
Installation Instructions	177
Appendix D - Installing Required Perl Modules	202
Introduction	202
Manual Installation Procedure	202
More Automated Installation Procedure	204
Specific Installation Instructions for DBD::Oracle	204
Term::ReadKey Issue	206
Appendix E - Installation Checklist	207
Appendix F – Setting up OEM Metric for SQL Plan Management	210
Appendix G – Providing access to JMS queues	215
Database access	215
WebLogic access	216

1. INTRODUCTION

PURPOSE

This document describes the installation and configuration information for Oracle Health Insurance Back Office. This is the working name for a set of Oracle Health Insurance products as provided by Oracle for implementing and executing the core ‘Back Office’ processes of an healthcare (insurance) payer.

TARGET AUDIENCE

This document is intended for database and application server managers and others responsible for installation of Oracle products. Even though various command samples are provided in this document, this document does not at all intend to offer a course in Oracle technology stack management.

TYPOGRAPHIC CONVENTIONS

Given that Linux commands and file names case sensitive, conventions in this document may differ from the ones included in other Oracle product documentation.

COMMAND SYNTAX

Command syntax is represented in font `monospace`. The following conventions apply to command syntax:

`monospace`

Monospace type indicates OS commands, directory names, user names, path names, and file names.

brackets []

Words enclosed by brackets indicate keys (e.g., Key [Return]). Attention that brackets have a different meaning when used in command syntax.

italics

Italics indicates a variable, including variable parts of the file names. It is also used for emphasizing.

UPPERCASE

Uppercase letters indicate Structured Query Language (SQL) reserved words, initialization parameters and environment variables.

backslash \

Each backslash indicates a command that is too long to fit on one line:

```
dd if=/dev/rdsd/c0t1d0s6 of=/dev/rst0 bs=10b \
count=10000
```

braces { }

Braces indicate mandatory items: `.DEFINE {macro1}`

brackets []

Brackets indicate optional items: `cvtcrt termname [outfile]`

Attention that brackets have a different meaning when used in ordinary text.

ellipses ...

Ellipses indicate a random number of similar items:

```
CHKVAL fieldname value1 value2 ... valueN
```

italics

Italics indicates a variable. Replace the variable by value:

library_name

vertical bar |

The vertical bar allows choosing either braces or brackets:

`SIZE filesize [K|M]`

NAMING PRODUCTS

Whenever Oracle Health Insurance (OHI) Back Office is mentioned in this document, the *entire suite* is intended (containing products like Oracle Insurance Claims Management for Health, Oracle Insurance Policy Management for Health and others).

RELATED DOCUMENTATION



Oracle Health Insurance Certification on MOS (logon to MOS, choose tab ‘Certifications’, specify product name ‘Oracle Health Insurance Back Office’ and the relevant version)



Reading, Writing and Authorizing Oracle Health Insurance Application Files (docs.oracle.com)



Oracle Health Insurance Release Installation (docs.oracle.com)

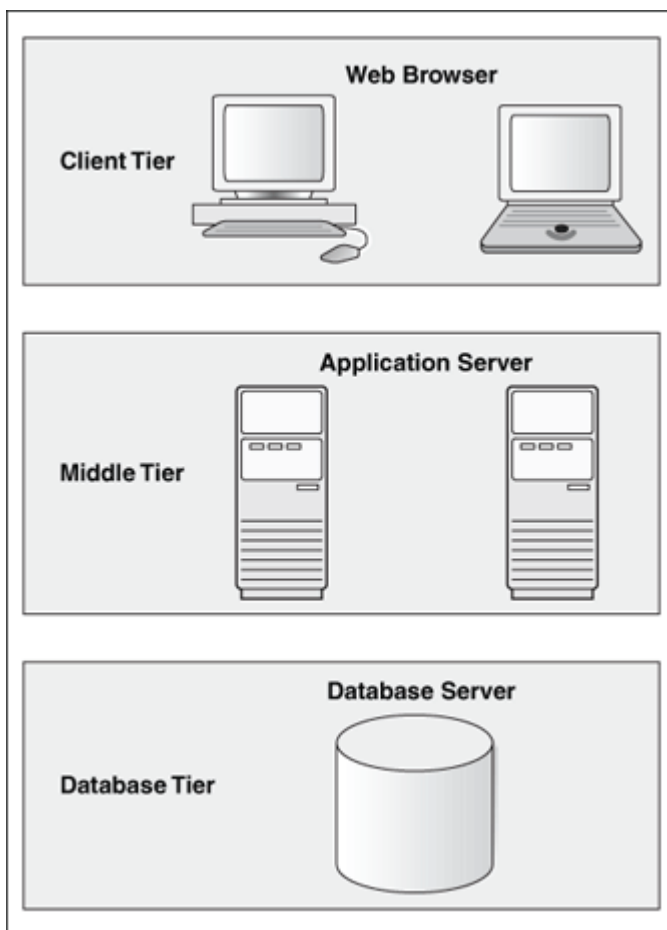
2. COMPONENTS AND REQUIREMENTS

In order to perform a quick and successful installation, a number of requirements have to be met regarding Oracle Software in the system. This chapter describes the requirements for the installation of OHI Back Office, an integrated suite of products. Check to ensure that the system meets these requirements before starting the installation process.

2.1. ARCHITECTURE

The application OHI Back Office is implemented in the so-called *web-architecture*.

This architecture is based on 3 tiers:



Desktop or Client Tier

The client tier, at the top of the image, may contain one of the following two configurations:

- A Web browser where the application is launched through Java Web Start,
- The Java Stand-Alone Launcher (JSAL)

Both require Java through an installation of Java Development Kit (JDK) or Java Runtime Engine (JRE).

Middle Tier

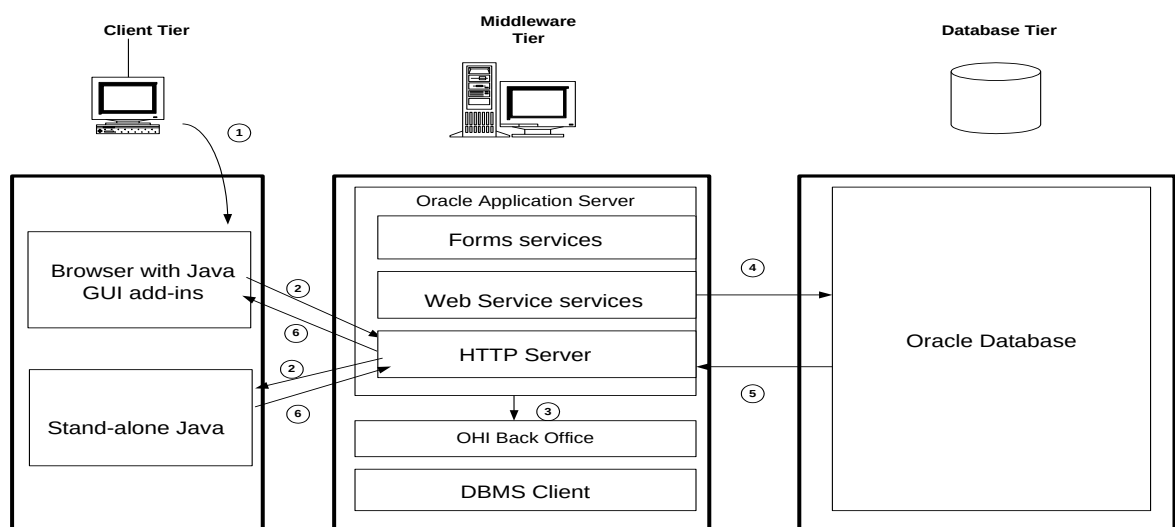
The Application Server (AS), Database Client and the OHI Back Office application software run on this tier.

Database Tier

The Database Server (RDBMS), which contains the OHI Back Office data as well as business logic and business rules, runs on this tier.



Attention: This refers to *logical* tiers; the structure does not entail the *physical* implementation of the tiers; the physical aspects and any other relevant details can be implemented on 1 or multiple servers.



Client Tier.

Mode	Browser	Java component	Remarks
Applet in browser (traditional)	Required throughout session	JRE + JPI (java browser plugin)	Not supported in all browsers; No longer supported for OHI
Embedded JNLP	Required throughout session	JRE + JPI (java browser plugin)	Not supported in all browsers; not supported for OHI
Java Web Start	Only for download of up to date JNLP file	JDK or JRE; no JPI (no browser plugin)	All browsers; with Single Sign-On, no Single Sign-Off; supported for OHI
Forms Standalone Launcher	Not used	JDK or JRE; no JPI (no browser plugin)	All browsers; no Single Sign-On Requires distribution of command file; not supported for OHI

In the last few years, most browsers have discontinued support for the JPI (Java PlugIn) until only Internet Explorer remained. With the end of Microsoft support for Internet Explorer on June 25, 2022, the JPI is no longer supported for Forms, and therefore no longer supported for OHI. Use Java Web Start instead for running the OHI Back Office user interface.

Forms Standalone Launcher (FSAL) is a supported option for Oracle Forms, and should work for OHI, but OHI Development have only very limited experience with it and offer no active support of FSAL for OHI.

This architecture results in the following software layers:

Software layers
OHI Back Office application software
Oracle Application Server software
Oracle Database Client software
Oracle Database Server software
Operating System software

The Application Server, Database Client and Database Server software are also referred to as *Oracle system software*.

Installation of this Oracle system software is required in order to run the OHI Back Office application software.

It is very convenient to install all related Oracle software such as Application Server, Database Client and Database Server under the same operating system user.



Attention: If OHI Back Office is used in a RAC environment (Real Application Clusters), then a number of layers will be added on the database servers. See the RAC documentation for details.

2.2. APPLICATION SOFTWARE INSTALLATION

OHI Back Office has to be installed in its' own home directory, separated from the ORACLE_HOME directories which contain the Oracle Database and/or Oracle Application Server software.

Installation of OHI in the same home directory as Oracle system software is not supported.

2.3. RELATED PUBLICATIONS

Additional information regarding Oracle Database and Oracle Application Server software is available in the following documentation for the relevant platform:



Oracle Database Installation Guide



Oracle Fusion Middleware Documentation Library

2.3.1. RAC

Additional information regarding Oracle Real Application Clusters software is available in the following documentation for the relevant platform:



Oracle Clusterware and Oracle Real Application Clusters Installation and Configuration Guide



Oracle Clusterware and Oracle Real Application Clusters Administration and Deployment Guide

2.3.2. Single Sign On

Single Sign On (SSO) is supported with OHI Back Office, using the standard functionality of the Oracle system software as implemented in Oracle Access Manager (OAM).

-  [Middleware, Developer Tools, Forms, Release 12.2.1.4, Working With Oracle Forms, Using Forms Services with Oracle Access Manager](#)
-  [Oracle Access Management 12.2.1.4.0](#)
-  [Get Started](#)
-  [Administering Oracle Access Management](#)
-  [Oracle Access Management 12.2.1.4.0. - Install](#)
-  [Tutorial - Installing Oracle Access Management 12c](#)
-  [Tutorial - Configuring OHS WebGate](#)
-  [RCU Requirements for Oracle Databases](#)

2.4. SYSTEM REQUIREMENTS

2.4.1. Certification

OHI Back Office is certified for several platforms and specific Database, Application Server and browser versions.

Before starting the installation, check if the desired product combination is certified for OHI Back Office.

The certification information is available online:

-  [Certification on My Oracle Support](#)

2.4.2. Hardware Requirements

Depending on e.g., the number of users, the usage type of OHI Back Office and the desired performance, a number of requirements regarding CPU, memory, disk space, networking, etc. have to be met.

A specific sizing advice has to be performed in order to determine the numbers for a specific customer environment. Such a sizing advice can be performed by Oracle.

2.4.3. Configuration of Printer Queues

The printer queues (in fact output queues or output commands or scripts) that will be used for OHI Back Office should be configured and tested on your OS.

This step must be performed on the Middleware Tier.

2.4.4. Software Requirements

OHI Back Office uses the following Oracle system software, which should be installed and configured before OHI Back Office is installed:

2.4.4.1. Database Tier

- Oracle Database Server
- Oracle Database Server patch set (Release Update Revisions (RUR) or Release updates (RU)) (if required)
- Oracle Database Server interim patch(es) (if required)

The installation of the Database software is not described in this manual.

2.4.4.2. Middleware Tier

- Oracle Fusion Middleware WebLogic Server
- Oracle Fusion Middleware Forms and Reports Services
- Oracle Fusion Middleware Forms and Reports Services patch set (if required)
- Oracle Fusion Middleware Forms and Reports Services interim patch(es) (if required)
- Oracle Database Client
- Oracle Database Examples (formerly Companion)
- Oracle Database Client patch set (if required)
- Oracle Database Client interim patch(es) (if required)

Installation and configuration of the Middleware Tier software is described in more detail in Chapter 3.



Attention 1: The version of Oracle Database Client software has to be the same as the Oracle Database Server software version.



Attention 2: Of the Oracle Database Client software the DBMS Utilities and the Precompiler software are required.

To this end, install the Administrator software.

After installation, executables proc, imp, exp, sqldr and tkprof have to be available in \$ORACLE_HOME/bin.



Attention 3: Of the Oracle Database Examples software, the Oracle Database Products software has to be installed (for the Pro*C precompiler demo software).

After installation, directory demo has to be available in \$ORACLE_HOME/precomp.



Attention 4: Please consider the fact that if the Examples software is installed on an existing software tree, the required patch set will have to be reinstalled again (because otherwise the older, unpatched utility versions and/or precompiler software will be used).



Attention 5: In case of SSO a description of the software requirements can be found in Appendix C.



Attention 6: Oracle Fusion Middleware 12c (12.2.1) has removed the `mod_plsql` module from the Oracle HTTP Server (OHS). If your custom software uses `mod_plsql` you will have to find an alternative, such as Oracle REST Data Services. See the Oracle Web Tier - Statement of Direction (document ID 1576588.1) on My Oracle Support.

Furthermore, the following additional software is required on the OS of the Application Server:

- ANSI C compiler

Perl

`zip` and `unzip` utilities for Linux

JDK

2.4.4.3. Client Tier

- For Java Web Start mode: Java JRE or JDK

It is best practice to have your IT (Desktop) department supply a Java Client installation – including regular updates!! - on the desktops that are used to start the OHI Back Office user interface.

For more information about license and support please visit:

<https://www.oracle.com/technetwork/java/java-se-support-roadmap.html>

For exact versions of the required software and patch sets, see:



Oracle Health Insurance Certification information on MOS (which references generic Forms services certification regarding this topic)

2.4.5. Operating System Requirements

For more information on the requirements, please refer to the following platform-specific documentation:



Oracle Fusion Middleware Documentation Library



Oracle Database Installation Guide

2.4.6. User Interface Requirements

For detailed requirements regarding OHI Back Office and different screen resolutions and Windows settings, see:



Oracle Health Insurance Certification information on MOS

2.5. DISK SPACE AND MEMORY REQUIREMENTS

The Installation Guides for the required database and middleware products provide a description of the requirements pertaining to Disk Space and Memory.

The following table displays the required disk space for OHI Back Office components for one OHI environment.

These are minimum estimations, not exact calculations.

2.5.1. Database Tier

Product	Required Disk space (MB)
Initial OHI Back Office database (*)	20000
Total	20000

2.5.1.1. Middleware Tier

The components marked with (*) are variable and will develop as the number of end users increases.

Product	Required Disk space (MB)
WLS and Forms	5000
Other (Pro*C, OS shell scripts, tools, perl, Java, configuration files)	250
Batch Scheduler output (*)	2500
Batch Scheduler log files (*)	100
OHI Installation directory including several patch levels	6000
Total	13850

For OHI Back Office no additional memory requirements apply, other than the requirements for the Application and Database Server.

For more information on these requirements, please refer to the following (platform-specific) information:



Oracle Database Installation Guide



Oracle Fusion Middleware Documentation Library

2.6. RESTRICTIONS

2.6.1. National Language Support (NLS)

OHI Back Office can be used with a set of predefined languages. See paragraph [Localization Support](#) for details.

However, one restriction must be considered:

- The *territory* component of the NLS parameter has to have a final status *before* the application goes live and *may not* be changed. If a change is required, however, this can be requested from OHI Back Office; software may be provided which allows for this change to be implemented.

2.7. APPLICATION SOFTWARE AND TEMPLATES

2.7.1. Application Software

OHI application software should and may not be modified, conform SLA's; this would annul the Oracle guarantee.

Reference software is maintained by Oracle and *issued via (patch) releases* and can be recognized by the uppercase prefix and lowercase extension.

Some samples of the reference software include e.g., `OZG_PROC.sh`, `OZG_START_BATCH.sh`, `SYS1107S.sh`, etc.

These files may not be altered.

2.7.2. Templates, Utilities & Sample Scripts

Templates, utilities and sample scripts, however, *can* be altered by customers.

Templates are distributed in releases and are copied to the `$OZG_BASE/conf` directory.

They contain non-generic, often customer-specific settings (e.g. directory paths, host names, etc.) and therefore they cannot be part of the OHI application software.

Templates, utilities and sample scripts can be recognized by the lowercase naming.

Some examples include `ozg_init.env`, `alert_example.pl`, `ohi_svc_osb_configure.sh` and `svl.properties.template`.

2.7.2.1. Change History

When customer-specific changes are made in templates, utilities and scripts, it is advised to always keep a history of changes in a uniform way.

Indicate here when something is altered, who made the change and the reason for it.

Example

```
Change history
=====
Date          Author          Version: Change
-----
13-09-2007 M. Koel          1.2: Use of MS Cabinet files is deleted.
```

3. SETTING UP THE ENVIRONMENT

This chapter deals with the preparations of the environment before installing OHI Back Office, after all requirements listed in chapter 1 have been checked and met.

3.1. LOCALIZATION SUPPORT

Localization is the process of customizing an application for a given culture and locale. It does not modify the functionality of the application.

OHI localization has different aspects that have dependencies on each other:

- The user interface language
- The character set of data
- The time setting (including time zone) of software clients, servers and databases

For standard Oracle products, and therefore for OHI, the variable `NLS_LANG` plays an important role in localization. This variable has three parts:

`<language>_<territory>.<characterset>`. The territory plays an important role in determining the time zone if that is not specified in other ways.

`NLS_LANG` is used by all OCI (Oracle Call Interface API) client applications. Typical OCI clients are the Forms user interface, `SQL*Plus`, `SQL*Loader`, `DataPump` and development tools like `SQL*Developer`.

When `NLS_LANG` is not set, the default database setting is used.

Default values for other language settings are derived from `NLS_LANG`, but can be overruled by setting other variables:

- `NLS_DATE_FORMAT`
- `NLS_NUMERIC_CHARACTERS`

3.1.1. User Interface Language

The user interface language affects the prompts, labels and messages a user sees, not just in OHI Back Office, but also in database clients like `SQL*Plus`.

The following languages are currently supported and can be setup as part of the `NLS` environment variable `NLS_LANG`:

- `AMERICAN`
- `SPANISH`
- `DUTCH`

The OHI Back Office Forms application can be started with a language-specific URL parameter (`&config=[namedconfig]`). This will use a setting for the `NLS_LANG` value that is configured according to the paragraphs [formsweb configuration](#) and [servlet env](#)

Within the OHI Back Office Forms application, each user can set his/her own preferred language, to translate menu options, button labels and messages. This option is available via menu option Edit -> User Settings : Preferred Language. Users should select the same language as is used in the URL parameter.

3.1.2. Character Set

A character set defines which characters can be handled (stored, displayed, etc.) This is especially important for special characters (Cyrillic, Chinese, East-European etc.). Characters that are not included in the current character set will be converted, with unpredictable results.

Character sets can be specified at many different locations:

- Application level
- Operating System level
- Database level

When data travels through these different systems (e.g., when a text is copied from Word, pasted in Forms and then stored in the database, or read by SQL*Loader and stored in the database) it will be converted multiple times, and each conversion has the potential to “mutilate” the data. E.g., diacritic characters like ï, é, ë might not be displayed correctly in your terminal session or stored in the database.

Therefore, it is important to set the correct character set at each level. A complicating factor can be incoming data, (e.g., files that need to be imported) where we have no control over the character set of the data.

Character sets can be subsets of other character sets. “Mutilation” will only occur when data is converted from a superset to a subset, so it is important to use character sets that contain as many characters as possible.

Therefore, OHI requires the database character set to be AL32UTF8. This is a multibyte character set (variable in length, 1 to 4 bytes per character) that is a superset of all other character sets. This ensures that all data can be stored correctly. See paragraph [Unicode Character Set](#).

To make sure data is presented correctly to the database, and to comply with requirements for output data, we also need to specify which character set to use for reading input files and for creating output files.

This functionality is very important to prevent specific characters being lost or misinterpreted when exchanging these files with other parties and/or applications.



Hint: The language, territory and character settings as used by OHIPATCH cannot be influenced. A hardcoded NLS_LANG value is used to make sure installation files are always interpreted in the correct character set in which they are delivered (WE8ISO8859P15) and with the expected territory setting. The language used will always be (American) English.

3.1.2.1. OS locale Setting

Operating Systems also have a set of environment variables named “locale”. In Linux, this includes a variable LANG, with values like *en_US.UTF-8*. If this value does not match the character set of a file to be processed or the setting used for NLS_LANG, mutilation may occur when viewing and/or changing its contents.

For JDBC based clients (as used by the web services as provided with OHI Back Office) the LOCALE setting typically defines the default language, territory and character set. However, when for example SOAP and/or XML messages are passed an ‘encoding’ clause in the message

normally specifies the character set in which the message is offered. OHI Back Office functionality will react on such encoding specifications.



Attention: As a database administrator you will typically set environment variables for the database software by running `ozg_init.env`, resulting in the `DBMS_LANG` value being used to set the `NLS_LANG` value. In that way it determines the language and character set used by utilities as export and `SQL*Plus`.

If you want this to show up in the same language and character set as used for the Linux terminal setting you may want to overrule the `NLS_LANG` after a call to `‘. ozg_init.env’` or make sure your locale is set to comply with `NLS_LANG`.

Suppose you use the value below for `NLS_LANG`:

```
"DUTCH_THE NETHERLANDS.WE8ISO8859P15".
```

When you have your locale set to `LANG=en_US.UTF-8`, diacritic characters (like `ë` in `‘geëxporteerd’`) may show up incorrectly because the terminal uses UTF-8 characters while the utilities use the single byte Latin characters set as specified by `NLS_LANG`.

There are two ways to show these characters correctly:

- change your locale setting to `‘nl_NL.iso885915’`
- after the call to `ozg_init.env` set `NLS_LANG` to:
`"AMERICAN_AMERICA.AL32UTF8".`

3.1.2.2. When and how to prevent ORA-12713 errors in the user interface

When the database uses `AL32UTF8` as character set it can store many more characters than are supported by the character sets like `WE8MSWIN1252`.as may be specified in `NLS` settings for the user interface.

When data is retrieved from the database, it is converted to the client character set (as specified by `NLS_LANG`). Because the client character set in `NLS_LANG` is a subset of the database character set `AL32UTF8`, some special characters, when present in data retrieved from the database, cannot be converted to an equivalent in the client character set. These characters will be replaced by the default ‘unknown’ character for that character set (typically a square or an inverted question mark).

This can occur in any application (Forms, `SQL*Plus`, `SQL*Developer`, `DataPump`) that retrieves data from the database while using a client character set other than `AL32UTF8`. In the Forms user interface, the use of such a ‘limited’ client character set can also result in exceptions throwing the `ORA-12713` message. This happens when such an ‘unknown’ character is present in a string variable passed from `PL/SQL` within the database to `PL/SQL` code within the Forms module (through an assignment). This seems to be a `PL/SQL` limitation which can be circumvented by passing the data from the database to Forms through a query. If you run into such errors it is an option to contact OHI to check whether such a code change is feasible (if it is, a bug can be registered to repair this).

Such ‘unknown’ characters have been stored in the database by a client with a different client character set, that is not a subset of the current client character set.

Some examples:

- loading files, like claims files, which are specified in a different character set
- web service calls with a message encoded in a UTF-8 character set
- database import with a different setting for NLS_LANG during the import session
- SQL*Plus insert script with a different setting for NLS_LANG during the session

NOTE: With the database character set configured as AL32UTF8, the characters should have been stored correctly if the encoding of the passed values and the specified character set did match with each other. It is just the conversion during the retrieval that cannot handle the characters correctly.

If you encounter such conversion issues, consider using the client character set AL32UTF8. Because the client character set is then equal to the database character set, no conversion will occur. This will prevent the conversion issues as well as the ORA-12713 errors in Forms. However, a direct consequence is that all characters supported by AL32UTF8 can be entered with the client. That is because the client character set also determines the characters that can be inserted with the client. This may lead to conversion issues in other places, when the data is retrieved, e.g., when files need to be delivered in a more restrictive character set (for example because an external party cannot support AL32UTF8 based files and requires WE8ISO8859P15 characters). This may result in loss of characters replaced by the default 'unknown' character during that retrieval process. So please be aware of such consequences and carefully determine what your requirements are and what measures you must implement to minimize character set conversion issues.

NOTE 1: Unfortunately, Forms 12.2.1.2 and higher cannot use AL32UTF8 as the client character set (as defined by NLS_LANG) without recompilation of all Forms, Libraries and Menus with that NLS_LANG setting. This is not supported by the Forms compilation options in OHIPATCH. Therefore, we suggest using client character set UTF8 instead for Forms.

NOTE 2: Character set conversions and limitations, which occur when different applications interact and messages are exchanged between these applications, are in itself not a limitation of OHI Back Office. They are a consequence of the real-world situation where not all applications are using one universal character set.

UTF8 is a binary subset of AL32UTF8 (meaning the internal storage of all characters in UTF8 is the same as in AL32UTF8) that will suffice for all European languages.

To specify UTF8 for use in the Forms user interface you can specify an NLS_LANG value in the [servlet.env](#) files as described later in this document. This will overrule any optional setting in [ozg_init.env](#).

An example value for a Dutch environment:

```
NLS_LANG=DUTCH_THE_NETHERLANDS.UTF8
```

Be aware that using UTF8 for the client character set is no guarantee that all UTF-8 characters are displayed correctly in Forms. The fonts used by Forms may not support specific characters. Typically, several symbols are not supported by the default fonts mapped for your java runtime environments. For example, characters like 'Œ' show up in Forms as '□□'. But no ORA-12713 will occur; this is just a font mapping issue in the applet. More information can be found in My Oracle Support in bug 2720366 (which is closed as not being a Forms bug).



MOS Doc ID 158577.1 does contain additional info about NLS_LANG aspects. It also describes that it is best to use the character set that is used on the client platform. This means that the Forms client does normally not need to be set to (AL32)UTF8. It is best to set it to the codepage of your main client platform (probably Windows). MOS document 179133.1 describes the standard code pages used by Windows and the corresponding Oracle character set.

To specify the client character set in other clients than Forms, set NLS_LANG and/or DBMS_LANG in the generic part of `ozg_init.env`. Note NLS_LANG has to be configured on the Middleware Tier *and* the Database Tier.

3.1.3. Time and Time Zone

In OHI Back Office (and OHI Data Marts) all time data is expressed in local time, without a time zone indication. This requires that all parts of the application run in only one time zone.

When time is stored in the database this is done without a time zone specification.

To avoid time mismatches and follow local daylight-saving time (DST) settings:

- the Operating Systems of the database server(s) and the application server(s) must all use the same local time
- the database time zone and the session time zone must be set to the same local time zone

When a session is created using a client (SQL*Plus, SQL* Developer) for which the time zone differs from the time zone used by the OHI application, you may run into time issues.



This is most likely to occur when SQL*Plus, SQL* Developer, etc. are used from a (Windows) laptop that is not in the same region / time zone as the database server.

To prevent such issues please use

```
alter session set time_zone = '<region>/<location>'
```

For example:

```
alter session set time_zone = 'Europe/Amsterdam'
```

Even better: use

```
alter session set time_zone=dbtimezone
```

to align the time zone of the client with the time zone used by the database.

3.2. PERFORMING THE PRE-INSTALLATION TASKS

3.2.1. Creating an OS Account for the OHI Back Office Batch Scheduler

The `batch` account is the OS account that runs the OHI Back Office batch scheduler software.

Log on as `root` under OS and use the operating system user administration utility for creating the `batch` account.

This task must be performed on the Middleware Tier.

The name `batch` is recommended, but this is not mandatory. A different account name can be used as well.

3.2.2. Assigning Default Shells for the Required OS Accounts

For the OS accounts `batch` and `oracle` the default shell in Linux has to be set to `bash`.

3.2.3. Creating an OHI Back Office Database

Before OHI Back Office is installed, a database needs to be created because the OHI Back Office installation will create several database accounts, roles and objects in this database.

For OHI Back Office the multitenant database architecture as introduced with database release 12c is a prerequisite. The installation requires a pluggable database (PDB) within a container database (CDB) for installing the OHI Back Office database objects. Starting with database release 19c, you can create a maximum of 3 PDBs in one CDB without a multi-tenancy license.



When this document references the ‘database’ for OHI Back Office, this refers to the pluggable database unless stated differently.

3.2.3.1. Multiple pluggable databases in one container database

OHI Back Office supports the use of more than one PDB in a CDB. The main advantage is that administration and resource usage is more efficient as they share the same common base.

Several OHI Back Office-prescribed initialization parameters can also be set at PDB level. When using more than one PDB, it is up to the administrator to decide at which level these initialization parameters will be set. The installation software will only check the value of the parameter regardless of the level that it is set at (CDB\$ROOT or PDB).

When changing a parameter value at PDB level it will be available in all existing or new sessions, depending on the parameter involved. When resetting a parameter at PDB level (using `alter system reset [...]`) the parameter defaults to the value set at CDB level or the Oracle default. In that case a restart of the PDB is needed to make the value available.

Overall memory requirements are typically defined at CDB level, so the number of pluggable databases that can be plugged in will be limited by the available resources at CDB level.

3.2.3.2. Unicode Character Set

All customers must use the AL32UTF8 character for the (pluggable OHI) database. This means the container database needs to be created with this character set. Choose AL16UTF16 as the national character set (for container and pluggable database).

Although the application uses CHAR for the NLS_LENGTH_SEMANTICS setting per session, the database setting can and should remain on BYTE to prevent potential issues with database patches.

After creating the database, **do not set** NLS_LENGTH_SEMANTICS to CHAR. Leave it at BYTE for the root container and the OHI Pluggable Database. See also My Oracle Support note 144808.1 for more information about this setting.

The OHI Back Office application automatically sets the NLS_LENGTH_SEMANTICS to CHAR for OHI related sessions when these sessions log on to the database. This is done by an OHI specific system logon trigger. This is done for the following database accounts:

- The OHI table owner and the OHI batch account
- The OHI_VIEW_OWNER and OHI_DPS_USER account
- The OHI user accounts (presence in ALG_FUNCTIONARISSEN is sufficient)
- The accounts that have received a grant on an OHI table owner object (except for SYS & SYSTEM when granted inappropriately) or that have been granted a standard OHI role.

3.2.3.3. Install/check database component Oracle JVM

The database option Oracle JVM (JServer JAVA Virtual Machine) must be installed in the database. If you did not install it initially, you can add it by running some scripts. Consult the Database Reference guide for the relevant script.

The select statement below checks whether the JVM is present. (Use user SYS or an account with DBA privileges).

```
select comp_id
,      comp_name
,      version
,      version_full
,      status
from   dba_registry;
```

This should return a line for comp_id JAVAVM:

```
JAVAVM  JServer  JAVA Virtual Machine  19.0.0.0.0  19.5.0.0.0  VALID
```

NOTE: the value of version_full depends on the patch s (RUR or RU) you installed.

You should check whether all java objects are valid:

```
select owner
,      object_type
,      status
,      count(1)
from   all_objects
where  object_type like '%JAVA%'
group by
      owner
,      object_type
,      status
order by
      owner
,      object_type
,      status;
```

This should return information like:

OWNER	OBJECT_TYPE	STATUS	COUNT (1)
MDSYS	JAVA CLASS	VALID	2888
MDSYS	JAVA RESOURCE	VALID	51
OJVMSYS	JAVA DATA	VALID	3
SYS	JAVA CLASS	VALID	34475
SYS	JAVA DATA	VALID	1208
SYS	JAVA RESOURCE	VALID	1642
SYS	JAVA SOURCE	VALID	2

Numbers do not have to match exactly. The MDSYS owned classes are not a requirement, so when they are not shown that is no issue.

Be sure the initialization parameters `SHARED_POOL_SIZE` and `JAVA_POOL_SIZE` are large enough. When using automatic shared memory management set minimum values that are clearly large enough: e.g., at least 496Mb and 128Mb respectively for smaller environments running on machines with a limited footprint.

3.2.3.4. Check database component XML DB

OHI Back Office requires the installation of XML DB. As this is a required component starting with database release 12c you do not have to take any action. You can verify the presence of the component using the instructions described in Chapter 2 of the XML DB Developer's Guide.

3.2.3.5. Database Instance Parameters

This paragraph describes the non-hidden (not prefixed by an underscore) database instance parameters that must be set. The values given here are minimum values. When a parameter is not set in accordance with these values, an alert or an error may occur during installation of the OHI database structure. Please see the certification documentation for the latest details about hidden parameters or additional different (temporary) settings.

A complete set of parameters and checks/settings that apply can be found in an Appendix of the Oracle Health Insurance Release Installation manual.

We advise to implement settings at the PDB level, whenever possible. For some specific parameters this is a requirement as CDB based settings are not visible at the PDB level.

When you will run multiple OHI BO PDB's in a single CDB please consider increasing CDB parameters accordingly, for instance the `JOB_QUEUE_PROCESSES` minimum value for 3 OHI BO PDB's should be at least 30.

If you set a memory parameter at CDB level, the CDB setting must be higher than the sum of the PDB settings, because the CDB itself also needs memory. If this condition is not satisfied, the PDB value might still be dynamically limited to 50% or less of the value set at CDB level. Alert messages will show that this has happened.

If you try to increase a PDB level setting too much while a CDB level value is set, you will receive error messages like:

- ORA-02097: parameter cannot be modified because specified value is invalid
ORA-56748: invalid value 536870912 for parameter `shared_pool_size`; must be smaller than 50% of parameter `shared_pool_size` of the root container
- ORA-02097: parameter cannot be modified because specified value is invalid
ORA-56747: invalid value 3221225472 for parameter `sga_target`; must be smaller than parameter `sga_target` of the root container

When you run into an error like the ones above, you need to reset (remove) or increase the value of the CDB level parameter, restart the CDB and then set the value of the PDB parameter to the requested value.

We advise against enabling fully automated memory management, so leave `MEMORY_MAX_TARGET` and `MEMORY_TARGET` zero or do not set them (their default is zero).

Memory settings

```

CDB$ROOT:
JAVA_POOL_SIZE           >= 128M  # multiple of 16MB
PGA_AGGREGATE_LIMIT      >= 12G   # multiple of 16MB
PGA_AGGREGATE_TARGET     >= 128M  # multiple of 16MB
MEMORY_MAX_TARGET        = 0      # is default
MEMORY_TARGET            = 0      # is default
SGA_TARGET               >= 2048M # multiple of 16MB (assumes 1 OHI PDB + RCU PDB)
SGA_MAX_SIZE             >= 2192M # multiple of 16MB

PDB: (1)
SHARED_POOL_SIZE         >= 512M  # multiple of 16MB
DB_CACHE_SIZE            >= 128M  # multiple of 16MB
SGA_TARGET               >= 1024M # multiple of 16MB

```

Other settings

```

CDB$ROOT:
DB_BLOCK_SIZE            = 8192   # can be left default
DML_LOCKS                >= 500
EVENT                   event 10195 not set
PROCESSES                >= 200

PDB:
AWR_PDB_AUTOFLUSH_ENABLED = TRUE
JOB_QUEUE_PROCESSES       >= 10   # Needed for OZGISTAS; default ok
NLS_SORT                  = BINARY
OPEN_CURSORS              >= 500
OPTIMIZER_MODE            = ALL_ROWS
PARALLEL_DEGREE_POLICY    = MANUAL
PARALLEL_FORCE_LOCAL      = TRUE
PARALLEL_MAX_SERVERS      >= 8     # 0 when running into parallel issues
PARALLEL_MIN_TIME_THRESHOLD = 30
REMOTE_DEPENDENCIES_MODE  = SIGNATURE
SESSION_CACHED_CURSORS    >= 500
STATISTICS_LEVEL          = TYPICAL (2)
UNDO_MANAGEMENT           = AUTO
UNDO_RETENTION            >= 10800 # at least 3 hours

```

- (1) OHI installation software checks whether DB_CACHE_SIZE and SHARED_POOL_SIZE are set. When these are set only at CDB\$ROOT level, the values as seen from within the PDB are set at zero, which is reported by OHI as not being correct. Therefore, the values need to be set at PDB level in a 19c database.

When you run into an error like the one below, make sure you reset (remove) the value of the CDB\$ROOT parameter, restart the CDB\$ROOT and set the value of the PDB parameter.

ORA-02097: parameter cannot be modified because specified value is invalid

ORA-56748: invalid value 536870912 for parameter shared_pool_size; must be smaller than 50% of parameter shared_pool_size of the root container

- (2) STATISTICS_LEVEL = ALL may result in very slow CPU-intensive execution of some SQL statements.

The following parameters are influenced by the OS environment variables used by the process of the connecting session. When they are not set or specified the database instance values apply. Below example values are shown for a Dutch environment. You may want to set them at the database (PDB) level.

```

PDB:
NLS_LANGUAGE              = DUTCH    # Choose the appropriate language
NLS_NUMERIC_CHARACTERS    = ".,."   ** # personal choice

```

```
NLS_TERRITORY           = "THE NETHERLANDS"
```

** = this setting cannot be changed once the application is used

3.2.3.6. Database services

Oracle advises to create at least one service for the pluggable database (with a name that differs from the PDB name) so you do not fully rely on the default service.

The advantages of using multiple services are:

- It is possible to distinguish between certain usage groups, which makes it easier to investigate performance issues, because e.g. AWR reports contain details per service.
- If necessary, it is possible to manipulate the RAC nodes where certain usage types run without changing any client configuration.
- If necessary, you can activate resource management in the database to limit the resources used by each service, and prioritize services in that way.
- You comply with the advice not to use the default service

Note that the service names do not have to be the same as the aliases used in tnsnames.ora entries.

The table below contains the services OHI recommends as a starting point. The “Node(s)” column only applies to RAC and assumes a 2-node system.

(Cluster) Service name	Node(s)	Used for
<pdb>_online	1	Forms Application, OHIJET application via PSL Web Services
<pdb>_batch	2	OHI Batches + background processing (e.g. AVF, SPM Evolve)
<pdb>_batch_all	1+2	(RAC only) Certain OHI Batches can benefit from multi-node processing
<pdb>_web	1	SVL and HSL Web Services
<pdb>_rpl	2	Ad hoc queries, reporting tools, OHI DataMarts ETL

For RAC, the node should be the preferred node with fail-over on the other node, not an “exclusive” node.

The creation of services differs for single-instance systems and for RAC systems.

For RAC, you should use SRVCTL.

For single-instance systems, create additional database services with the commands below (execute as DBA).

These services will automatically be registered with the listener.

```
set serveroutput on
begin
  begin
    DBMS_SERVICE.CREATE_SERVICE
    ( service_name => 'vohi'
      , network_name => 'vohi.ohi.oracle.com'
    );
  exception
    when others
    then
      -- easy way to continue when the service already exists
      dbms_output.put_line('Skipped: '||sqlerrm);
end;
DBMS_SERVICE.START_SERVICE
```

```

    ( service_name => 'vohi'
    );
end;

```

Also make sure a startup trigger like the one below is created for starting the additional service(s) when the pluggable database is started. Adapt it to your needs.

```

create or replace trigger ohi_start_services
after startup on database
declare
    e_service_already_started EXCEPTION;
    PRAGMA EXCEPTION_INIT(e_service_already_started,-44305);
begin
    FOR rec IN
    ( SELECT NAME
      FROM   dba_services
      WHERE  NAME like 'vohi%'
    )
    LOOP
        BEGIN
            dbms_service.start_service(rec.NAME);
        EXCEPTION
            WHEN e_service_already_started
            THEN
                --this means the service is already
                --running so no need to crash
                NULL;
        END;
    END LOOP;
end;

```

Beware that when cloning databases, you might need to change your service definitions and service startup code. Also clean up obsolete services in table *DBA_SERVICES* by calling `dbms_service.delete_service`.

3.2.3.7. Tablespaces

The following tablespaces have to be created for OHI Back Office data and indexes (*OZG_DIM**. is for storing configuration data, *OZG_FACT**. is for storing the operational changing data):

```

OZG_DIM_FIN_IND
OZG_DIM_FIN_TAB
OZG_DIM_REL_IND
OZG_DIM_REL_TAB
OZG_DIM_SYS_IND
OZG_DIM_SYS_TAB
OZG_DIM_ZRG_IND
OZG_DIM_ZRG_TAB
OZG_FACT_FIN_IND
OZG_FACT_FIN_TAB
OZG_FACT_REL_IND
OZG_FACT_REL_TAB
OZG_FACT_SYS_IND
OZG_FACT_SYS_TAB
OZG_FACT_ZRG_IND
OZG_FACT_ZRG_TAB
OZG_LOG_IND
OZG_LOG_TAB

```

The tablespaces have to meet the following requirements:

- Locally Managed

Automatic allocation

Automatic Segment Space Management (ASSM)

8K block size



Attention 1: OHI Back Office requires the use of a *default temporary* table space for temporary segments.

3.2.3.8. Other Requirements

Other requirements that will be checked:

- Database CHARACTERSET is AL32UTF8
- PLAN_TABLE must exist
- System statistics must be present
- Direct grants from Secure Application Role OZG_ROLE are not allowed
- Table settings for regular tables must be (this is arranged by the application but values may be increased when deemed necessary):

PCT_FREE	=	15% (5% for logging tables and 10% for a small subset)
INI_TRANS	=	minimal 4. for some more concurrency sensitive tables 16 or 20



Attention 2: It is *not* permitted to have activated (trace) events in the database in the production environment unless this has been requested explicitly by OHI Development or Oracle Support Services.



Attention 3: If Database or Application Server settings have been changed which were not advised or prescribed, OHI Development or Oracle Support Services may ask the customer to have these settings reversed if problems occur that may seem to be related to these settings.

The underlying reason is to prevent unnecessary instability risks. When using customized applications that interfere with OHI Back Office functionality this has to be considered as well.



Advice 1: OHI advises the use of *Oracle Resource Management*.

For additional information please see the Resource Management paragraph.



Advice 2: OHI advises the use of *Flash Recovery Area*.
The following parameters have to be set up for this:

```
DB_RECOVERY_FILE_DEST
DB_RECOVERY_FILE_DEST_SIZE
```



Advice 3: OHI advises the use of *Oracle Managed Files*.
The following parameters have to be set up for this:

```
DB_CREATE_FILE_DEST
DB_CREATE_ONLINE_LOG_DEST_n
```

3.3. SETTING UP THE ENVIRONMENT VARIABLES

OHI applications require the necessary settings for OS environment variables (so these are *not* database parameters in the initialization/server parameter file) to be included in the file `ozg_init.env`. This file should be executed when the OHI owner account (usually `oracle`) logs on to the application server (you might consider creating a similar environment settings file for a dedicated database server; from OHI BO perspective the settings are only required on the server that acts as application server).

This initial execution is usually achieved by calling `ozg_init.env` in the file `.profile` (or `.bash_profile` in Linux `bash` shell) in the OS login home directory.

Log on with the `oracle` account and set up the environment variables according to the instructions in this section.

An example of a call in `.profile`:

```
./u01/app/oracle/product/OHI/admin/ozg_init.env
```

Setting up the variables mentioned here is *mandatory*.

A sample template file with the required settings for the OS environment variables is available in the `$OZG_BASE/conf` directory.



Hint: Copy file `ozg_init.env` to the directory `$OZG_ADMIN`. Please read the next paragraphs about the mandatory directory structure to understand the meaning of this variable and apply the desired settings, using the instructions of this chapter.

3.3.1. Syntax of the Environment Variables

The syntax for setting up environment variables in (Bash) shell is as follows:

```
export variable_name=value
```

3.3.2. Setting up the Environment Variables

The environment variables required for OHI Back Office consist of 3 subsets:

- **Variables for Oracle Database Server**
These variables have to be set up in the Database Tier, according to the instructions in the *Oracle Database Installation Guide*.
- **Variables for Application Server**
These variables have to be set up in the Middleware Tier, according to the instructions in the *Fusion Middleware Installation Guides*.
- **Variables for OHI Back Office**
These variables have to be set up in the Middleware Tier and partly in the Database Tier, if the Tiers are installed on separate servers.

The following sections provide a more detailed description of the configuration of the *specific* database/application server and OHI Back Office variables, for each variable.



Attention 1: Other Database and Application Server variables must be set up according to the instructions in the relevant product installation manuals.

NLS_LANG Variable

This variable must be configured for the desired language, territory and character set for use by client tools such as Oracle Forms. See paragraph [Localization Support](#)

It can be overruled, see later in this manual.

For a Dutch implementation this is typically:

```
"DUTCH_THE NETHERLANDS.WE8MSWIN1252".
```

For Mexican implementations this is typically:

```
"SPANISH_MEXICO.WE8MSWIN1252".
```

DBMS_LANG Variable

This OHI variable specifies a value that is used to set the NLS_LANG variable for use by the database tools (such as Import, Export, SQL*Plus, SQL*Loader etc.) and software. When these tools are called from OHI software the (OHI specific) DBMS_LANG value is used to overrule the value of NLS_LANG (described in the previous paragraph).

It specifies the default language and character set to run these tools in, meaning that for example .log and .out files as created by batches that run in SQL*Plus will be created in this character set by default. Specific output files, such as created from 'batch runs' or .xml files, can be created in a different character set. This different character set can be specified at different levels:

3. By a Back Office parameter as overall default.
4. At module-specific level.
5. As a parameter for the batch request that produces the file.

DBMS_LANG is also used as default when reading or interpreting files (including SQL*Plus scripts).

Possible values are identical to these given for NLS_LANG.

This variable has to be configured on the Middleware Tier *and* the Database Tier.

NLS_DATE_FORMAT Variable

This variable has to be configured for the desired date format. For the Dutch implementation this is normally DD-MM-RRRR.

This variable has to be configured on the Middleware Tier *and* the Database Tier.



This setting cannot be changed once OHI Back Office is used.

NLS_NUMERIC_CHARACTERS Variable

This variable has to be configured for the desired combination for decimal and group separator. For the Dutch implementation this is normally: “, .”.

This variable has to be configured on the Middleware Tier *and* the Database Tier. This setting cannot be changed once OHI Back Office is used.

NLS_SORT Variable

This variable has to be set on BINARY for performance reasons.

This variable has to be configured on the Middleware Tier *and* the Database Tier.

TNS_ADMIN Variable

This variable has to be configured to the directory which includes the Oracle Net files. On the application server (on which no Listener runs) this includes files `tnsnames.ora` and `sqlnet.ora`. On the database server (on which the Listener does run) the file `listener.ora` will also be present.

This variable has to be configured on the Middleware Tier *and* the Database Tier.

Because both Tiers have an Oracle Home directory for database software, we recommend using `$ORACLE_HOME/network/admin` for ease of management but you may also opt for a central location on a network share (which makes the functioning of your environment dependent on the availability of that network share!).

OZG_ORATAB_OZG Variable

This variable has to specify the entry in `$ORATAB` which refers to the home (root) directory of the OHI Back Office software. Typically this is an entry in `$ORATAB` (typically `/etc/oratab`) with name OZG (or nowadays often OHI) referencing a line like:

```
OHI:/u01/app/oracle/product/OHI:N
```

It will be used to determine the location for OZG_ROOT.

This OZG_ORATAB_OZG variable has to be configured on the Application Middleware Tier *and* the Database Tier.

OZG_ORATAB_DB19 Variable

This variable has to specify the entry in `$ORATAB` which refers to the home directory of Oracle 19c (19.x.x.x) Database software.

This variable has to be configured on the Middleware Tier *and* the Database Tier.

OZG_ORATAB_FRS122119 Variable

This variable has to specify the entry in `$ORATAB` which refers to the home directory of the Oracle 12c R2 Forms & Reports Service software. This is the Oracle Home directory of the Fusion Middleware 12.2.1.4.0 installation.

This variable has to be configured on the Middleware Tier if you want to use Forms 12.2.1.19.0 for one or more of your OHI environments.

OZG_ORATAB_WLS122119 Variable

This variable has to specify the entry in `$ORATAB` which refers to the home directory of the Oracle WebLogic Server software (the `wlserver` directory within the Fusion Middleware home).

This variable has to be configured on the Middleware Tier if you want to use Forms 12.2.1.19.0 for one or more of your OHI environments.



Although Forms 12.2.1.19.0 requires Fusion MiddleWare 12.2.1.4.0, it cannot run from the same ORACLE_HOME as a Forms 12.2.1.4.0 installation. Instead, it needs a duplicate installation of Fusion MiddleWare 12.2.1.4.0. To distinguish this from the FMW home used for Forms 12.2.1.4.0, we need this separate variable OZG_ORATAB_WLS122119.



Note that OHI release 10.24.3.0.0 and higher will only support Forms 12.2.1.19.0.

OZG_ROOT Variable

This variable has to specify the home directory of OHI:

```
export OZG_ROOT=`dbhome $OZG_ORATAB_OZG`
```

This variable has to be configured on the Middleware Tier *and* the Database Tier.

OZG_ADMIN Variable

This variable has to specify the directory in which OHI administration and configuration files are located: `$OZG_ROOT/admin`.

This variable has to be configured on the Middleware Tier *and* the Database Tier.

PATH Variable

The directory in `$OZG_ADMIN` has to be included in the `PATH` variable.

This variable has to be configured on the Middleware Tier *and* the Database Tier.

OZG_BASE Variable

This variable has to be configured to specify the home directory of OHI Back Office for the current OHI environment. More than one OHI Back Office can exist in subdirectories of the OHI home directory `$OZG_ROOT`. `OZG_BASE` typically refers to a folder with the name of `$TWO_TASK` within `$OZG_ROOT`: `$OZG_ROOT/$TWO_TASK`. (This assumes the OHI environment name is the same as the PDB service name for that environment.)

This variable has to be configured on the Middleware Tier.

OZG_PATCH Variable

This variable has to be configured to specify the directory in which the OHI (patch) releases have to be downloaded and unzipped: typically, `$OZG_ROOT/patch`.

This variable has to be configured on the Middleware Tier.

FORMS_PATH Variable

This variable has to be configured for the directory in which the OHI Back Office runtime Forms modules are located for the current environment: `$OZG_BASE/bin`. This must match the value of `FORMS_PATH` in the `servlet.env` files as described later in this document.

This variable has to be configured on the Middleware Tier.

FORMS_USER_DATE_FORMAT Variable

This variable, which indicates the date format used in screens, has to be configured as `DD-MM-RRRR`.

This variable has to be configured on the Middleware Tier.

FORMS_SCROLL_ALL_BUT_ONE Variable

This variable, which is user-friendly when scrolling through records in a multi-record block in a screen, has to be set to `TRUE`.

This variable has to be configured on the Middleware Tier.

FORMS_TRACE_PATH Variable

This variable has to be set up to specify a directory in which the dump files due to crashes of the Forms runtime executables have to be located.

This variable has to be configured on the Middleware Tier.

FORMS_FLAG_DIFFERENT_SUBORD Variable

This variable has to be set to 1.

This variable has to be configured on the Middleware Tier.

FORMS_DATETIME_LOCAL_TZ

This variable has to be set to GMT so it equals to the default value of FORMS_DATETIME_SERVER_TZ being GMT. This prevents time differences within screens.

This variable has to be configured on the Middleware Tier.

OZG_OUT Variable

This variable has to be configured to specify the OS directory in which the OHI Back Office batch scheduler writes the output, e.g. `$OZG_BASE/out`.

This variable can be used in the system parameters for OHI Back Office subsystem SYS (“System”); screen `SYS1010F`.

This variable has to be configured on the Middleware Tier.

OZG_LOG Variable

This variable has to be configured to specify the OS directory in which the OHI Back Office batch scheduler writes the log files, e.g. `$OZG_BASE/log`.

This variable can be used in the system parameters for OHI Back Office subsystem SYS (“System”); screen `SYS1010F`.

For a detailed description of the use of OS environment variables for use in the output and log path of the OHI Back Office batch scheduler, please refer to the following document:



Reading, Writing and Authorizing Oracle Health Insurance Application Files

This variable has to be configured on the Middleware Tier.

3.4. SETTING UP THE OHI DIRECTORY STRUCTURE

Under operating system user `oracle`, the following *mandatory* directory structure (the “custom” directory being an exception) has to be created under the `$OZG_BASE` directory on the Application Server. The OHI application source files and executables for each OHI Back Office environment (so for each ‘environment’ directory) require a separate environment subdirectory structure:

<code>\$OZG_ROOT</code>	#Oracle Health Insurance install directory	
<code>/admin</code>	#Configuration files	= <code>\$OZG_ADMIN</code>
<code>/patch</code>	#(Patch) releases	= <code>\$OZG_PATCH</code>

```
/environment    #Directory for each environment = $OZG_BASE
                #Can occur n times in 1 $OZG_ROOT
/bin            #Forms
/conf           # Template directories and ohipatch.conf
/help           #Online help documentation
/install        #DDL scripts
/java           #Java files
/log            #Batch scheduler log files
/out            #Batch scheduler output
/sh             #OS shell scripts
/sql            #SQL modules
/utils          #utilities typically used during installation
/xml            #XML files
/custom         #Custom Development managed files
```

3.4.1. bin

This directory contains all OHI Back Office Forms (Forms, Menus, Forms Libraries, Reference Forms and Object Libraries), Pro*C software and application logos.

3.4.2. conf

This directory contains the configuration file for the installation menu OHIPATCH and subdirectories with templates.

3.4.3. help

This directory contains all OHI Back Office online help documentation, which can be accessed from the application.

3.4.4. install

This directory contains all OHI Back Office DDL scripts e.g. tables, constraints, indexes and triggers.

3.4.5. java

This directory contains all OHI Back Office Java files.



Attention: Certificates and/or keystores which are possibly required for secure connections on the Internet, have to be located in the \$OZG_BASE directory of the relevant environment.

3.4.6. sh

This directory contains all OHI Back Office OS shell scripts and SQL*Loader control files.

3.4.7. sql

This directory contains all OHI Back Office SQL modules.

3.4.8. utils

This directory contains the software for the installation menu OHIPATCH and some other perl and shell utilities used during installation. Some of those are also used at runtime.

The directory `$OZG_BASE/utils` must contain the following files on the application server.

Filename	Purpose
d12c2*	Utilities to convert and compile forms sources.
OHI_CMD.pl	Perl command file to set correct environment when executing Oracle utilities and executables.
OHI_EXTENSIONS.jnlp	Configuration file for Forms Java Web Start implementation.
OHILOGSCAN.pl	Utility to scan installation log files.
OHIPATCH.pl	OHI Back Office Perl installation menu for installing major releases and interim patches.
OHIPLIB.pm	OHI Back Office Perl library module.
ohirf34w_nl-NL.res; ohirf34w_en-US.res;	OHI Back Office Forms Terminal Resource File (for configuring keyboard mappings) for one or several languages. These are not all required, just copy the one(s) you need. from the <code>\$OZG_BASE/conf/Back-Office</code> folder. See the paragraph about forms configuration for more information.
OHI_WEBSTART.jnlp	Configuration file for Forms Java Web Start implementation.
OZG_GIF.jar	OHI Back Office jar file for Oracle Application Server.

3.4.9. xml

This directory contains all OHI Back Office XML files.

3.4.10. custom

This directory can contain custom development managed files. For instance, files that are processed by custom developed scripts should be placed here or in a subdirectory of this directory. The “custom” directory is not mandatory; however, the naming is prescribed.

Additionally, the following 2 directories have to be created.

Normally, these are created as subdirectories of `$OZG_BASE`. It is permitted to create these directories in a different physical location:

3.4.11. log

This directory contains all OHI Back Office batch scheduler log files (see the environment variable `$OZG_LOG`).

The OS users `oracle` and `batch` have to have write privileges for this directory.

3.4.12. out

This directory contains all OHI Back Office batch scheduler output files (see the environment variable `$OZG_OUT`).

The OS users `oracle` and `batch` have to have write privileges for this directory.

Using the OS user `oracle`, create the following *mandatory* directory structure in `$OZG_ROOT` on the Middleware Tier for the tools, utilities and releases of the OHI Back Office application:

3.4.13. admin

This directory contains the environment setting script `ozg_init.env` and some OHI Back Office files that need to be shared over multiple OHI environments. This directory is required on all Middleware Tier servers. This directory is optional on Database Tier servers.



Attention: If the Database Tier and the Middleware Tier have been installed on different servers and you choose to use script `ozg_init.env` on the Database Tier server(s), be aware that the contents of this script will differ between tiers.

Therefore, these directories *cannot* be shared

The directory `$OZG_ADMIN` has to contain the following files on the application server. The directory `$OZG_ADMIN` on the database server contains only the files that are marked with **(***)**.

Filename	Purpose
<code>ozg_init.env (***)</code>	Environment file for OHI Back Office settings.
<code>ozg_batch_start.sh</code> (optional)	OS command file to start OHI Back Office batch schedulers for all environments.
<code>ozg_batch_stop.sh</code> (optional)	OS command file to stop OHI Back Office batch schedulers for all environments.

3.4.14. patch

This directory contains all OHI (patch) releases which have to be installed. This directory only applies to the application server(s).

3.5. INSTALLATION OF OHI BACK OFFICE APPLICATION LOGOS

OHI Back Office uses 2 logos (which are loaded dynamically at runtime):

- **OZGIMG01.gif**
Is displayed on the OHI Back Office start screen `OZGSTART`.
Size: 390 x 160 pixels
- **OZGIMG02.gif**
Is displayed on the OHI Back Office Info screen `OZGABOUT`.
Size: 120 x 45 pixels

The logos have to be placed in directory `$OZG_BASE/bin` on the Middleware Tier. If they cannot be found an error message will be shown during startup of the user interface.

Initial logos are available in the `$OZG_BASE/conf` folder; the customer is free to replace these logos with private company logos; the only requirement is that the naming has to be maintained.

3.6. SETTING UP UNIVERSAL ORACLE OS SCRIPTS

3.6.1. oratab

The file `$ORATAB` (normally `/etc/oratab` or `/var/opt/oracle/oratab`) is used by Oracle software to register software installation homes and databases. OHI uses the same file to register software installation homes and OHI environments.

Entries *have to be* included for the following products.

Be sure to use transparent naming of the entries put in ORATAB. The best way is to name the entry the same as the directory where the software is stored. There can be exceptions for this, it is not an obligation.

The codes that are used for the entries can be selected freely (but they have to be indicated in uppercase characters). The values of the variables `$OZG_ORATAB_*` in `$OZG_ADMIN/ozg_init.env` must match the entries in `$ORATAB`

- **Oracle WebLogic Fusion MiddleWare**

Sample code: WLS12214

Sample code: WLS122119

- **Oracle Fusion Middleware Forms & Reports Services**

Sample code: FRS12214

Sample code: FRS122119

- **Oracle Database**

If the Middleware Tier and the Database Tier are installed on 1 server, the entry refers to the Database Server software home, otherwise to the Database Client software home on the application server.

Sample code: DB19

- **OHI**

This entry identifies an OHI installation, containing one or more OHI environments. The entry will be used to determine the root folder for OHI environments (`$OZG_ROOT`).

Sample code: OHI (previously OZG)

Sample entries in \$ORATAB

```
...
DB19:/u01/app/oracle/product/19/db_1:N
WLS12214:/u01/app/oracle/product/frs12214/wlserver:N
FRS12214:/u01/app/oracle/product/frs12214:N:
WLS122119:/u01/app/oracle/product/frs122119/wlserver:N
FRS122119:/u01/app/oracle/product/frs122119:N:
OHI:/ohi/app/OHI:N
...
```

Sample corresponding variables \$OZG_ADMIN/ozg_init.env

```
...
export OZG_ORATAB_DB19=DB19
export OZG_ORATAB_WLS12214=WLS12214
export OZG_ORATAB_FRS12214=FRS12214
export OZG_ORATAB_WLS122119=WLS122119
export OZG_ORATAB_FRS122119=FRS122119
export OZG_ORATAB_OZG=OHI
...
```

`ozg_init.env` uses the standard Oracle utilities `dbhome` and `oraenv` to translate the `$OZG_ORATAB_*` values to the locations in `$ORATAB`.

It is important that no other `oratab` is indicated in these utilities. Check that `$ORATAB` in these files (located in `/usr/local/bin` and the `$ORACLE_HOME/bin` directories) does not refer to a non-existing `oratab` file.

When reference is made to a non-existing `oratab` file, this can be modified in the relevant utilities, or a symbolic link can be created for the non-existing `oratab` file which refers to the correct `oratab` file.

3.6.2. oraenv

`oraenv` is an Oracle utility that is installed with Oracle software. It can be used to set the required environment variables to access a given database, software installation, etc.

For certain (older) versions of `oraenv` the error message “unlimited: bad number” may be encountered, or the execution of `oraenv` may get stuck (on `exec $ORACLE_HOME/bin/osh`). In that case the workaround as described in My Oracle Support note 1023496.6 has to be implemented.

3.7. INSTALL ADDITIONAL PERL MODULES

In order to use the OHI installation menu `OHIPATCH`, you need to install additional Perl modules.

It is *mandatory* to install these modules *before* starting an initial installation of OHI software. The installation menu will not work until these modules are installed.

For installation instructions, see [Appendix D – Installing required Perl modules](#).

3.8. CUSTOMIZING TOOLBAR ICONS

When you are using the Online HTTP Link in OHI BackOffice it is possible to define your own icons for the HTTP link buttons, so it becomes easier to distinguish their functionality. Register the file names of the icons in screen SYS1147F (Maintain HTTP Link). They should be GIF files.

Include your custom GIF files in a custom jar file that is used in addition to the `OZG_GIF.jar` file delivered with the OHI Back Office application.

To create a jar file, go to the (application server) directory where you placed your GIF files and use the following command:

```
jar -cvf OZG_GIF_CUSTOM.jar *.gif
```

This will put all `*.gif` files in your folder in the file `OZG_GIF_CUSTOM.jar`.

Next steps:

1. Sign `OZG_GIF_CUSTOM.jar` with an official certificate (see the next paragraph)
2. Place `OZG_GIF_CUSTOM.jar` in a Virtual Directory so it can be downloaded by the end users' browsers (e.g. in `$OG_BASE/utls` where `OZG_GIF.jar` is placed by default).
3. Create a new file `OHI_CUSTOM_EXTENSIONS.jnlp`:

Copy file `OHI_EXTENSIONS.jnlp` (default in `$OZG_BASE/utls`) to `OHI_CUSTOM_EXTENSIONS.jnlp` in the same directory.

Edit `OHI_CUSTOM_EXTENSIONS.jnlp` and change `OZG_GIF.jar` to `OZG_GIF_CUSTOM.jar`

`<resources>`

`<jar href="./OZG_GIF_CUSTOM.jar"/>`

`</resources>`

4. Register the new `OHI_CUSTOM_EXTENSIONS.jnlp` in `OHI_WEBSTART.jnlp`:
Edit `OHI_WEBSTART.jnlp` and copy the line with `OHI_EXTENSIONS.jnlp`
Change the new line to use


```

        name="CustomExtensions"
        OHI_CUSTOM_EXTENSIONS.jnlp.

<resources>

<!-- Application Resources -->

<j2se version="%java_version%" href="http://java.sun.com/products/autodl/j2se"/>

<jfx:javafx-runtime version="2.2+"/>

<jar href="%contextRoot%/java/frmall.jar" download="eager" main="true" />

<extension name="Extensions" href="%envVirtualFolder%/utils/OHI_EXTENSIONS.jnlp"/>

<extension name="CustomExtensions"
href="%envVirtualFolder%/utils/OHI_CUSTOM_EXTENSIONS.jnlp"/>

</resources>

```

When using WebStart, there appears to be no need to register OZG_GIF_CUSTOM.jar in formsweb.cfg, in the archive parameter.

3.8.1. Signing a .jar file

If you create a new .jar file, it is necessary to sign it to prevent security messages constantly appearing when the .jar file is downloaded by a browser from the application servers.

The jar file should be signed with an official certificate, authorized by an official Certificate Authority (CA). You can use the Java jarsigner utility to do this. See the Java documentation. If your organization is its own Certificate Authority, other is your organization may be better placed to sign the jar file(s). To sign a jar file with a certificate from your own organization using the WebLogic tooling discussed below, you will need to import an Intermediate Certificate or a Certificate Chain first. See the WebLogic documentation for details.

The following commands offer you a sign the jar file in a less secure way, using a so-called self-signed certificate. With the ever-tightening security of Java applets in the browser and of the browsers themselves, this self-signed certificate requires you to lower the security level of the Java runtime and the browser. Use this method only for testing and never for production environments.

Instructions for testing with a self-signed certificate:

5. Copy the default script from your Oracle Forms ORACLE_HOME folder and create a local version:

```
cp $ORACLE_HOME/forms/templates/scripts/sign_webutil.sh
my_sign_webutil.sh
```

6. Adjust your local copy and at least specify a value for these variables:
 - o DN_CN
 - o KEYSTORE_PASSWORD
 - o JAR_KEY_PASSWORD

7. Sign your custom jar file using your custom script:


```
. ozg_init.env FRS122119
my_sign_webutil.sh OZG_GIF_CUSTOM.jar
```

Beware that your keystore will be created as file \$HOME/.keystore by default.

After this has been created you need to use the same password for subsequent signing calls using the same keystore.

8. After signing copy the .jar file to a location where it can be downloaded by the Forms sessions started by users. This is typically \$OZG_BASE/utls or \$ORACLE_HOME/forms/java

When using WebStart, there appears to be no need to register OZG_GIF.jar or OZG_GIF_CUSTOM.jar in formsweb.cfg, in the `archive` parameter.

4. CONFIGURATION ORACLE FUSION MIDDLEWARE SOFTWARE FOR OHI BACK OFFICE

This chapter describes the configuration of the Oracle Forms and Reports components of the Oracle Fusion Middleware (FMW) product “Oracle Forms and Reports Services” (FRS) for the OHI Back Office application. .

4.1. INSTALLATION AND INITIAL CONFIGURATION OF REQUIRED SOFTWARE

This installation has been verified on a Linux platform. Other platforms are currently not certified nor supported. For specific versions of Linux and possible extra requirements, see the “Certification” information for the product “Oracle Health Insurance Back Office”

4.1.1. Acquire the Software

Make sure you download the correct software for your platform. As only the 64 bit platform is supported download the following components:

- "REL: Oracle Fusion Middleware 12c Infrastructure 12.2.1.4.0" This is available as download on [Oracle Software Delivery Cloud](#).
- “DLP: Oracle Forms and Reports 12.2.1.19.0 (Oracle Forms and Reports)”, also available from the [Oracle Software Delivery Cloud](#). We will refer to this product as FRS.

Optional: Oracle WebLogic Server 12.2.1.4 Generic Installer for Oracle WebLogic Server and Oracle Coherence. This is available as download on <https://www.oracle.com/middleware/technologies/weblogic-server-installers-downloads.html>



Attention: Forms 12c requires the FMW Infrastructure, which includes OPSS (Oracle Platform Security Services) database objects. If you plan to install OHI Service Layer Web Services (SVL) or OHI HTTP Service Layer (HSL) in a different home, potentially on a different application server, you may want to use the Oracle WebLogic Server 12.2.1.4 Generic Installer to install those products without the overhead of the OPSS database objects.

A 64bit JDK must be available. Please make sure you use a certified JDK version. Check for certified versions on support.oracle.com. Oracle Forms and Reports 12c (12.2.1.19.0) is certified with JDK 1.8.0_341+

Several patches are required on top of these basic software installations.

Proceed by installing the additional required patches listed in the OHI certification document "OHI_BO_required_patches_on_DB_and_FMW.pdf" on [Secure Sites](#).

Be aware that additional patches may be identified later.

4.1.2. Install Oracle WebLogic Server

OHI Back Office uses only the Forms Services of the Oracle FMW product stack. For the 12c ‘Application Server’ you need to install the Oracle WebLogic Server (WLS) first. The WLS installation creates a Middleware home directory structure. After you have installed WLS you can install FRS.

Fusion Middleware 12c does not have a “Forms & Reports Stand-alone” installer, as once existed with older versions. FRS 12c requires a WLS installation of the type “FMW Infrastructure”. FRS itself is dependent on some components within this FMW Infrastructure which itself require (and are dependent on) Oracle Platform Security Services (OPSS). OPSS in turn requires certain database schemas to be available for Security data and for auditing. These schemas are installed in an available database by running the Repository Creation utility (RCU) for each WLS/FMW Forms domain.

The result is a FMW installation (more precisely, a WebLogic Domain) that is linked to c.q. depends on database contents. One Domain can service multiple OHI environments (e.g. 3 development environments). These OHI environments are typically refreshed or cloned individually.

If we install the OPSS and Audit schemas in one of the OHI BO databases (the same PDB), that will impact the way OHI Back Office environments can be cloned. We can no longer simply copy (plugout and plugin) an OHI BO database from one environment to another environment, if the FMW Domain is different. The FMW Managed Server (typically named WLS_FORMS) will not start if the corresponding OPSS and Audit schemas are not available using the original JDBC connectors that are part of the target FMW installation.

Several alternatives exist, to prevent or work around this dependency:

- Install the OPSS and Audit schemas in a separate PDB in the same CDB as the OHI BO PDB. Do not clone the OPSS PDB. Starting with database version 19c, this can be done without a multi-tenancy license, as long as you do not put more than 3 PDBs in one CDB.
- Install the OPSS and Audit schemas in a separate database (separate CDB+PDB). Do not clone the OPSS database. This requires some more server memory and management effort.
- Adapt the clone procedure: execute extra activities to link the target FMW Domain to the cloned OHI BO PDB: rerun the script config.sh, connect to new DB, get the RCU configuration, modify passwords, etc.

From an OHI perspective we strongly advise to use a separate database (PDB) for creating the OPSS schemas, as these schemas belong to a WLS Forms configuration and not to a specific OHI environment. You may choose a single or a few databases (PDBs in one or more CDBs, depending on the required high-availability for different Domains) to contain several sets of OPSS schemas, with different prefixes per set, for all WLS Forms configurations. The database version of this database may differ from the OHI database version and should be certified as ‘Target Database for RCU’ for FMW Infrastructure.

For all documentation describing the Oracle Fusion Middleware environment, concepts, installation requirements, installation process, etc. please see:



Oracle Online Documentation Library Middleware 12c Release 2 (12.2.1.4):
WebLogic Server available online at

<https://docs.oracle.com/en/middleware/fusion-middleware/weblogic-server/12.2.1.4/index.html>



Oracle Online Documentation Library Middleware 12c Release 2 (12.2.1.19):
Oracle Forms and Reports, available online at

<https://docs.oracle.com/en/middleware/developer-tools/forms/12.2.1.19/index.html>



Oracle Fusion Middleware System Requirements and Specifications 12c (12.2.1.4.0) (document E95158-07):

<https://docs.oracle.com/en/middleware/fusion-middleware/12.2.1.4/sysrs/>



Install, Patch and Upgrade Oracle Fusion Middleware 12c:

<https://docs.oracle.com/en/middleware/fusion-middleware/12.2.1.4/install-patch-tasks.html>



Upgrade Oracle Forms 12c:

<https://docs.oracle.com/en/middleware/developer-tools/forms/12.2.1.19/install-fnr/upgrading-oracle-forms.html#GUID-20AC6348-AA11-4D36-A460-8C06A0218A66>



Attention: This manual does not describe an upgrade. Use the installation instructions as provided with the OHI release that first certifies the upgrade version.

The upgrade from Oracle Forms and Reports 12.2.1.4 to 12.2.1.19.0 is an “out of place” upgrade. It requires a full software installation in a new Oracle Home, c.q. Middleware Home. It is possible to upgrade your existing Domains (in place) or a copy of those Domains (out of place), though. The OPSS schema in the FMW Repository (RCU) of the existing Domains can only be upgraded “in place”. For more details see the documentation “Install, Patch and Upgrade Oracle Fusion Middleware 12c”.

We will use variable `$MW_HOME` to refer to the top level Middleware home. We will use the name `frs122119` (Fusion Middleware 12c 12.2.1.19.0 for the Middleware home where we will install Forms and Reports 12.2.1.19.0). So `$MW_HOME` could be `/u01/app/oracle/product/frs122119`.

This top-level folder in the directory structure contains a subfolder `wlserver` for the WebLogic Server (WLS) software.

This means the Middleware home is also the Oracle Home for the Forms installation.

A FMW home will always contain one WLS home in the subdirectory `wlserver`, resulting in `/u01/app/oracle/product/frs122119/wlserver`.

Environment variable `$MW_HOME` is set automatically when the WLS environment scripts are run (called by `setWLSEnv.sh` as present in `$MW_HOME/wlserver/server/bin`).

To start the installation, check you are running a certified Java JDK (not a JRE):

```
java -version
```

and then execute this command, using the `oracle` account:

```
java -jar fmw_12.2.1.4.0_infrastructure.jar
```

**Attention:**

It is assumed the java bin directory is present in your PATH, otherwise prefix the executable name with the relevant path, like for example:

```
/usr/java/jdk1.8.0_341/bin/java -jar  
fmw_12.2.1.4.0_infrastructure.jar
```

This is typically needed when you have configured an existing older JDK for a previous FMW installation as the default.

**Attention:**

Before you start the installer, make sure the following environment variables are not set:

- ORACLE_HOME
- LD_LIBRARY_PATH

Remove any other ORACLE_HOME directories from the PATH variable

When you have started the installer, please follow the instructions below. If this is the first installation on the machine you will be asked to specify an inventory location.

- Installation Location: Specify a new location for the Fusion Middleware home, e.g.,
/u01/app/oracle/product/frs122119
- Installation Type: Fusion Middleware Infrastructure (no examples)
- Proceed with the prerequisite checks and finish the software installation.

4.1.3. Install Oracle Forms and Reports Services

After installation of the Fusion MiddleWare Infrastructure software the FRS product must be installed.

For exact versions of the required Oracle application software and patch sets, see:



Certification tab on My Oracle Support for product Oracle Health Insurance Back Office

Within the FMW home the FMW product FRS will be installed, with its own subdirectories in this same home folder. You must specify the same Oracle Home as where the FMW Infrastructure installation was installed.

Summary of the installation steps for FRS:

- Set your environment:

```
. /u01/app/oracle/product/frs122119/wlserver/server/bin/setWLSEnv.sh
```

- Check and - if necessary - adapt your environment variables:

ORACLE_HOME (empty)

ORACLE_BASE (e.g., /u01/app/oracle/product)

JAVA_HOME (set to directory of a certified Java 1.8 JDK)

- Unzip the downloaded distribution file. Locate file fmw_12.2.1.19.0_fr_linux64.bin and cd to that directory.
- Start the installer with the command:
./fmw_12.2.1.19.0_fr_linux64.bin

- Decide what to do regarding Auto Update
- Select the FMW Home that was created during the installation of WebLogic Server and specify the same folder for the Forms and Reports product, e.g.,
`/u01/app/oracle/product/frs122119`
- Installation Type: Forms and Reports Deployment
- Proceed with the installation steps (and implement failing prerequisites if needed) until the installation is finished.
- Execute the link actions as described in [step 9](#) in paragraph "Installing Using Oracle Universal Installer"
- Check if an Oracle Forms Quarterly Patch Release (QPR) has become available for Forms 12.2.1.19.0. See MOS Note 2834458.2, "Oracle Forms Quarterly Patch Release (QPR) information".
- Install the latest Stack Patch Bundle (SPB) or Critical Patch Update (CPU) for Fusion MiddleWare.
- Install the additional required patches listed in the OHI certification document "OHI_BO_required_patches_on_DB_and_FMW.pdf" on [Secure Sites](#).
-

Run the Repository Creation Utility

Before you can create a Forms Domain, you must run the Repository Creation Utility (RCU) to create database schemas and objects for Oracle Platform Security Services (OPSS) and Auditing.

This is a mandatory step, even if you are not using Single Sign-On or OWSM. The Forms installation does not directly use these schemas, but the WebLogic Admin Server and the Managed Server WLS_FORMS will not start if these objects are not present.

For requirements for the database where the RCU creates the schema definitions you can use the link [RCU Requirements for Oracle Databases](#)

Summary of the installation steps when running the RCU:

- Set your environment:
`./u01/app/oracle/product/frs122119/wlserver/server/bin/setWLSEnv.sh`
- Just to be sure, set your NLS_LENGTH_SEMANTICS to BYTE:
`export NLS_LENGTH_SEMANTICS=BYTE`
- Start the installer:
`$MW_HOME/oracle_common/bin/rcu`
- Select "Create Repository" and "System Load and Product Load".
- Enter the Database Connection Details (typically connect as a regular DBA, i.e., SYSTEM)
- Select "Create new prefix" and enter for example "SRV1FRS1" (or a more meaningful prefix, maximum length is 12 characters, see the remark below) or "FRSD2" to identify the domain name chosen later in this documentation (underscores are not allowed). Next select the following Components:
 - "Common Infrastructure Services" (mandatory)
 - "WebLogic Services" (mandatory)
 - "Oracle Platform Security Services" (this will add 3 others: Audit Services, Audit Services Append, Audit Services Viewer)



Attention: One Forms Domain can service many different OHI environments, so do not include any indication of the OHI environment in the prefix. Consider using a prefix that easily identifies this specific domain in this specific Forms installation on this specific application server. That way, you can create several OPSS schema sets in one (central?) PDB

- Enter password(s) for schemas.
- Map Tablespaces:
You may want to store the objects of the different schemas in the same tablespace. You may want to specify a different name for a specific tablespace or for the dedicated Temp Tablespace.
Optionally, first create a different tablespace using the button "Manage Tablespaces". Then change the "Default Tablespace" and "Temp Tablespace" for the different "Schema Owners".
- Proceed with creating the schemas (this should not take more than one to two minutes).

As a result, the selected database will contain the specified schemas.



Attention: You may want to change the new accounts. By default, the passwords will expire after several months. After the passwords have expired, you will not be able to start the WebLogic Servers. So, either mark your calendar to change the passwords in time or change the accounts so the passwords never expire.

4.1.4. Configure Forms and Reports domain

When the installation of the repository objects is finished the creation of an Oracle Forms and Reports Domain can start. Only Forms Services will be configured. The resulting WLS Domain will contain the artifacts below. Before starting the configuration, please take notice of the attention points below:

- the *java components*
- the Admin Server
- the Managed Server(s) (JVM's) for running the Forms Service,
- the *system components* such as the Oracle HTTP server (OHS).



Attention: In case you were familiar with older Forms versions, the concept of an "Oracle Instance" for Forms does not exist in FMW 12c, instead a "Forms Instance" folder structure is created within the Forms domain folder.



Attention: You have a choice for the Node Manager:

- Use one Node Manager for each Domain (the default choice in WLS 12c)
- Use the same Node Manager for multiple (or all) Domains on the same server. In the step "Advanced Configuration", check, "Node Manager" to specify an existing Node Manager of the same WebLogic version.



Attention: To avoid port conflicts with other components/servers in any other FMW installations on the same server make sure you know the ports used by your current configuration and design a practical port assignment configuration. You can enter these ports when running the configuration Wizard.

- In the step “Advanced Configuration” below, check “Admin Server”, “Node Manager”, “Topology” and “System Components”. This will give you the opportunity to enter port numbers for all relevant components.



Attention: If you will run on a standalone application server machine and the environment is only used by a small user community (typically a custom development or test environment) you can remove the cluster during the Wizard step “Coherence Clusters”. That step only becomes available if you check “Managed Servers, Clusters and Coherence” in the step “Advanced Configuration”. For production environments and environments used for testing the production situation always select the clustered option because this offers more flexibility and ease of use when more than one Managed Server is needed: the cluster can easily be extended when the Forms ‘application’ is deployed on the cluster instead of a specific Managed Server.

Before starting the FRS configuration make sure you implement the necessary settings in your environment:

- `ozg_init.env`:
an entry for `OZG_ORATAB_FRS122119` should be present which looks like the settings in the template file in the `$OZG_BASE/conf/generic` folder. Most important is that the additional settings for this environment are set like `DOMAIN_NAME`, `DOMAIN_HOME` and `FORMS_INSTANCE`. Choose a useful domain name which corresponds with the prefix for the OPSS schemas created earlier by running RCU.
- The file `oratab` (referred to by variable `$ORATAB`) must contain an entry for the identifier identified by `OZG_ORATAB_FRS122119` and for `OZG_ORATAB_WLS122119` (see the template file or their definitions earlier in this manual).

When these settings are present in `ozg_init.env` and `oratab` please switch to the FRS environment settings through:

```
. ozg_init.env $OZG_ORATAB_FRS122119
$ORACLE_HOME/oracle_common/common/bin/config.sh
```

When you run the configuration tool please take note of the following remarks

- Create Domain: choose to create a new domain:
 - Domain Location:
The default location is `$MW_HOME/user_projects/domains/base_domain`, where `base_domain` will be both the location (subdirectory) and the name of the domain. You may want to choose a location outside the directory of the Middleware home.



Attention: The template script `ozg_init.env` assumes the domain is located in the Middleware home directory when you do not change `DOMAIN_HOME`. Make sure the value of `DOMAIN_NAME` and `DOMAIN_HOME` match the value you choose here.

Instead of the default value `base_domain` we will use domain name `frs_d2` (`frs_d2` for Forms and Reports Services domain 2 for release 12cR2) and refer to this name in later documentation steps

- Product Templates: Choose option 2 and 3 from the list below. The first option is required and the rest is selected when you select Oracle Forms:
 1. Basic Weblogic Server Domain [wlserver]
 2. Oracle Forms [forms]
 3. Oracle HTTP Server (Collocated) [ohs]
 4. Oracle Enterprise Manager [em] (automatically added)
 5. Oracle JRF [oracle_common] (automatically added)
 6. Weblogic Coherence cluster Extension [wlserver] (automatically added)
- Application Location:
 - The default subdirectory name is derived from earlier input. You may want to choose a location outside the directory of the Middleware home, in a separate applications folder unique to this domain
- Administrator Account:
 - Specify a username/password combination to manage the new domain later through the console with this username/password combination.
- Domain Mode and JDK:
 - Domain Mode: 'Production'
 - JDK: check this is a 1.8 JDK that conforms to the minimum requirement
- Database configuration type:
 - AutoConfiguration Options: RCU Data
 - Vendor: Oracle
 - Driver = Oracle's Driver (Thin) for Service connections
 - Enter the details to connect to your RCU database
 - Schema Owner = <earlier chosen prefix>_STB (e.g. FRS_D2_STB)
 - Button "Get RCU configuration"
- Component Datasources:
 - Check the values for the Schemas and if needed enter the passwords you have given during the RCU step. They will be used in the next step to test the datasource connections.
- Advanced Configuration:
 - You can choose to manually add Managed Server(s) and the cluster later, from the Administration Console, or have them created by this Wizard. Here, we describe the second option, running the Wizard.

- To specify a specific port, a different name or enable SSL for the Admin Server, check the option “Administration Server”.
- To specify a user name and password for the new Node Manager or for an existing Node Manager, check the option “Node Manager”.
- To specify Managed Server(s) names, ports and SSL, specify the cluster name and port or specify Coherence details, check the option “Topology”.
- Check the option “System Components” to add an Oracle HTTP Server (OHS). This is required for OHI Back Office, to facilitate for example downloading files.
- Check the option “Deployments and Services” to define machines (including port number used by the Node Manager) and target deployments.

Depending on the selection in the steps above, several other steps will be enabled.

- Administration Server:
 - Check the default name and modify if needed
 - Check the default port and modify if needed
- Node Manager:
 - Specify the type of the Node Manager and the credentials
- Managed Servers:
 - Check the default name WLS_FORMS and modify if needed
 - Check the default port and modify if needed
 - Add more Forms Servers if needed, giving each a unique and free port number
- Clusters:
 - Check the default name of the Weblogic Cluster and modify if needed
 - Enter HTTP Frontend details, if applicable for your architecture
- Skip Server Templates and Dynamic Servers unless you do want to use this
- Assign Servers to Clusters:
 - Move your Forms Managed Server(s) to the right to include them in the cluster, if applicable
- Coherence Clusters:
 - This cannot be removed
 - Make sure the name is unique
- Machines:
 - Make sure you use the tab “Unix Machine” for Linux servers (remove ‘regular’ machines in the Machine tab if present) and create a new “Unix machine” if not present and enable it. The “Name” does not have to match the host name
 - Choose the port nr for the machine as this will be the port number for the Node Manager
- Assign Servers to Machines:
 - Move the server(s) to the correct Machine(s)
- Move on with ‘Next’ through Virtual Targets and Partitions (use them if appropriate) until System Components is active (it will become visible if it was not yet shown)
- System Components:

- Unless you have a custom architecture (e.g., with Frontend Host Capture), add a System Component with Component Type = OHS and as name for example “ohs1”. To be able to do this it was required that you selected “Oracle HTTP Server (Collocated) - 12.2.1 [ohs]” in the step “Templates” and “System Components” in the step “Advanced Configuration”.
- When you choose ‘Next’ an extra configuration option OHS Server will appear
- OHS Server:
 - Enter the details of the Admin Host, where the Node Manager runs. The Admin Port number is not the port where the Node Manager listens (default 5556) but must be a new, unique, port.
 - The Listen Address, the Listen Port and the SSL Listen port specify the address the users will use to access OHI Back Office:
- Assign System Components:
 - Move the system components to the correct Machine
- Skip Deployments Targeting & Services Targeting unless you want to use this.
- Configuration Summary
 - Proceed with ‘Create’ to create the complete domain structure

When you apply the configuration, it usually takes some minutes to execute all the steps.

Make sure you save the configuration results as reported in the configuration report, especially the URL of the Admin Server Console (./console) which can be used to derive the EM Console (./em).

Before doing any further specific environment configuration, you should check the domain installation you just created, by starting the components:



Attention:. Variable ORACLE_HOME is not set by the scripts supplied with Weblogic. The Weblogic scripts use MW_HOME. Only ozg_init.env will set ORACLE_HOME to the same value as MW_HOME.



Attention:. Environment variable DOMAIN_HOME is set by the script \$MW_HOME/user_projects/domains/frs_d2/bin/setDomainEnv.sh and points to \$MW_HOME/user_projects/domains/frs_d2



Attention:. Repeatedly executing \$DOMAIN_HOME/bin/setDomainEnv.sh will keep on expanding EXTRA_JAVA_PROPERTIES with repeated values. You may want to unset EXTRA_JAVA_PROPERTIES before calling \$DOMAIN_HOME/bin/setDomainEnv.sh

- Set the Weblogic environment:
 - . \$MW_HOME/wlserver/server/bin/setWLSEnv.shWhere MW_HOME, if defined, is e.g., /u01/app/oracle/product/frs122119
- Start the Node Manager:
 - . \$MW_HOME/user_projects/domains/frs_d2/bin/startNodeManager.sh

- Start the Admin Server:
`$MW_HOME/user_projects/domains/frs_d2/startWebLogic.sh`
 This will ask for the weblogic password.
- Connect to the Admin Server console at:
`http://<server>:<port>/console`
 Where <port> is the port of the Admin Server, e.g., 7001.
- Start the Managed Server WLS_FORMS.
- Test access without OHS to the WLS_FORMS Managed Server:
`http://<server>:<port>/forms/frmservlet`
 Where <port> is the port of WLS_FORMS, e.g., 9001



Attention:. We have not configured OHS for Forms yet, so the Forms Managed Server is not yet accessible over OHS.



Attention:. The default configuration will only start Java if the browser has the JRE plugin (the JPI, which is no longer supported), so only Internet Explorer can show the default Forms window. Other browsers will display an empty page, but the page source should show you an html file named `basejpi.htm` with parameters for a “Forms applet definition”.

If you encounter issues in the steps above, solve those first. If you can access the console and the Forms Server, shutdown all the components:

- Use the Admin Console to shut down the Forms Managed Server.
- Use the Admin Console to shut down the Admin Server
- Stop the Node Manager:
 - `$DOMAIN_HOME/bin/stopNodeManager.sh`

4.2. CONFIGURE ORACLE HTTP SERVER

In a standard FRS installation, an Oracle HTTP Server (OHS) is installed and configured.

The following settings must be modified in the configuration file for OHI Back Office to work correctly. Restart the HTTP Server after implementing these changes.

This assumes you named the Oracle HTTP Server “ohs1”.

4.2.1. Prepare script startup

To start the HTTP Server from the command line, issue the following command and type the node manager password when prompted.

First time:

```
$DOMAIN_HOME/bin/startComponent.sh ohs1 storeUserConfig
```

Subsequent stop and start calls can be made without the extra parameter and without specifying any node manager password.

- Start HTTP Server:
`$DOMAIN_HOME/bin/startComponent.sh ohs1`

- Stop HTTP Server.

```
$DOMAIN_HOME/bin/stopComponent.sh ohs1
```

The HTTP Server must be started once, for the directories associated with "ohs1" instance to be created.

4.2.2. Register the forms URL

The web tier is not configured by default, so you will need to do the following.

Copy and edit the forms.conf config file supplied by OHI to the "moduleconf" directory under the "ohs1" instance:

```
cp $MW_HOME/forms/templates/config/forms.conf
$DOMAIN_HOME/config/fmwconfig/components/OHS/instances/ohs1/moduleconf/
vi
$DOMAIN_HOME/config/fmwconfig/components/OHS/instances/ohs1/moduleconf/forms.c
onf
```

Activate the section for the `cluster` and change the host and port to match the `WLS_FORMS` properties. If you created more than one Managed Server for Forms, register all of them.

```
<Location /forms>
    SetHandler    weblogic-handler
    WebLogicCluster host1:port1<<,host2:port2>><<,host3:port3>>
    DynamicServerList OFF
</Location>
```



Attention:. After this initial setup, you can use the Oracle Fusion Middleware Control (<hostname>:<AdminServer port>/em) to make further changes to *.conf files present in folder

`$DOMAIN_HOME/config/fmwconfig/components/OHS/instances/ohs1.`

Unfortunately, this does not work for forms.conf which is located in subfolder moduleconf and which is included dynamically.

You can access these *.conf files using the following path:

Target Navigation (Tree icon on the left) -> "HTTP Server" -> ohs1.

From the dropdown "Oracle HTTP server", choose "Administration" > "Advanced Configuration".

On the Advanced Server Configuration page, choose a conf file in the "Choose a File" pull down Menu and press the button "Go".

.

4.2.3. Register Virtual Directories

In the same configuration file

```
$DOMAIN_HOME/config/fmwconfig/components/OHS/instances/ohs1/moduleconf/forms.c
onf, the Virtual Directory Mapping /OHI/ has to be implemented to support the physical
mapping to directory $OZG_ROOT.
```

This Virtual Directory Mapping can subsequently be used in formsweb.cfg for files that need to be downloaded by the user browser, e.g., the `OZG_GIF.jar` file which contains the icons for the button bar in top of the OHI Back Office user interface screens.

Sample setting:

```
Alias /OHI/ "/u01/app/oracle/product/OHI/"
```

4.2.4. Allow access to the Release Documentation

To allow users to view the release documentation (via screen “Raadplegen release informatie” / “Consult release information”), the following settings have to be added in `forms.conf`.

Adapt the location for your installation.

```
# OHI BO: for Release documentation

<Directory "/u01/app/oracle/product/OHI/patch/1?..?..?..*/doc/">
    Options Indexes MultiViews
    AllowOverride None
    Require all granted
</Directory>
```

This code allows access to the `/doc` directory of OHI (patch) releases (always in format 19.99.9.9.9999) so the documentation can be viewed in a browser.



Attention: `forms.conf` is the file you created in the previous step. It is located in the directory

```
$DOMAIN_HOME/config/fmwconfig/components/OHS/instances/ohs1/moduleconf
```

Because of this complex path, we advise creating a symbolic link to it in `$OZG_ADMIN`, after that directory has been created (described later in this document).

4.2.5. Allow access to application components

All files that need to be downloadable via the browser need to have their directory registered with explicit permissions in the Oracle HTTP Server (OHS) configuration files.

Entries are needed for each OHI BackOffice environment if you use explicit environment names. If you use a wildcard for the environment name (replace `/vohi/` by `/*/` in the syntax below) you only have to enable this once for multiple environments.

To allow users to download essential parts of the application the following settings must be added in `forms.conf`. Adapt the location for your installation.

```
# OHI BO: for online help

<Directory "/u01/app/oracle/product/OHI/vohi/help/">
    Options Indexes MultiViews
    AllowOverride None
    Require all granted
</Directory>

# OHI BO: for batch scheduler log files

<Directory "/u01/app/oracle/product/OHI/vohi/log/">
    Options Indexes MultiViews
    AllowOverride None
```

```
    Require all granted
</Directory>
# OHI BO: for batch scheduler output
<Directory "/u01/app/oracle/product/OHI/vohi/out/">
    Options Indexes MultiViews
    AllowOverride None
    Require all granted
</Directory>
# OHI BO: for JNLP and icon file
<Directory "/u01/app/oracle/product/OHI/vohi/utils/">
    Options Indexes MultiViews
    AllowOverride None
    Require all granted
</Directory>
```

In order to have a better readability of .log and .out files in Internet Explorer it is also wise to add lines like these in `forms.conf`.

```
# for better readability of text files in Internet Explorer
<IfModule mime_module>
    AddType text/plain .out
    AddType text/plain .log
</IfModule>
```

4.3. CONFIGURE FORMS SERVER

This paragraph describes how to configure your Forms Server environment.

You may want to have a running WLS_FORMS Managed Server so you can quickly test the changes described in this paragraph. WLS_FORMS does not require a restart for the changes to take effect.

4.3.1. Configure general settings in formsweb.cfg

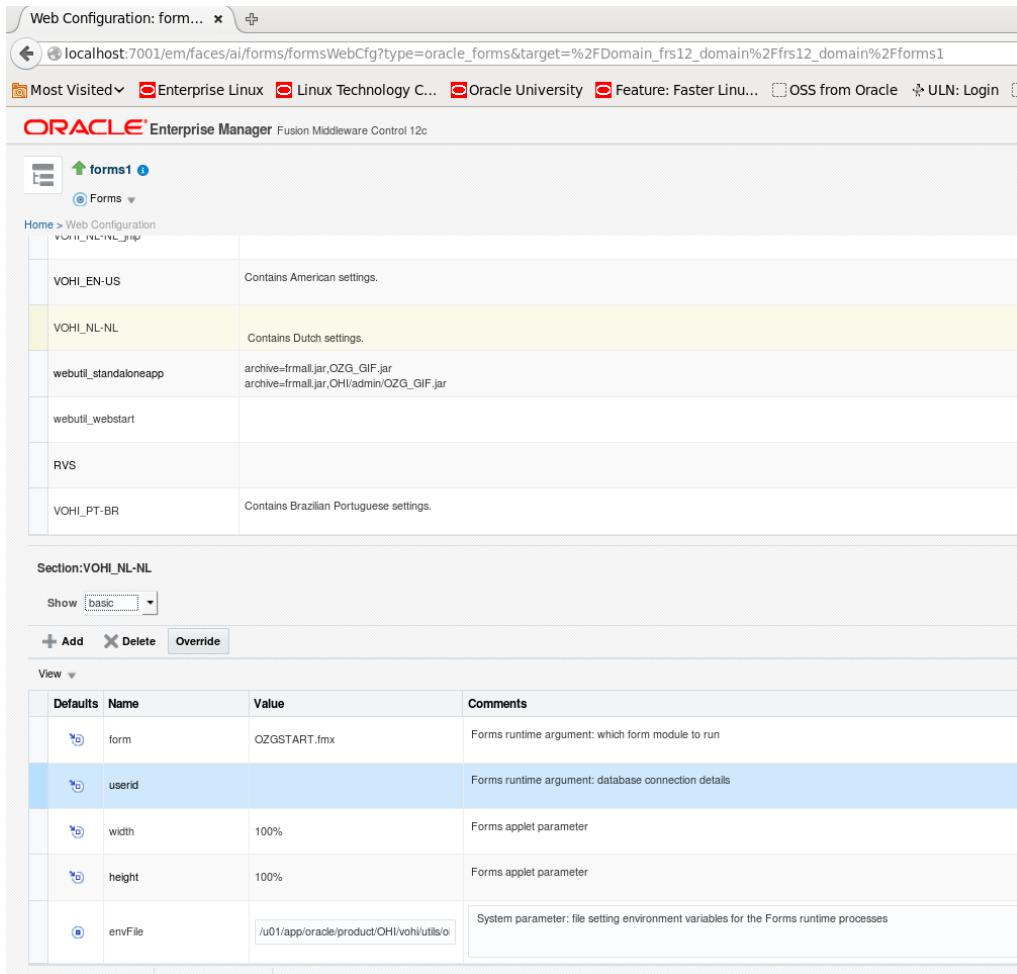
Configuration file `formsweb.cfg` in folder

`$DOMAIN_HOME/config/fmwconfig/servers/WLS_FORMS/applications/formsapp_12.2.1/config` now has to be configured for the environment-specific settings.

This file can be edited manually but it is also possible to change this using Oracle Enterprise Manager Fusion Middleware Control. The URL `<hostname>:<AdminServer port>/em` can be used to start this console.

An example of the page where the configuration parameters can be changed is shown below. The WLS_FORMS server needs to be running.

You can access the page using the following steps: Target Navigation (Tree icon) -> Forms -> forms1. From the dropdown “Forms”, choose “Web Configuration”.



Web Configuration: form... x

localhost:7001/em/faces/ai/forms/formsWebCfg?type=oracle_forms&target=%2FDomain_frs12_domain%2Ffrs12_domain%2Fforms1

Most Visited Enterprise Linux Linux Technology C... Oracle University Feature: Faster Linu... OSS from Oracle ULN: Login

ORACLE Enterprise Manager Fusion Middleware Control 12c

forms1

Forms

Home > Web Configuration

Configuration	Description
VOHI_EN-US	Contains American settings.
VOHI_NL-NL	Contains Dutch settings.
webutil_standaloneapp	archive=frmall.jar,OZG_GIF.jar archive=frmall.jar,OHI/admin/OZG_GIF.jar
webutil_webstart	
RVS	
VOHI_PT-BR	Contains Brazilian Portuguese settings.

Section: VOHI_NL-NL

Show: basic

+ Add - Delete Override

View

Defaults	Name	Value	Comments
	form	OZGSTART.fmx	Forms runtime argument: which form module to run
	userid		Forms runtime argument: database connection details
	width	100%	Forms applet parameter
	height	100%	Forms applet parameter
	envFile	/u01/app/oracle/product/OHI/vohi/utls/o	System parameter: file setting environment variables for the Forms runtime processes

The <default> configuration will be inherited by all other configurations, so generic changes can be added over there. Specific configurations with different settings can be created to support different environment settings, for example to run with different colors, a different page title or other different characteristics as described later.

The settings themselves are logically grouped together. The group name used in the screen will be used to help in finding where to specify the setting.



Attention: When upgrading an existing OHI Back Office environment, do not copy the existing formsweb.cfg from your old environment. Always use the file formsweb.cfg as delivered with the FRS installation as a starting point and merge your changes from the existing old environment into it, either by editing the file on the Operating System or by using the console.



Attention: You need to remove the parameter `serverArgs` from the <default> configuration (or comment it out directly in the file). If you forget this specific startup actions will not work and Java Web Start will not load the icons for the button bar.



Attention: directories specified in formsweb.cfg must not contain logical names. So do not use e.g., `$OZG_BASE`, but e.g.,
`/u01/app/oracle/product/OHI/vohi/`

The table below shows the parameters which require a setting for the use of OHI Back Office. These can best be specified in the default configuration because they are usually identical for

each OHI BO environment. You can always override them for configurations that need a different setting. Be sure you adapt file paths and names to your actual server setup!

The column “Group” is only relevant when using the WebLogic console.

Parameter name	Group	Value	Comment
form	Basic	OZGSTART.fmx	First window to open when forms application is started (fixed value).
term	advanced	/u01/app/oracle/product/OHI/vohi/utills/ohirf34w_nl-NL.res	This defines which keyboard mapping file to use. In the example it is defined as ohirf34w_nl-NL.res (delivered as default configuration resource file for Dutch key descriptions). Use a definition file for the language you like to use. OHI delivers some sample mapping files in \$OZG_BASE/conf/Back-Office which you should copy manually to your \$OZG_BASE/utills folder and possibly adapt when needed. These standard files may be updated when you install a new OHI release. When you want to use different mappings for different environments, please use the “term” setting as documented in the next paragraph.

The following parameters are optional. Values are given but are not required. Normally these settings are applied to the default configuration, but you can implement these differently for each (OHI BO) environment. This is typically done by implementing different sections (called “Named configurations”) within the forms web configuration (`formsweb.cfg`), which is done in the next paragraph.

Parameter name	Group	Value	Comment
splashScreen	applet	no	Specifies the .GIF file that should appear before the applet appears. Set to NO for no splash. Leave empty to use the default splash image.
logo	applet	no	Specifies the .GIF file that should appear in the righthand corner of the Forms menu bar. Set to NO for no logo. Leave empty to use the default Oracle logo.
background	applet	no	Specifies the .GIF file that should appear in the background. Set to NO for no background. Leave empty to use the default background.
separateFrame	applet	false	If set to false, the applet will appear within the browser window. If set to true, the applet will appear as separate MDI window.
width	applet	100%	Default width of the applet. A percentage value uses all available space on your screen. Values above 100% will activate a scrollbar. Omitting the percentage sign means a size in pixels.
height	applet	99%	Default height of the applet. Values above 100% will activate a scrollbar.

Parameter name	Group	Value	Comment
dontTruncateTabs		False	Specify true if you do want tab names to be written in full (meaning less tabs are visible when there are many)
clientDPI	<unlisted>	100	100 enforces the standard size (compatible with earlier releases). <u>If not set the screens are a little smaller than in older releases which may result in prompts or characters partially invisible, also in 12c.</u>
colorScheme	applet	swan	Teal, Titanium, Red, Khaki, Blue, Olive, or Purple. New: BLAF, swan
centerOnStartup	<unlisted>	False	When true the Forms application window will be centered when started.
isResizable	<unlisted>	True	When false the main application window cannot be resized.
alwaysOnTop	<unlisted>	False	When true the Forms application window will stay on top of all windows and cannot be hidden.
clientDPIRatio	advanced	80	Can be used to scale, by means of a percentage, the application window based on the display size (may be useful with extreme resolutions on large screens). Ignored when clientDPI is specified.

4.3.2. Configure EnvironmentSpecific Settings in formsweb.cfg

Customer and/or OHI BO environment-specific entries in formsweb.cfg have to be set up depending on the actual situation.

Create a new section for each environment. This new section (commonly called a “named configuration”), which inherits the settings from the default section, will be referred to in the URL the user uses to start the application.

Override at least the following variables:

Parameter name	Group	Value	Comment
envFile	basic	/u01/app/oracle/product/OHI/vohi/ohi_servlet_wls_VOHI_NL-NL.env	Specifies the environment specific file that contains environment specific environment settings. FORMS_PATH and NLS_LANG are examples. It is wise to store these files in the \$OZG_BASE location so they are cloned with the OHI environment.
formParams	<unlisted>	p_mdi_window_label=OHIVM	OHI specific variable which is used in the title of the MDI window (only visible if separateFrame=true). Maximum length for the label is 10 characters.
pageTitle	html	OHI Test Environment	The title for the browser page which is used for this environment.
term	advanced	/u01/app/oracle/product/OHI/vohi/ohi_rf34w_nl-NL.res	Optional: When a specific keyboard mapping is needed per environment, specify this in the term parameter. You can create language specific keyboard mappings per environment in this way.
workingDirectory	advanced	/u01/app/oracle/product/OHI/vohi	The directory should specify the same folder as \$OZG_BASE. This is necessary for the online help and for tracing purposes.

When using the WebLogic console, be sure to apply each separate change to prevent losing them when switching to a new group of settings.

4.3.2.1. Java Web Start

As described in Chapter 1, OHI Back Office now only supports Java Web Start to run the OHI Forms application on the Client Tier.

The Java Web Start mode requires some extra configuration parameters in formsweb.cfg:

Parameter name	Group	Value	Comment
basejnlp	--	/u01/app/oracle/product/OHI/vohi/ohi_utils/OHI_WEBSTART.jnlp	File as delivered with OHI BO in \$OZG_BASE/ohi_utils.
webstart		enabled	Enable Java Web Start
envVirtualFolder		/OHI/vohi	Maps the OHS virtual folder for the environment. This variable is only used by OHI_WEBSTART.jnlp.

The file OHI_WEBSTART.jnlp is delivered with OHI Back Office and is based on \$DOMAIN_HOME/config/fmwconfig/components/FORMS/instances/forms1/server/base.jnlp. It references a file OHI_EXTENSIONS.jnlp which is also delivered. Both files are located in \$OZG_BASE/ohi_utils.

The OHI Back Office user interface can be run outside the browser through a link in the browser with the URL that refers to the named configuration in formsweb.cfg that contains the above parameters, with webstart=enabled:

```
http://<server>:<port>/forms/frmservlet?config=<config name>
```

After the Java application has started, the browser window can be closed or re-used for other purposes.

You can even start the user interface outside the browser by the command below, embedded in a command file or in a shortcut:

```
javaws http://<server>:<port>/forms/frmservlet?config=<config name>
```

This simply uses the javaws executable from the local Java Runtime Environment (JRE). The shortcut expects javaws can be found in a folder specified in the PATH environment variable on the Windows client.

4.3.2.2. Accessibility

To make OHI Back Office more accessible for the visually impaired, a parameter was introduced in “Theme” M-3053 in 2012. When this parameter is set, the OHI session will no longer skip the read-only fields during keyboard navigation. That way, a screen reader program will make the user aware of the existence of those fields.

This parameter can be set in two different ways:

1. The user adds the parameter to the URL for the application:

```
http://<server>:<port>/forms/frmservlet?config=<config name>&p_navigate_all=Y
```

2. The user uses a separate configuration in the URL for the application:

```
http://<server>:<port>/forms/frmservlet?config=<config name2>
```

For both options, configuration is needed in the file formsweb.cfg

For option 1:

Add p_navigate_all to "otherparams" setting in the default section:

```
otherparams=obr=%obr% record=%record% tracegroup=%tracegroup% log=%log%
term=%term% ssoProxyConnect=%ssoProxyConnect% p_navigate_all=%p_navigate_all%
```

Of course, this setting can also be made for specific named configurations only, by copying the original setting for otherparams from the default section to the named configuration sections you want to modify and adding the p_navigate_all=%p_navigate_all% only there.

For option 2:

Create a new named configuration <config name2> section by copying an existing section, and add the following line in that section:

```
otherparams=obr=%obr% record=%record% tracegroup=%tracegroup% log=%log%
term=%term% ssoProxyConnect=%ssoProxyConnect% p_navigate_all=Y
```

4.3.3. Configure Environment variables in *.env files

□ Attention:



Attention: When upgrading an existing OHI Back Office environment, do not simply copy your existing <environment>.env from your old environment. Always use the file default.env as delivered with the FRS installation as a template. For each environment, merge a copy of this template with your changes from the existing old environment, either by editing the file on the Operating System or by using the console.



Attention: Environment variable ORACLE_INSTANCE is no longer used.



Attention: directories specified in *.env files must not contain logical names. So do not use e.g., \$OZG_BASE, but e.g., /u01/app/oracle/product/OHI/vohi/

An environment file as indicated by the envFile setting defines at least the environment specific values for the following environment variables:

```
#-----  
TWO_TASK=vohi  
#-----  
#  
#-----  
OZG_LOG=/u01/app/oracle/product/OHI/vohi/log  
OZG_OUT=/u01/app/oracle/product/OHI/vohi/out  
#-----  
#  
#-----  
FORMS_PATH=/u01/app/oracle/product/OHI/vohi/bin  
FORMS_TRACE_DIR=/u01/app/oracle/product/OHI/vohi  
# Specify which language, territory settings and  
# character set to use  
NLS_LANG=DUTCH_THE_NETHERLANDS.WE8MSWIN1252  
#-----
```

You need to create an *.env file for each environment c.q. envFile setting in formsweb.cfg. Each environment file should be based on the default.env file which is delivered in folder:

```
$DOMAIN_HOME/config/fmwconfig/servers/WLS_FORMS/applications/formsapp_  
12.2.1/config
```

In the example setting above, a file \$OZG_BASE/ohi_servlet_wls_VOHI_NL-NL.env is referenced. That file contains settings for Dutch-speaking users.

From the settings copied from default.env you should normally comment out:

```
ORACLE_HOME  
ORACLE_INSTANCE  
TNS_ADMIN (should already be set as environment variable for user oracle)  
FORMS_PATH (set it at the bottom of the file)  
FORMS_MODULE_PATH (set it at the bottom of the file)  
PATH  
WEBUTIL_CONFIG (not used by OHI)
```

The value of FORMS_MODULE_PATH should be set to \$FORMS_PATH to limit the search path for Forms executables, or the line should be commented. If FORMS_MODULE_PATH is set to \$FORMS_PATH, place the line after the line with FORMS_PATH.

The value of FORMS_RESTRICT_ENTER_QUERY depends on the security level that is required in your company. Setting it to FALSE allows end users to add SQL expressions to the filters in Forms that are executed by the “Execute Query” action. This can facilitate SQL injection.

You can specify other environment settings in the .env file, like a different value for NLS_LANG in case a multi-language environment is configured. For more information about values for NLS_LANG please see the paragraph about [Localization Support](#) and the paragraph about setting up the [NLS_LANG](#) variable.

4.3.3.1. Cancel long running queries

When you want to offer functionality to the Forms users to cancel long running queries, a set of three environment variables can be used to influence this behavior:

```
FORMS_LOV_INITIAL=10000
```

```
FORMS_LOV_MINIMUM=1500
```

```
FORMS_LOV_WEIGHT=0
```

The first two variables are in milliseconds and the values as described are already present in the `ozg_init.env` template file provided. However, you might want to change this for a specific environment by overruling them in that environments' `.env` file.

The first parameter specifies how long a query should at least run before the cancel query dialogue button appears. The second setting determines a minimum polling interval to check whether the query already provided an answer.

The third parameter specifies how to prolong the time between the subsequent Cancel Query pollings. `FORMS_LOV_WEIGHT` has an effect in proportion to the average speed of a round trip between the Java client and Forms server. Therefore, the slower the round trips are, the longer a particular value of `FORMS_LOV_WEIGHT` will delay making another time-consuming round trip to the Cancel Query window.

For much more detail about these and some other settings please read MOS document 138159.1.

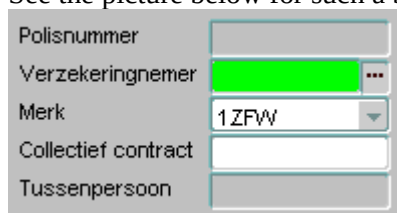
4.3.4. Configure Registry.dat

Using Oracle Enterprise Manager Fusion Middleware Control, it is possible to specify specific aspects of the look and feel of the application. The page is started from the dropdown “Forms”, option “Font and Icon Mapping”. The file is located at

```
$DOMAIN_HOME/config/fmwconfig/servers/WLS_FORMS/applications/formsapp_12.2.1/config/oracle/forms/registry/Registry.dat
```

You can define whether an LOV button should appear at the right side of an item when a list of values function is defined for this item.

See the picture below for such a button on the active second field:



To do so, specify `true` for the `app.ui.lovButtons` setting.

Be aware that the users need to know about this setting because the look and feel slightly changes. Also note that the button may temporarily cover part of a directly following field when the cursor is in the field.

On the same page c.q. in the same file, the required field visual attribute should be set to highlight **required fields**:

Please specify `true` for the `app.ui.requiredFieldVA` setting.

Adapt the color setting, for example to: 255,242,203

The result in the file will be:

```
app.ui.lovButtons=true
app.ui.requiredFieldVA=true
#      The background color is specified as an RGB triple.
app.ui.requiredFieldVABGColor=255,242,203
```

4.3.5. Using extra startup parameters

Suppose the section in formsweb.cfg is named VOHI_NL-NL. These environment settings are applied within Forms by using a URL like:

```
http://myhost.example.com:8890/forms/frmservlet?config=VOHI_NL-NL
```

If you want to add additional startup parameters to pass to the startup form you can use the parameters formParams, formParam1 and formParam2. Note that formParams, formParam1 and formParam2 have been added in the serverArgs part of OHI_WEBSTART.htm.

In this way you can specify a form and data context to start a specific form and query data in that form. You need to specify both the form and the data by means of parameters p_context_form and p_context_keys.

A fictitious call could be like this one:

```
http://myhost.example.com:7777/forms/frmservlet?config=VOHI_NL-
NL&formParam1=p_context_form=REL1234F&formParam2=p_context_keys=bl1_co
de=AB;bl1_nr=99
```

Note that this specifies values for multiple fields, separated by a comma.

A more realistic example based on the screen to maintain relations:

```
http://myhost.example.com:7777/forms/frmservlet?config=VOHI_NL-
NL&formParam1=p_context_form=REL1001F&formParam2=p_context_keys=rel_nr
=2218540200
```

Within OHI Back Office, the button 'Module Info' on the Info screen will show a list of potential queryable fields. Block name and field name should be concatenated with an underscore.

If you use these startup parameters to enter a condition for a field that is not a primary key, OHI Back Office will show a warning that the specified condition can slow down the query. That warning can be suppressed by adding the string '^!^' to the condition value.

Another example is shown below, where the condition on block rel field wg_nummer does not result in a warning because of the postfix added to the fictitious value ALIN1-7:

```
http://myhost.example.com:7777/forms/frmservlet?config=VOHI_NL-
NL&formParam1=p_context_form=REL1001F&formParam2=p_context_keys=rel wg
nummer=ALIN1-7^!^
```

4.4. CONFIGURE SECURITY AND MANAGE STARTUP/

SHUTDOWN

For production environments some basic actions should be executed to ensure a minimum level of robustness and security.

At least the communication with the Admin console should be encrypted and the startup and shutdown should be automated. Automatic restart of servers in case of crashes is a requirement.

In principle you should use the Node Manager process to start WebLogic Server processes. This should be configured in such a way that these servers will be automatically restarted in case of failures of processes. The steps for accomplishing that are described below as guidance, but there are other ways to do this.

If you experience problems (mainly hanging shutdown actions) when using the Node Manager to shutdown components, an alternative can be to connect to the Admin Server to execute shutdown commands.



Attention: When you get errors during startup of the WLS_FORMS server and the server log file reports authentication errors, please startup manually by connecting to the AdminServer through WLST and issue the `start('WLS_FORMS')` command. This will update the files in `$DOMAIN_HOME/servers/<SERVER>/data/nodemanager` directory. These files may be outdated.

4.4.1. Node Manager - Additional User Definition

We advise setting up an additional user for the Node Manager process. This is especially useful for potential integration with Enterprise Manager. The user name is typically “nodemanager”.

Make sure you have a running Admin Server (use `$DOMAIN_HOME/bin/startWebLogic.sh` if needed). The port is the Admin Server port (default 7001, but it is probably different for your environment).

If you did not create the additional user during the domain creation Wizard, these are the steps to create one later:

```
. ozg_init.env $OZG_ORATAB_FRS122119
$MW_HOME/oracle_common/common/bin/wlst.sh
connect('weblogic','<pw>','t3://<host>:<port>')
edit()
startEdit()
secConfig = cmo.getSecurityConfiguration()
secConfig.setNodeManagerUsername('nodemanager')
secConfig.setNodeManagerPassword('<nodemanager pwd>')
save()
activate()
disconnect()
exit()
```

Make a note of the username and password.

4.4.2. Node Manager - Change Properties File

In order to enable automatic restart of failed/failing Admin or Managed servers you should enable crash recovery for the Node Manager.

This is done by setting the property 'CrashRecoveryEnabled'=true

in \$DOMAIN_HOME/nodemanager/nodemanager.properties (if you use the domain based Node Manager).

To execute additional actions before starting WebLogic Servers or after stopping them, you can overrule the default properties of

'StartScriptEnabled' and 'StartScriptName' and

'StopScriptEnabled' and 'StopScriptName'

The default start script 'startWebLogic.sh' will implement the necessary domain environment settings.

4.4.3. Secure Storing of Credentials

You should avoid specifying unencrypted usernames and passwords in start and stop scripts. Instead, store the weblogic and nodemanager credentials in a set of secure files.

The commands below save the credentials for both user definitions in two files per user (a credential file and an accompanying key file).

User weblogic:

Encrypted in file \$OZG_ADMIN/weblogic_frs_d2-WLSConfig.properties which can be decrypted by WLST using the key file \$OZG_ADMIN/weblogic_frs_d2-WLSKey.properties.

User nodemanager:

Encrypted in file \$OZG_ADMIN/nodemanager-WLSConfig.properties which can be decrypted by WLST using the key file \$OZG_ADMIN/nodemanager-WLSKey.properties.



Attention: If you are running multiple domains on the same server (possibly from another version of WebLogic), be sure to choose different file names for the domains. Don't overwrite the files for the other Domain. Also make sure the files you specify do not exist, or you will get an error message that is hard to interpret

Run the commands below while you have an Admin Server process running. The port is the Admin Server port (typically 7001).

```
. ozg_init.env $OZG_ORATAB_FRS122119
$MW_HOME/oracle_common/common/bin/wlst.sh
lvOZGAdmin=os.environ.get("OZG_ADMIN")
connect('weblogic','<pw>','t3://<host>:<port>')
storeUserConfig(userConfigFile=lvOZGAdmin+'/weblogic_frs_d2-
WLSConfig.properties',userKeyFile=lvOZGAdmin+'/weblogic_frs_d2-
WLSKey.properties')
disconnect()
exit()
```

Run the commands below while you have a Node Manager process running (use

`$DOMAIN_HOME/bin/startNodeManager.sh` for starting).

```
. ozg_init.env $OZG_ORATAB_FRS122119
$MW_HOME/oracle_common/common/bin/wlst.sh
lvDomainHome=os.environ.get("DOMAIN_HOME")
lvDomainName=os.environ.get("DOMAIN_NAME")
lvOZGAdmin=os.environ.get("OZG_ADMIN")
nmConnect('nodemanager','<pw>','<host>','<port>',lvDomainName,lvDomain
Home)
storeUserConfig(userConfigFile=lvOZGAdmin+'/nodemanager-
WLSConfig.properties',userKeyFile=lvOZGAdmin+'/nodemanager-
WLSKey.properties',nm='true')
nmDisconnect()
exit()
```

To make sure the Node Manager can start the Admin Server you need to store the credentials of the weblogic user once more: store them in the file

`$DOMAIN_HOME/servers/AdminServer/security/boot.properties`.

Please create this file and add two lines as shown:

```
username=weblogic
password=<Password>
```

The first time the Admin Server is started this file will be encrypted. As a result of this the credentials of the weblogic user are now stored in two locations. Be aware of that when you change the password.

4.4.4. Securing Console Communication

In order to protect your console access, consider securing the communication to the Admin console, by only allowing communication over the https port, using either https (for interactive access from the browser) or the t3s protocol (for WLST scripts). This is described in the WebLogic Server documentation.

4.5. IMPROVING STARTUP TIME OF WEBLOGIC

You may experience intermittent delays when starting the WebLogic environment. This can be caused by the Random Number Generation functionality which may block until sufficient “noise” is generated (in fact an Entropy issue occurs). If your environment exhibits this behavior, you can test by multiple executions of the line below where random info is read from the random “generator”. If one or more of these calls experience delays, you may change to using `/dev/urandom`. There is a lot of discussion whether this is less secure or not so if you are not sure do not change this in a sensitive environment (production).

```
head -n 1 /dev/random > tmp.tmp
```

For more information about this topic please see MOS document 1574979.1.

4.6. ADDING MANAGED SERVERS TO 'CLUSTER_FORMS'

If you have chosen a cluster setup during the configuration of the Forms services, you will have a Weblogic cluster named "cluster_forms".

In the EM console (<http://<server>:<adminport>/em>) this will show up like:

The screenshot displays the Oracle Enterprise Manager Fusion Middleware console. The top section shows the 'Target Navigation' tree with the following structure:

- Application Deployments
- WebLogic Domain
 - frs12_domain (selected)
 - AdminServer
 - cluster_forms
- Forms
 - forms1
- HTTP Server
 - ohs1

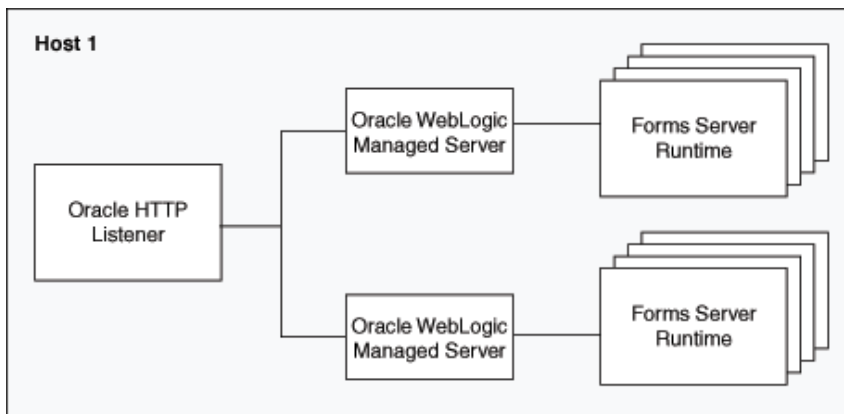
The bottom section shows the 'Servers' page for the 'frs12_domain'. It includes a summary of servers (1 Down, 1 Up) and a table of servers:

Name	Status	Cluster	Machine	State
AdminServer(admin)	Up			Running
WLS_FORMS	Down	cluster_forms	AdminServerMachine	Shutdown

Name	Status	Cluster Type	Servers	Cluster Messaging Mode
cluster_forms	↓	Configured	WLS_FORMS	Unicast

Your user community may generate more load than one Managed Server can handle with the default setup. The steps for tuning the Forms Listener servlet are similar to steps for tuning any high throughput servlet application. You have to take into account resource management and user needs for optimal tuning of your particular Forms Services configuration. For more information, see Oracle Fusion Middleware Performance Guide.

If tuning does not solve the issues, one of the options is setting up additional Managed Servers and load balancing the requests. This is described in the Forms Services Deployment Guide. A potential configuration is shown below:



Additional Managed Servers can be added to the cluster 'cluster_forms' in several ways, e.g.

- creating a new server manually and adding it to the cluster
- cloning an existing Managed Server, using the Clone option in the Control tab in the Administration Console:

Name	Type	Cluster
AdminServer(admin)	Configured	
WLS_FORMS	Configured	cluster_forms

You need to specify a new name and port for the new Managed Server.



Attention: In the WebLogic Server Basic license, the use of the WebLogic Server Administration Console for cloning a Managed Server instance is not permitted.

When the new Managed Server has been added to the cluster it can be started. Make sure to add it to your start/stop scripts and any monitoring tools you use.

To make sure the requests are balanced over the available Managed Servers the Oracle HTTP Server settings have to be adapted in

```
$DOMAIN_HOME/config/fmwconfig/components/OHS/instances/ohs1/moduleconf/  
/forms.conf.
```

Please change the WebLogicCluster line and add the additional server(s) like in the example below:

```
<Location /forms>  
    SetHandler weblogic-handler  
  
    WebLogicCluster host1:port1,host2:port2  
  
    DynamicServerList OFF  
</Location>
```

Of course, you need to restart the OHS afterwards to enable these settings.

4.7. UNDERSTANDING BATCH & BACKGROUND PROCESSING

This section briefly describes the functioning of the OHI Back Office batch scheduler and the OHI-related background processes that use the database scheduler.

4.7.1. Batch scheduler

The OHI Back Office batch scheduler is a Pro*C daemon (a continuous process), implemented as a single process that runs on the application server. If there is more than one application server it is possible to start a batch scheduler process per application server (please see [this paragraph](#) for more information regarding this).

The process is implemented by an executable named SYSS004S.

Scheduled ‘batch’ processes

The OHI Back Office batch scheduler *iteratively* performs the following activities for the OHI batch scripts requested in the Back Office application:

- Starting the new script requests
- Checking the ongoing script requests

This process of starting and checking can be influenced by means of a number of settings in the application (screen System/Management/General/System parameters (in Dutch: *Systeem/Beheer/Algemeen/Systeemparemeter*, tab page *Batch scheduler*).

One of the settings is the *Polling interval*; during this period the batch scheduler “sleeps” and will only be woken up by a newly created request. Scheduled script requests will not wake up the batch scheduler and will therefore not be started until the polling interval has passed.

1 - Starting new script requests

When incoming script requests (newly created or already planned) have to be started, first there is a check to see if the maximum number of parallel script requests has been reached or not.

In order to determine this number (to be set up in OHI Back Office by means of *Max. parallel script requests*) the number of script requests with status *Start* and *Running* is summed.

If this maximum number has not yet been reached, the new script request can be started.

Note: the batch scheduler will always keep some spare capacity to honor sub-process requests to avoid a deadlock situation where running master processes cannot finish because their sub-processes do not start.

If the maximum number has been reached, the script request to be started will still have status *Waiting*.

The remaining “capacity” (this means the number of script requests that can still be started) can be retrieved in the log file of the batch scheduler (`$OZG_LOG/<batch account>_<sid>_<host>.log`) when it runs in *verbose* mode.

If no script requests are created in the *Polling interval* period, and if therefore no new script requests have to be started, the batch scheduler will proceed with activity 2: check the ongoing script requests.

2 - Checking the ongoing script requests

Regarding the started processes (= "ongoing script requests") 2 matters are checked:

- Has the script request been started within the margin?
- Is a script request with status "Running" (ongoing) indeed still active?

Re 1. Has the script request been started within the margin?

If the batch scheduler observes a script request in table `ALG_SCRIPT_AANVRAGEN` waiting to be started, then the status will have the value *W* (*Waiting*).

If the request has to be started, then the batch scheduler will change the status to *Start* and the process will be started on the OS.

The data of the started process can be found in the log file of the batch scheduler if it runs in *verbose* mode.

An example of a script request process of the type ‘sqlplus’:

```
15:31:12:nohup nice -19 $OZG_BASE/utils/OHI_CMD.pl sqlplus -l -s
/ @$OZG_BASE/sql/ZRG4032S $OZG_OUT/manager/12779968.out 12779968
>$OZG_LOG/manager/12779968.log 2>&1 & echo $!
```

The script *itself* will (by means of the generic startup code) set the status of the relevant script request to *Running* in table `ALG_SCRIPT_AANVRAGEN`.

If the status transition *Start*->*Running* has not occurred within the configured period (batch scheduler setting *Start delay*), the "Bijzonderheden" (Details) in the UI window for managing script requests, will display the following message “*job niet gestart binnen marge*” (job not started within the margin). The status of the script request will then be set to “*Failed*”.

In most cases, the script will have been started: the process has been started already, but it simply has not reached the startup code yet within the specified “*start delay*”.

Once the script completes, it will set the status of the relevant script request to *Ready* or *Error* (= functional error).

Any **ERROR** message “*job not started within the margin*” will then be changed into an **INFO** message.

What are the possible causes of a script not starting within the margin?

- **Start delay too low**
This setting has been configured so low that the server cannot activate the script fast enough.

- **Server capacity insufficient**

The server is loaded so heavily that it takes long before a script is activated.

- **Privilege errors, etc.**

The script cannot be executed on the application server, it cannot log in to the database, it does not have the required execute privileges in the database, etc. so the status cannot be set to “Running”.

Re 2. Is a script request with status "Running" indeed still active?

If the script request has status `Running`, then the batch scheduler will perform a check on the OS or in the database (depending on the type of script) to ensure that a process is indeed still active for this script request.

If it turns out that this process is no longer active, then the script request is considered to have `Failed`; the process has not been finalized correctly (i.e., the end code in which the status is set to `Ready` or `Error` has not been executed).

At the end of this check phase, the batch scheduler will check if the Oracle Scheduler job for processing real time events is still running. If not, it will be restarted.

After performing the above checks (= activity 2 – *Checking the ongoing script requests*) the batch scheduler will proceed with activity 1 (*Starting new script requests*), etc.

4.7.2. OHI Background processing

The OHI Back Office batch scheduler is ‘assisted’ by Oracle Scheduler database job processes: job scheduler processes executing OHI related background maintenance tasks and event processing tasks are implemented with the standard job scheduler provided in the Oracle database.

OHI uses several scheduler jobs:

Name	Description
OHI_MAINTENANCE_JOB_P	A master job that controls the other OHI job executions. This job is always active.
OHI_EVENT_JOB_x_P	Process business event framework (BEF) handlers for near real time events. At least one job is always active; more than one job can be active. The number of jobs is determined by a Back Office parameter and can be dynamically adjusted with the parameter value. When the master job executes housekeeping (normally each batch scheduler polling interval), the changed value is picked up.
OHI_INTERNAL_EVENT_JOB_x_P	Process internal event handlers for the ‘autonomous processing events’ framework (AVF, as abbreviation for the Dutch name). At least one job is always active; more than one job can be active, as determined by a Back Office parameter
OHI_EVOLVE_SPM_JOB_P	Evolve SQL Plan Baselines as created when using SQL Plan Management (for more information about using SQL Plan Management please see information later in this manual).
OHI_SYNC_REDEF_JOB_P	Synchronize the data in tables that are in a table redefinition process.
OHI_MONITOR_EVN_JOB_P	Monitor event processing and compress event data for reporting

OHI_PURGE_BLG_JOB_P	Purge payment messages in FSA#BET_VERKEER_REGEL_LOGGING (alias BLG) that passed the retention period (specified by a Back Office parameter).
OHI_PURGE_EON_EVD_JOB_P	Purge event data for receivables and receipts (“Vorderingen” and “Ontvangsten”).
OHI_PURGE_EVN_JOB_P	Purge data of Event processing
OHI_PURGE_MEL_JOB_P	Purge messages from the event handlers (stored in ALG#MELDINGEN, table alias MEL, identified by column EDE_ID not being null) that passed the retention period (specified by Back Office parameter with name “RetentieMEL” and description “Aantal dagen retentie meldingen”).
OHI_PURGE_OWM_JOB_P	Purge records from table VER#PSO_MUTATIES_INCASSOWIJZE that are older than 3 months
OHI_PURGE_RFL_JOB_P	Purge records from table RBH#REL_FORMS_LOGGING that are older than the retention period (specified by Back Office parameter with name “RetentieRFL” and description “Aantal dagen retentie logging pers. geg.”).
OHI_PURGE_SVH_JOB_P	Cleanup redundant/obsolete units of work of the batch scheduler (for jobs that did not finish all their collected work units due to a time constraint or failure).
OHI_PURGE_TLG_JOB_P	Purge trace messages from ALG#TRACE_SESSION (alias TSS) and ALG#TRACE_LOG (alias TLG) that passed the retention period (specified by Back Office parameter with name “RetentieTLG” and description “Aantal dagen retentie debugregels”).
OHI_PURGE_UOC_JOB_P	Purge records from table ALG#UOR_OPR_CONTEXT that passed the retention period (specified by Back Office parameter with name “RetentieOPC” and description “Retentie REST-service-context”).
OHI_PURGE_UOR_JOB_P	Purge records from tables ALG#UID_OHI_REFERENTIES and ALG#UID_WAARDEN that are older than 40 days and are no longer linked to an OHI object.
OHI_PURGE_VBN_JOB_P	Purge records from table ALG#VECOZO_VSP_BERICHTEN that passed the retention period (specified by several Back Office parameters, with names “RetentieVBN_BINAIR”, “RetentieVBN_DATA” and “RetentieVBN_MISLUKT” and descriptions “Retentie binaire VECOZO berichten”, “Retentie verwerkte VECOZO berichten” and “Retentie niet verwerkte VECOZO berichten” respectively).

You can query these jobs (and their STATE and, when applicable, their REPEAT_INTERVAL) through the USER_SCHEDULER_JOBS view in the OHI application owner account. Or use view DBA_SCHEDULER_JOBS and filter on OWNER= the application owner.

When the OHI Back Office batch scheduler starts, these scheduler jobs are started or scheduled when not yet present.

When the stop command is issued for the batch scheduler, the trigger on the updated record in `ALG_BATCH_SCHEDULERS` will issue the stop command for all background jobs. So be sure a graceful stop is executed by calling `OZG_STOP_BATCH.sh` if these jobs need to be prevented from starting on their next intervals.

All these jobs use the `OHI_MAIN` job class. By default no service name is assigned to this job class, meaning the jobs will use the default service name, typically being `SYSDATABASE`.

To determine which scheduler jobs are running with which `service_name` you can execute the query below:

```
select action, service_name, username, schemaname, module
from v$session
where username = 'BATCH'
and module = 'DBMS_SCHEDULER'
order by action;
```

In a RAC environment the `SYSDATABASE` service will typically be running on multiple instances. In order to run these background jobs on a specific instance you should configure a database service that runs on a specific instance and assign that service to the `OHI_MAIN` job class with a statement like below:

```
begin
  dbms_scheduler.set_attribute
    ( name      => 'SYS.OHI_MAIN'
    , attribute => 'service'
    , value     => 'MYSERVICE'
    );
end;
```

To assign the service where your current session is running, use for value:

```
sys_context('userenv', 'service_name')
```



Attention: Due to a bug in database 19c which is still present in Release Update 19.13 make sure you apply the interim patch for the bug below:

31743698 : DBMS_SCHEDULER JOBS GET STUCK IN SCHEDULED STATE AND NEVER RUN

Without this patch the scheduler jobs may become inactive when a service name is assigned to the `OHI_MAIN` job class

4.7.3. Different types of batches

The scripts started by the batch scheduler can be of different types, such as a *SQL module*, *Perl script* or *OS shell script*.

4.7.4. See also

For tuning the batch scheduler see the explanation in '[batch scheduler settings](#)'.

5. INSTALLATION OHI BACK OFFICE APPLICATION SOFTWARE

Installation of the OHI Back Office application software consists of the following steps:

5.1. CREATING OS ACCOUNTS & AUTHORIZATION

5.1.1. Overview of Operating System accounts

OHI Back Office requires two Operating System accounts on the Application Server:

- An account that owns the OHI software tree `$OZG_BASE`. This **MUST** be the same account that owns the Oracle Software (WebLogic, Forms and Database Client). Usually, this account is named `oracle`. This account is used to start OHI Release installations using `OHIPATCH` and needs to be able to connect to the OHI database with the database account of the application owner (usually `OZG_OWNER`).
- An account that starts the OHI Batch Scheduler. Usually, this account is named `BATCH`. This account needs to have limited access to most of the contents of `$OZG_BASE` (Read and Execute) but will write log files and output files in the directories `$OZG_OUT` and `$OZG_LOG` and their subdirectories. This account also needs to log on to the database with the database account for the batch user (usually also named `BATCH`).

To support the automatic login to the database by `OHIPATCH` and the Batch Scheduler process, two wallets are needed. See later, Creating wallets.

Within the OHI Back Office database, the following accounts are created as part of the application:

- The application owner (usually `OZG_OWNER`)
- The user for Virtual Private Database views: `OHI_VIEW_OWNER`
- The user for execution of custom dynamic PL/SQL: `OHI_DPS_USER`
- The user for execution of batch requests: usually `BATCH`

See hereafter, Creation of database accounts and privileges.

5.1.2. Creation of database accounts and privileges

In order to create the required database accounts and roles and to assign the correct authorizations, the OHI Back Office installation script `OZGI001S.sql` has to be run via `SQL*Plus` under account `SYS`.

This script creates an Oracle database schema which contains OHI Back Office objects. The standard suggested name for this schema/account/user is: `OZG_OWNER`. This is referred to as the application owner.

Additionally, a secure database role is created for object authorization: `OZG_ROL`.

For security reasons, another account is created that will execute dynamic PL/SQL code with limited privileges at runtime. The fixed name for this account is: `OHI_DPS_USER`. The password of this account should be specified during the initial installation, but the application does not need to know it as a proxy connect will be used when a logon to this account is needed.

To support the Virtual Private Database (VPD) implementation an additional account is created that owns duplicate view definitions of the non-VPD related views owned by `OZG_OWNER`. The (also fixed) name for this account is: `OHI_VIEW_OWNER`. The password of this account should be specified during the initial installation, but the application does not need to know it as a proxy connect will be used when a logon to this account is needed.

A separate account is created for the OHI Back Office batch scheduler. The standard name for this account is: `BATCH`. If necessary, it is possible to use a different account name; the name of the account to be used has to be registered in the OHI Back Office System parameters.

Finally, all required authorizations will be granted to these accounts and roles.

5.1.3. Creating wallets

As explained in Overview of Operating System accounts, the OS accounts `oracle` and `batch` both need to be able to log in to the OHI BO database automatically, from scripts, but with different database accounts. This must be configured by using a Secure External Password Store (SEPS). This SEPS feature uses an Oracle Wallet. A SEPS can store one or more username/password combinations in an encrypted file.

A wallet will be used to pass the username/password combinations for the batch account to the database during batch processing and for the application owner database account during installation of releases.

The settings explained below cause all `sqlplus /@<db_connect_string>` statements to use the information in the wallet at the specified location to authenticate to the database. The `<db_connect_string>` identifies a username/password combination in the wallet, if present. A `<db_connect_string>` which is known in the wallet identifies exactly one username/password combination.

This same `<db_connect_string>` must be available as an alias to a database service in your `tnsnames.ora` file, so the details for the database connection can be found.

The wallet itself consists of two files, `ewallet.p12` and `cwallet.sso`. The last one contains the actual username/password combinations and must be protected with correct operating system access rights: any OS user who can read the wallet can use it to connect to the accounts stored in the wallet! Only the OS user `batch` and the OS user `oracle` need this file access.

Two additional files are created when the wallet is opened for the first time: `ewallet.p12.lck` and `cwallet.sso.lck`.

Before the wallet can be used to pass credential information to the database for Oracle Net connections, the Oracle Net client must know where to look for the wallet. This is specified in the `sqlnet.ora` file using the `WALLET_LOCATION` parameter and should specify the directory location of the wallets created in the next chapter. That `sqlnet.ora` file is found through the `$TNS_ADMIN` environment variable.

Access to the wallet(s) is arranged through OS file permissions.

Any OS account that can access (read) the wallet can use all username/password combinations in that wallet.

For security reasons, the OS account `batch` must not be able to log on to the database using the application owner database account. This means the OS account `batch` must not be able to access the same wallet as the OS account `oracle`.

A complication is that the wallet location can only be specified in the `sqlnet.ora` file, which in turn is found through the `$TNS_ADMIN` environment variable. That same `$TNS_ADMIN` environment variable also determines the location of the `tnsnames.ora` file that contains the details of the database services. The contents of this file changes frequently and is often distributed centrally.

Because of the security requirement, the OS account `batch` needs a separate value for `$TNS_ADMIN` and a separate version of `sqlnet.ora` and the wallet files, but you may not want to duplicate the `tnsnames.ora` file.

The easiest way to supply a separate value for the `$TNS_ADMIN` environment variable for the OS account `batch` is to set the default value in generic profile scripts and overrule that value in

the profile for `batch`, and not (re)set it in any scripts. Be aware that the template for `ozg_init.env` contains a line to set `$TNS_ADMIN`. If you need to keep that line active, you will have to make it conditional, based on the current OS user (e.g. if `$USER = "BATCH"`.....)

To avoid duplication of the contents in `$TNS_ADMIN/tnsnames.ora`, you can use the `ifile=...` directive to include the same (original) file from two “wrapper” `tnsnames.ora` files in separate directories. These wrapper files can be identical and will not change.

All these considerations result in a possible setup that is summarized in the picture below. This assumes your current `$TNS_ADMIN` directory is `<tnsbase>/tnsadmin`.

OS account `oracle`:

is part of the groups `sysdba` and `ohibo`

- has an environment variable `$TNS_ADMIN` that points to a directory `<tnsbase>/tnsadmin` that is not accessible for OS account `batch`
- can read the file `tnsnames.ora` in directory `<tnsbase>/common`
- uses `<tnsbase>/tnsadmin/sqlnet.ora` that contains:

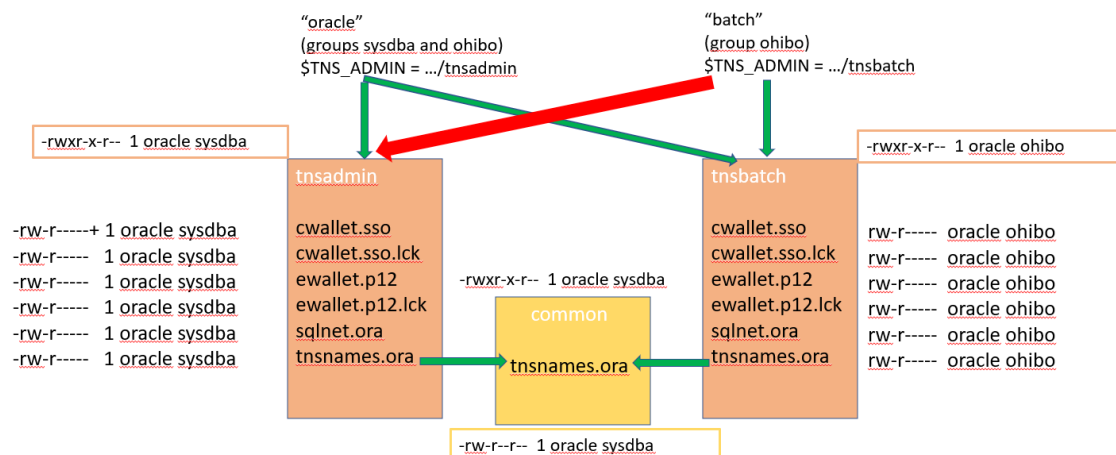

```
WALLET_LOCATION =
  (SOURCE= (METHOD=FILE) (METHOD_DATA= (DIRECTORY=<tnsbase>//tnsadmin)))
```

OS account `batch`:

- is part of the group `ohibo` but not of group `sysdba`
- can read all files in directory `<tnsbase>/tnsbatch` because of the group membership
- can read the file `tnsnames.ora` in directory `<tnsbase>/common`
- cannot read any files in directory `<tnsbase>/tnsadmin`
- has an environment variable `$TNS_ADMIN` that points to a directory `<tnsbase>/tnsbatch`, that is owned by `oracle` (so `oracle` can administer this directory).
- uses `<tnsbase>/tnsbatch/sqlnet.ora` that contains:

```
WALLET_LOCATION =
  (SOURCE= (METHOD=FILE) (METHOD_DATA= (DIRECTORY=<tnsbase>//tnsbatch)))
```

Files and privileges for separate wallets



NOTE: the `*.lck` files in the picture are created the first time the wallet is used.

Both `<tnsbase>/tnsadmin/tnsnames.ora` and `<tnsbase>/tnsbatch/tnsnames.ora` contain only:
`ifile=<tnsbase>/common/tnsnames.ora`

5.1.3.1. Wallet for OS account ORACLE

Following the setup discussed above, create the wallet for OS account `oracle` as follows.

With OS account `oracle`

- create the directory `<tnsbase>/common` with the correct privileges and ownership
- move your existing `<tnsbase>/tnsadmin/tnsnames.ora` file to `<tnsbase>/common` and check the permissions and ownership.

- create file `<tnsbase>/tnsadmin/tnsnames.ora` and put one line in it:

```
ifile=<tnsbase>/common/tnsnames.ora
```

- edit file `<tnsbase>/tnsadmin/sqlnet.ora` and add the lines:

```
WALLET_LOCATION =  
(SOURCE=(METHOD=FILE) (METHOD_DATA=(DIRECTORY=<tnsbase>/tnsadmin)))  
SQLNET.WALLET_OVERRIDE = TRUE  
SSL_CLIENT_AUTHENTICATION = FALSE
```

NOTE: `SSL_CLIENT_AUTHENTICATION` is set to false only to avoid issues when not using SSL/TSL encryption or using certain types of SSL/TSL encryption for the SQL*Net traffic. When using TSL encryption for connections to Autonomous Data Warehouse (ADW), do **not** include this setting.

- create the wallet in `<tnsbase>/tnsadmin` using the syntax `'mkstore -wrl [wallet_location] -create'`. The example below creates it in the current directory (identified with '.'):

```
mkstore -wrl . -create
```

The two files that implement the store (`ewallet.p12` and `cwallet.sso`) are now created. You will be asked to define a password to protect the contents of the wallet. This password will be asked whenever you access the wallet with management commands.

- Store the credential for `oracle` to use when running OHIPATCH using the syntax:

```
mkstore -wrl [wallet_location] -createCredential [db_connect_string]  
[username] [password]
```

A credential consists of a combination of a database alias (a.k.a. `db_connect_string`), username and password. The alias must be a known 'service name alias', a TNS entry, in `tnsnames.ora`. Add it to the file `tnsnames.ora` if necessary.

```
cd <tnsbase>/tnsadmin
```

```
mkstore -wrl . -createCredential OHIDEV01_INSTALL OZG_OWNER  
MYOZGOWNERPASSWORD
```

The command above will ask for the wallet password and create a credential intended for installing releases on the OHI BO Environment OHIDEV01. Note that the actual database that is used only depends on the value of the alias `OHIDEV01_INSTALL` in `tnsnames.ora`.

- Check the privileges of the files in `<tnsbase>/tnsadmin`.
- Test access using SEPS:

```
sqlplus /@OHIDEV01_INSTALL
```

This should connect to the database without asking for a password. Check the user and the database name:

```
SQL> show user
SQL> select * from global_name;
SQL> exit;
```

5.1.3.2. Wallet for OS account BATCH

Following the setup discussed above, create the wallet for OS account `batch` as follows.

With OS account `oracle`

- create file `<tnsbase>tnsbatch/tnsnames.ora` and put one line in it:
`ifile=<tnsbase>/common/tnsnames.ora`
- copy file `<tnsbase>tnsadmin/sqlnet.ora` to `<tnsbase>tnsbatch/sqlnet.ora` and modify the line:
`WALLET_LOCATION =`
`(SOURCE=(METHOD=FILE) (METHOD_DATA=(DIRECTORY=<tnsbase>/tnsbatch)))`
- create the wallet in `<tnsbase>/tnsbatch` using the syntax `'mkstore -wrl [wallet_location] -create'`. The example below creates it in the current directory (identified with '.'):


```
mkstore -wrl . -create
```

The two files that implement the store (`ewallet.p12` and `cwallet.sso`) are now created. You will be asked to define a password to protect the contents of the wallet. This password will be asked whenever you access the wallet with management commands.

- Store the credential for database user `batch` to use when running the batch scheduler using the syntax:

```
mkstore -wrl [wallet_location] -createCredential [db_connect_string]
[username] [password]
```

A credential consists of a combination of a database alias (a.k.a. `db_connect_string`), username and password. The alias must be a known 'service name alias', a TNS entry, in `tnsnames.ora`. Add it to the file `tnsnames.ora` if necessary.

```
cd <tnsbase>/tnsadmin
```

```
mkstore -wrl . -createCredential OHIDEV01_BATCH BATCH MYBATCHPASSWORD
```

This will ask for the wallet password and create a credential intended for running the batch scheduler for the OHI BO Environment OHIDEV01. Note that the actual database that is used only depends on the value of the alias `OHIDEV01_BATCH` in `tnsnames.ora`.

- Check the privileges of the files in `<tnsbase>/tnsbatch`.
- Test access using SEPS and OS account `oracle`:

```
sqlplus /@OHIDEV01_BATCH
```

This should connect to the database without asking for a password.

- Now switch to OS user `batch`, check the value of `$TNS_ADMIN` and test:

```
sqlplus /@OHIDEV01_BATCH
```

This should connect to the database without asking for a password. Check the user and the database name:

```
SQL> show user
SQL> select * from global_name;
SQL> exit;
```

```
sqlplus /@OHIDEV01_INSTALL
```

This should NOT connect to the database.

5.1.3.3. Remarks

- The simplest way to use this mechanism is to create an entry for the existing environment database alias, for example:

```
mkstore -wrl . -createCredential OHIDEV01 BATCH MYBATCHPASSWORD
```

However, this means that every OS user who uses for example the syntax `sqlplus /@OHIDEV01` (and has operating system read access to the wallet files) will connect as user `batch` to the OHIDEV01 environment. We advise you to always include the account name in the alias when using SEPS.

- If you use a connect string that is different from the default connect string, the start and stop commands for the batch scheduler require this special connect string as additional parameter (see later).
- For a RAC environment we advise to have at least a specific service identifying each of the node(s) on which the batch scheduler (and the batches, possibly using separate services) should run. The same advice is valid for running OHIPATCH.
- If you are using OHI Data Marts, you need to create a credential for the alias that is used as database identification for the Data Marts database in the Data Marts related batches (e.g. for the ETL processes, ZRG0E01S, ZRG0S01S and ZRG0D01S). This credential must be created in the wallet in `<tnsbase>\tnsbatch`.

IMPORTANT: If you do use a different connect string to identify the batch username/password combination for the Data Marts batch user (e.g. DMDEV01_BATCH instead of DMDEV01) be aware that the 'application users' that start the ETL processes should use this new connect string to define the connection to the Data Marts database. Inform them about this as they need to pass this connect string as parameter.

- Use the following syntax to show a list of currently defined credentials:

```
mkstore -wrl [wallet_location] -listCredential
```

- Use the following syntax to change an existing credential:

```
mkstore -wrl [wallet_location] -modifyCredential [db_connect_string]
[username] [password]
```

An example for when the password of the batch account above has changed to `ohibo2010`, issue the following command:

```
mkstore -wrl . -modifyCredential prod_batch batch ohibo2010
```

- Use this syntax to remove a credential:

```
mkstore -wrl [wallet_location] -deleteCredential [db_connect_string]
```

So credential `prod_batch` can be removed by issuing:

```
mkstore -wrl . -deleteCredential prod_batch
```



Attention: When connecting to the database as OS user oracle using the syntax `'sqlplus / as sysdba'`, read access to the wallet file `cwallet.sso` is needed if the wallet is activated in the active `sqlnet.ora` file, even when `TWO_TASK` is unset. So make sure both the OS user that owns the oracle software and the batch user have read access to their respective wallet files, but other users do not.

5.1.4. Creating a logon database trigger for the batch account

It is possible to further prevent unauthorized use of the batch scheduler account(s) by creating a logon database trigger for the batch account. Since database version 11.1.0.7 this trigger is no longer *mandatory*.

5.1.4.1. Details

The Oracle batch account has database role OZG_ROL_BATCH to have access to the OHI Back Office database objects stored within the schema of the OHI BO table owner.

The database batch account is authorized by means of the wallet file as this specifies the password for the database account. If someone gets hold of the wallet files he/she can log in with '/@<alias>' instead of by entering a username/password combination, from any workstation that is connected to the network, and impersonate the batch account. So, it is clear access to the wallet files must be limited.

When you know someone has 'stolen' the wallet used by the batch account you should change the password of the batch account, but you can also limit unauthorized access pro-actively in a different way as described now.

This security risk can be mitigated by checking, in a logon database trigger, the IP address of the client session for the Oracle `batch` user. If the IP address is not included in the list of permitted IP addresses, the session will be refused, and the database connection will be aborted. In this way the administrator has an additional means to prevent improper use of the batch account.

5.1.4.2. Example

As an example, this is a logon trigger for a fictitious situation:

Batch user	batch (in the database, OS user may differ)
Permitted servers	Local connection on the database server 144.21.160.66 (database server) 144.21.160.68 (application server via Oracle*Net)

The matching trigger code is included here (and is also available in the `$OZG_BASE/conf/Back-Office` folder in file `ozg_on_logon_batch.trg`):

```
create or replace trigger batch.ozg_on_logon_batch
after logon on batch.SCHEMA
begin
  /* Perform optional IP checks as mentioned in the OHI Back Office
  Installation Guide */
  declare
    l_addr varchar2(100) := sys_context('userenv','ip_address');
  begin
    if nvl(l_addr,'local') not in ('local'
                                   , '144.21.160.66'
                                   , '144.21.160.68')
    then
      raise_application_error('-20001','Connection refused');
    end if;
  end;
end ozg_on_logon_batch;
/
```

5.1.4.3. Implementation of limitative Logon Trigger

The implementation of the logon trigger that limits unauthorized access is as follows:

- Determine the name of the database batch account.
- Determine the servers with their IP addresses that have access to the batch account.
- Modify the trigger code from the sample for your situation.
- Create the trigger using a user with DBA privileges (e.g., SYS or SYSTEM).
If the trigger is not created under the `batch` user, then it will *not* be activated when logging on!

5.1.5. Creating a logon database trigger for the OHI owner account

Of course the same mitigation applies to reduce the risk for when the wallet used to logon to the OHI object owner is stolen. Only connections should be allowed by the server(s) which are used to run the OHIPATCH processes.

5.2. INSTALLATION OF THE APPLICATION

There are several ways to install the application, of which the most important ones include:

5.2.1. An Initial Installation

This is the case if the database contains an empty schema (created as described in the previous paragraph with script `OZGI001S.sql`) in which all objects will be installed from scratch.

In order to create the OHI Back Office objects for the owner account `OZG_OWNER` and to locate the application sources, an OHI Back Office initial release must be installed.

A description can be found in the following document:



OHI BO Release Installation Guide

5.2.2. A Copy of an Existing Installation

This is the situation when an existing environment is copied to the new environment and consists of two main parts:

- Copy the database objects of the application to another database.
- Copy the application server objects (by means of copying the `$OZG_BASE` directory, and afterwards modifying several settings; see the following paragraphs).

5.3. CONFIGURE AUTHORIZED APPLICATION SERVERS

After the database objects have been installed you need to register in the database which application server is allowed to start the OHI Back Office user interface against this database. The IP address of the application server (or servers) has (have) to be saved in the database for security reasons.

The 'granted' ip addresses will be stored in table `ALG#IP_ADRESSEN`.

There are pl/sql packaged procedures to assist you in adding or deleting addresses:

- "Owner".ALG_IAS_PCK.INS("IP address") for adding a new IP address.
- "Owner".ALG_IAS_PCK.DEL("IP address") for deleting an existing IP address.

Use these routines while connected to the database as the OHI BackOffice schema owner or as a privileged administrator account. Finish the transaction with a commit to make a change permanent.

5.4. COMPILE AND CHECK APPLICATION SOFTWARE

5.4.1. Compilation

Now compile all relevant application objects (database and application server side). To do this, execute the following activities in the OHI Back Office installation menu:

- Activity 120 for the compilation of database objects (e.g., packages and procedures in the database), for creating synonyms and for updating grant privileges.
- Activity 800 for the compilation of client objects (e.g., screens and menus)

A description can be found in the following document:



OHI BO Release Installation Guide

As part of step 800, 2 Pro*C modules (SYS1108S.pc and SYSS004S.pc) are compiled, using the gcc compiler. This may fail, with error messages like :

```
..
START: Compile Pro*C SYS1108S.pc
ERROR: Execute Pro*C preprocessor for SYS1108S failed with errorcode 1
..
START: Compile Pro*C SYSS004S.pc
ERROR: Execute Pro*C preprocessor for SYSS004S failed with errorcode 1
..
```

The log file for step 800 may show more detailed error messages.

search path issue

A common cause is the absence of a directory in the search path of the pre-compiler.

To fix this, set your environment variables for the database client and edit the configuration file of the pre-compiler:

```
. ozg_init.env $OZG_ORATAB_DB19
vi $ORACLE_HOME/precomp/admin/pcscfg.cfg
```

Add the correct directory to the variable `sys_include=` (within the brackets), after checking that the directory exists. It may have a different version in its name on your server.

For Linux 7, add

```
, /usr/lib/gcc/x86_64-redhat-linux/4.8.2/include
```

For Linux 8, add

```
, /usr/lib/gcc/x86_64-redhat-linux/8/include
```

glib-headers issue

Another issue may occur on Linux 8.7 or higher.

Symptom:

The log file of step 800 shows errors like:

```
START: Compile Pro*C SYS1108S.pc
Pro*C/C++: Release 19.0.0.0.0 - Production on Thu Jan 12 16:56:58 2023
Version 19.17.0.0.0
Copyright (c) 1982, 2019, Oracle and/or its affiliates. All rights
reserved.
System default option values taken from:
/ohi/oraBase/product/db19c/precomp/admin/pcscfg.cfg

Syntax error at line 166, column 45, file /usr/include/sys/cdefs.h:
Error at line 166, column 45 in file /usr/include/sys/cdefs.h
#define __glibc_fortify(f, __l, __s, __osz, ...) \
.....1
PCC-S-02014, Encountered the symbol "... " when expecting one of the
following:
    an identifier, define, elif, else, endif, error, if, ifdef,
    ifndef, include, include_next, line, pragma, undef, exec,
    sql, begin, end, var, type, oracle,
    an immediate preprocessor command, a C token, create,
    function, package, procedure, trigger, or, replace,
Syntax error at line 168, column 9, file /usr/include/sys/cdefs.h:
Error at line 168, column 9 in file /usr/include/sys/cdefs.h
    ? __ ## f ## _alias ( __VA_ARGS__ )
\
.....1
PCC-S-02014, Encountered the symbol "##" when expecting one of the
following:
```

Cause:

The version of glibc-headers is upgraded to 2.28-211.0.1.el8 in Oracle Linux 8.7. This version contains the unrecognized syntax. See Document 1365193.1 on My Oracle Support for a similar issue. According to this document, the cause is "The [AIX] header files in this version contain c99 syntax and the Oracle precompiler does not support c99 syntax and there is currently no plan to support it"

Workaround:

Move back to a lower version of the glibc-headers.

To use version 2.28-189.5.0.1.el8_6 instead, execute the following on the application server, while logged in as root:

- download the lower version of glibc-headers rpm

```
yum install --downloadonly --downloadaddir=/opt/glibc_ol86
glibc-headers-2.28-189.5.0.1.el8_6.x86_64
```
- unpack the rpm into the new directory

```
cd /opt/glibc_ol86
rpm2cpio glibc-headers-2.28-189.5.0.1.el8_6.x86_64.rpm |
cpio -idm
```
- edit the configuration file of the pre-compiler:

```
. ozg_init.env $OZG_ORATAB_DB19
vi $ORACLE_HOME/precomp/admin/pcscfg.cfg
```

Include the new directory /opt/glibc_ol86/usr/include in the variable sys_include= (within the brackets), **before** the existing /usr/include

5.4.2. Check

Now check the findings of OHI Back Office Object Check (via installation menu activity 900) or by starting a script request in the application (System/Management/General/Object check; in Dutch: Systeem/Beheer/Algemeen/Objectcontrole) and processing the findings. The latter requires a running batch scheduler and a correct directory configuration (see later).

5.5. CONFIGURING DIRECTORIES

Use screen "System/Management/General/System parameter" to configure directories for reading and writing OHI Back Office application files (such as output of script requests, online help information and release documentation).

For the virtual directories be sure to specify the correct port of the Oracle HTTP Server (OHS) or make the entries relative to the HTTP Server by starting them with /OHI (the virtual root folder configured in OHS).



Attention: If multiple Middle Tier servers are used, the directories have to be created on a shared file system so all nodes/instances have access.

For a detailed description see:



Reading, Writing and Authorising OHI Back Office application files

5.6. REGISTERING BATCH SCHEDULER ACCOUNT

The name of the batch scheduler account (normally `batch`) must be registered in the system parameters (screen "System/Management/General/System parameter", tab page "Batchscheduler", item "Batch account").

6. COMPLETING THE INSTALLATION

This chapter describes the completion of installation of OHI Back Office.

6.1. SYNONYMS FOR DATABASE LINKS

If synonyms have to be created for objects regarding database links (e.g. for OHI Data Marts and/or GL objects), then the database link has to meet the following requirements:

- The domain has to be included in the database link (e.g. `prod.world` instead of `prod`);
- The database link has to have the same name as the global name of the database.

6.2. STARTING OHI BACK OFFICE BATCH SCHEDULER

The OHI Back Office batch scheduler can now be started; see [Starting](#).

6.3. CHECKING THE INSTALLATION

The technical installation now has to be tested in order to determine whether or not it has been successful.

The minimal requirements to be tested include:

- OHI Back Office User Interface
 - Access to batch request log and output files
 - Access to online help
 - Access to release documentation
 - Menu authorization
- OHI Back Office batch scheduler
 - Functionality
 - Output
 - Messages in batch scheduler log file
 - Messages in log files of script requests
 - Availability of log and out directories for all users
 - Availability of database accounts for users of log and out directories
- Starting/stopping the complete OHI Back Office environment (including Database Server, Application Server, and OHI Back Office batch scheduler)
- Checking the log files (see also [Checks](#))

6.4. CREATING BACKUPS

After testing the installation, a full cold backup has to be created for the relevant file systems of the application server. A full backup of the database is also required, using your preferred method.

This may be a good time to test your backup and restore procedure for the environment.

6.5. ACTIVATING JOBS

The regular jobs for systems administration now have to be activated.
See the following chapters for possible jobs in the area of Oracle and OHI Back Office.

7. ORACLE ADMINISTRATION RELATED TO OHI BACK OFFICE

The OHI Back Office application runs on an Oracle database and uses Application Server runtime software.

Administration of this environment is the responsibility of the DBAs of the customer.



Attention: As mentioned earlier, this document does not intend to provide instructions in terms of the Oracle Database & Application Server administration.

The DBAs of the customer should have sufficient knowledge, skills and experience in order to perform this administration task in a correct way.

If the administration is not performed correctly, Oracle can provide no guarantees as to the correct functioning of the OHI Back Office application.

In order to emphasize the importance, the most significant points of interest involving Oracle Database & Application Server administration *related to* OHI Back Office are mentioned here *explicitly*.

The following enumeration is therefore a *minimal set of mandatory tasks and activities, which are required for the correct implementation of* OHI Back Office, and they should *not* be considered as a "manual" for the DBA's. Evidently, each activity mentioned must be tested and documented extensively.

Chapter 8 provides experience and advice as to how the administration could and should be performed. The administration activities described provide a good start.

7.1. BACKUP & RECOVERY

7.1.1. General

File system backups for the application server and output files are not described but are of course important.

There are various options to create a database backup.



Oracle recommends the use of Recovery MANager as a database backup tool.



If ASM is used as a shared storage solution, the use of Oracle RMAN is mandatory.

Since the 10g release of the DBMS, Oracle provides the possibility to use a Flash Recovery Area (FRA) for storing backup related data.



Oracle recommends using FRA, particularly for a faster recovery of the Database.

Ensure proper backup and restore procedures for the software used (e.g. Database, Application Server, OHI Back Office, etc.) and the relevant file systems.

Testing and checking of the backup and restore procedures is considered a standard procedure for system management and must be performed on a regular basis.

When it comes to using hot or cold backups, no specific requirements apply in combination with OHI Back Office.

7.2. STARTUP AND SHUTDOWN

7.2.1. Order

- After starting the database listener and the database, the OHI Back Office batch scheduler has to be started;
- Before the database is shut down, the OHI Back Office batch scheduler has to be stopped;

7.2.2. Automation of Startup/Shutdown

- It is highly recommended to restart/shutdown the Oracle system software automatically during startup/shutdown of the relevant servers;

7.2.3. Checks

- Check of the various Oracle system software files, e.g., trace files, alert files, audit files, AS log files `error_log*`, `access_log*`, `default-web-acces.log`, `em-web-acces.log`, `ipm.log`, `application.log` (Forms!)) etc.
- Check the OHI Back Office batch scheduler log files for possible errors;
- Check cause(s) for failed script requests (status M (failed)).

7.2.4. RAC

In an Oracle RAC environment, some additional requirements apply.

In an Oracle RAC environment two or more instances may be active.

An important issue is that starting and stopping an Oracle RAC environment is performed in a slightly different manner than a single instance environment.



Use the `srvctl` command to start and stop RAC components *instead of* `SQL*Plus`.

When starting instances with `SQL*Plus`, services are not always started automatically as well. If afterwards these services are started manually, then CRS is not aware of these services.

7.3. PERFORMANCE TUNING AND MONITORING

- OHI Back Office uses the Oracle *cost-based optimizer*; to this end it is mandatory that regular jobs are run which e.g., perform the following tasks: Collecting *statistics* for *tables* and *indexes*, collecting *system statistics*, collecting statistics for the SYS schema. For the OHI Back Office tables, *monitoring* is selected by default, which allows for *stale* statistics to be used.
- Regular tracing of activities in order to recognize performance bottlenecks.
- Deployment of the right disk settings for Oracle; e.g., RAID levels, striping, controllers etc.;
- Performing regular checks of OS, Oracle and application performance by using the relevant diagnostics tools;
- Monitoring network performance, memory, disk I/O, disk space etc.
- Configuring and implementing Oracle Resource Management;
- Monitoring SPM repository growth, execution of evolve tasks, purging of baselines;

- etc.



Advice: Oracle recommends using Enterprise Manager Cloud Control for the active day to day monitoring.

7.3.1. RAC



Oracle recommends using Enterprise Manager Cloud Control for managing and monitoring an Oracle RAC environment.

To this end an Oracle Enterprise Manager *Management Pack* license is required as an extra option.

In an Oracle RAC environment one or more instances can be active.

For Database Management this entails more emphasis on performance monitoring.

In an Oracle RAC environment, the most important performance issues to be recognized include the following:

- Performance of the Interconnect
As a guideline this is maximum 5ms.
- Performance of the Global Cache

If errors are encountered in terms of performance when using OHI Back Office in an Oracle RAC environment, the delivery of performance data will continue in the same way as described in section [Collecting Performance Data](#).

Additional information regarding RAC Performance Monitoring can be found in the following documentation:



[Oracle Clusterware and Oracle Real Application Clusters Administration and Deployment Guide, chapter 14 'Monitoring Performance'](#).

7.3.2. Rebuilding Indexes

In our experience, few indexes ever need rebuilding.

Conditions for rebuild would be:

- Large free space (generally 50%+), which indexes rarely reach, *and*
- Large selectivity, which most index accesses never reach, *and*
- Response times are adversely affected, which they rarely are.

Therefore, the OHI general advice on rebuilding is:

- Do not rebuild indexes.
- Use `REBUILD` if the whole index structure is very poorly fragmented.
- Use `COALESCE` to reduce index fragmentation in a portion of the index.
- Use `SHRINK` to reduce index fragmentation in a portion of the index and the actual blocks associated to the index.

7.4. USING SQL PLAN MANAGEMENT

The OHI Back Office application supports the use of SQL Plan Management (SPM) functionality to reduce the possibility that SQL statements ‘suddenly’ execute a lot slower than before because of a changed execution plan. This paragraph describes the current support.

7.4.1. Introduction

A short introduction is given to SQL Plan Management in combination with the Oracle Health Insurance Back Office application.

7.4.1.1. SQL Plan Management

These paragraphs describe how OHI BO supports the SQL Plan Management (SPM) feature of the Oracle RDBMS. A customer can enable SPM for OHI BO to achieve SQL execution plan stability, thereby generally preventing that execution plans for SQL statements change until it has been proven the change is an improvement. This plan stability is needed as these changes could sometimes be for the worse and might be caused by for instance:

- the installation of a new database (and thus optimizer) version,
- the installation of a major patch set for the database (for instance an SPU),
- changes to optimizer statistics for a table based on its changed contents,
- other types of changes:
 - system statistics and system settings (e.g., db_cache, pga_aggregate_target),
 - optimizer related changes in the spfile,
 - schema and metadata definitions (new/modified indexes and/or constraints),
 - SQL profile creation,
- the adaptive cursor sharing feature,
- the cardinality feedback feature.

All of the above can introduce a change in the execution plan of a SQL statement during a hard parse. Typically, a changed plan will perform at least as good as the previously used execution plan for the SQL statement. However, every now and then the change encompasses a regression. And depending on the OHI BO process in which this regression occurs, the change can cause a significant performance issue and from that lead to a service disruption. To prevent this, the more explicit use of SPM is introduced in the OHI BO application.

Reason for this is that enabling SPM for the whole instance may be unwanted due to the additional overhead and attention this may involve, taking up more time of the administrators to actively manage this. This because also all database internal foreground and background SQL as well as SQL of other application logic running in the same database may be captured and use baselines. In order to prevent this OHI offers functionality to enable SPM usage specifically for OHI processes.

OHI customers may decide themselves whether they implement similar functionality for other SQL processes running in the same database or using SPM instance wide, on pluggable database level or for the whole container database.

7.4.1.2. Prerequisites

The following settings are advised for SPM:

- *Optimizer_use_sql_plan_baselines/optimizer_gather_sql_plan_baselines=false*
Setting these parameters to FALSE at the instance level, may seem strange when you

like to enable SPM for OHI. However, OHI supports the use of SPM on a database session-by-session basis for OHI specific. OHI BO will set both initialization parameters to TRUE in those sessions that are configured to use SPM. This is done via 'alter session' statements embedded in OHI BO application software.

It is allowed though to set the GATHER and/or use USE parameter to TRUE at instance level (PDB or CDB\$ROOT) but beware of the impact of all kinds of other SQL statements that may need to be captured and evolved. You may choose to only set the USE parameter to TRUE and gather baselines for statements that you deem relevant. The activation of SPM by OHI is an application service only and the database administrators remain fully responsible for monitoring and managing the use of SPM.

- *We advise following defaults for the two SPM configuration parameters:*

SPACE_BUDGET_PERCENT → 30%

PLAN_RETENTION_WEEKS → 60 weeks

SPM stores its data in the SYSAUX tablespace. The percentage parameter value denotes the threshold of storage used in SYSAUX by SPM above which the instance will write warnings to the alert.log file of the database. Depending on the size and usage of the SYSAUX tablespace on your site, you can increase the percentage threshold. The second parameter specifies the number of weeks during which execution plans will be retained by SPM. This parameter should be set to at least a full year.

You can set these parameters within the OHI pluggable database through commands like:

```
exec dbms_spm.configure('SPACE_BUDGET_PERCENT',30);
exec dbms_spm.configure('PLAN_RETENTION_WEEKS',60);
```

You can query the configured values through dictionary view
DBA_SQL_MANAGEMENT_CONFIG.

You can monitor the space usage approximately by using the query below for occupant SQL_MANAGEMENT_BASE (SMB, SQL Management Base):

```
select occupant_name
,      space_usage_kbytes
,      to_char
      (100*space_usage_kbytes/
       (sum(space_usage_kbytes) over ()))
      ,'990D9')
      || '%' percent
from   v$sysaux_occupants
order by
      2 desc
```

Beware that the percentage can become more than the specified percentage as this is only used to trigger the writing of warnings.

- *Following object and system privileges are required by OHI BO's owner schema:*
 - execute on sys.dbms_shared_pool
 - administer sql management object

7.4.2. Enabling SPM for OHI BO

7.4.2.1. User interface and WS support for SPM

OHI BO supports SPM for Forms modules and web services by means of a Back Office parameter specifying to enable or disable this (application wide).

7.4.2.2. Enabling SPM for a specific Batch definition

It is possible to enable the use of SPM baselines for plan stability at the level of individual batch definitions. This is done in the batch screen module (SYS1008F), which can be found from the menu via the following path (when running with English as application language): *System/Management/Module/Batch/Batch*.

Figure 2.1 shows a screenshot of this module. Near the middle it has the ‘Baselines?’ checkbox. By unchecking this, you ensure that later runs of this batch will not be using the SPM feature. When this is checked all future runs will issue the following two statements at the beginning of the run:

```
alter session set optimizer_use_sql_plan_baselines = true;
alter session set optimizer_capture_sql_plan_baselines = true;
```

The screenshot shows the 'Batch' management window. The 'Baselines?' checkbox is located in the middle section, under the 'Max. Subprocesses' and 'Max. No. of Hours' fields. It is currently unchecked and is circled in red. The window also displays various configuration options for the batch, including 'Batch Type', 'Output Type', 'Optional Sets Minimum', 'Print Automatically', 'Optimizer Mode', 'Service', and checkboxes for 'Errors', 'Warnings', 'Informational', 'New Standard', 'Overtime Trace', 'Lapsed', 'Stopped', 'Parallel Supported', and 'Parallel'.

No.	Set	Description	M	LOV	LOV Restriction	Validation
1						
2						
3						

No.	Parameter	Description	M	E	Prompt	Name in Batch	Operator	Column
1								
2								

Default Value: Reference Date: Correction Days:

Hint:

Figure 2.1: The Batch Management page (SYS1008) with baselines checkbox.

7.4.2.3. Enabling SPM for the whole database

Instead of using the Back Office parameter and script definitions for activating SPM it is possible and allowed to specify the use of SPM instance-wide by setting the two initialization parameters for this:

- optimizer_capture_sql_plan_baselines
- optimizer_use_sql_plan_baselines

When these are set to true all processes within the database will be using SPM.

7.4.3. SPM Maintenance and Administration

Introduction of the SPM feature for OHI BO requires little attention of the application DBA. However, there are a few topics that the DBA should be aware of. These are addressed in this paragraph.

7.4.3.1. Monitoring SYSAUX Tablespace Usage

SPM uses a repository (SQL management base, SMB) of execution plans that are to be used for executing the application SQL statements. This repository is stored in the SYSAUX tablespace. The Oracle DBMS monitors the space usage of this repository. It will signal when the storage of the SPM repository has reached the threshold value as specified by the SPACE_BUDGET_PERCENT parameter discussed in the 'Prerequisites' paragraph. This is done by writing a message to the instance's alert.log file.

```
Tue Oct 22 04:20:04 2013
SPM: SMB space usage (...) exceeds ...% of SYSAUX size (...).
```

The DBA should monitor the appearance of this message in the alert.log file. A way to do this is by introducing a monitoring metric using OEM Cloud Control, see the [Appendix](#) for setting this up. Whenever this message appears you should ensure that SYSAUX still has enough room to cater for the storage growth of the SPM repository: either extend the tablespace or increase the threshold value when enough free space is still available in the tablespace.

7.4.3.2. Monitoring OHI Evolve Job results

When batch jobs or other sessions run in 'SPM mode', not only will they use the execution plans that are stored in the SPM repository, but they can also introduce new execution plans into the SPM repository. SPM will prevent the use of these new execution plans until they have been proven to be at least equal to, or better than, the existing execution plans in the SPM repository. Validating that it is okay to use these new plans is called 'evolving' these plans.

The OHI BO application automatically evolves new plans through a daily OHI Evolve job. This is implemented through a background job as described for the [batch scheduler](#). A Back Office parameter is present to specify the number of days between actual executed evolves (usually set at 1) so the actual load can be planned to occur only once in a specified number of days. This can be used to postpone evolves during for example a period where all capacity is needed or you do not want other execution plans.

The results of this (normally daily) evolve job are recorded in the alg#sql_baselines_log table. It is advised that the DBA monitors the contents of this table on a regular (e.g., weekly) basis to validate that the evolve job is doing its work correctly. You can do this by checking the output of this query:

```
select *
from   alg#sql_baselines_log
where  action = 'EXCEPTION'
```

There should be no exceptions reported. If there are, please log a service request with OHI support.

7.4.3.3. Disabling the 'standard' Evolve Job

The database contains a standard advisor job as one of the jobs within the Automatic SQL Tuning Task. Starting with database release 12.2 this job will be enabled and will execute evolve actions for enabled and not yet accepted plans. These evolve actions do interfere with the OHI background maintenance job and disturb a correct working. Because the OHI job currently contains more relevant functionality it is very strongly advised to disable the standard job. Unfortunately, this task cannot be disabled separately so currently the only way to inactivate it is to give it no time to do its work.

To disable the task, execute:

```
DBMS_SPM.set_evolve_task_parameter
( task_name => 'SYS_AUTO_SPM_EVOLVE_TASK'
, parameter => 'TIME_LIMIT'
, value => 0
);
```

You can check in the days after this action the execution actually fails as intended by querying the executions:

```
select *
from   dba_advisor_executions
where  task_name = 'SYS_AUTO_SPM_EVOLVE_TASK'
order by
        execution_start desc;
```

7.4.3.4. Undoing results of OHI Evolve Job

You can undo the results of the most recent evolve job. In an exceptional situation, support may request you to do this. Evolve jobs are identified by a run_id. First you would need to determine the run_id value of the most recent evolve job. The following query will do this.

```
select max(run_id)
from   alg$sql_baselines_log
```

Using that run_id's value, execute the following packaged procedure, followed by a commit:

```
alg_qbl_pck.undo_evolve([run-id value]);
commit;
```

7.4.3.5. Packing SPM repository

OHI Support may request you to deliver the contents of the SPM repository. OHI Support may require the contents of your repository in order to resolve a performance related service request. Upon such request, execute the following packaged procedure.

```
alg_qbl_pck.export_baselines
```

This procedure will ‘pack’ your SPM repository into a table `alg#sql_baselines_stage`. Using datapump export you should then export this table into a dump file. This dump file can be sent to OHI support.

7.5. INSTALLATION, CONFIGURATION, UPGRADE, MIGRATION AND VERSION CONTROL

- Installation of new releases and patches for Oracle Database and Application Server, possibly by means of Rolling Upgrades.
- Installation of new releases and patches for OHI;
- Planning upgrades and migrations;
- etc.

7.5.1. RAC

- If setup correctly, CRS will automatically start and stop the components required for the cluster (incl. database and listeners); check this configuration of dependencies by means of `crs_stat`;
- Use `crsctl check cluster` to check all nodes in CRS.
- Use `ocrcheck` to check the integrity of OCR (and mirror);
- Use before and after the different stages of the installation process the Cluster Verification Utility `cluvfy`.
- See [installation and configuration](#) for further information regarding the installation.

7.6. ACCESS CONTROL AND SECURITY PRIVILEGES

- Access control of the relevant OS accounts (e.g., `root`, `oracle` and `batch`);
- Access control of the default Oracle database accounts (e.g., `SYS` and `SYSTEM`).
- Access control of the OHI Back Office Oracle database accounts (e.g., `OZG_OWNER` and `MANAGER`);
- Changing the passwords a.s.a.p. (after the initial installation);
- Configuring a password algorithm and validity duration, in other words setting up a policy in terms of password management;
- Access control when OHI Back Office is implemented in a web-based architecture on intranet or the Internet;
- File access. A good practice for better protection would be to use `umask 037` and assign all files in the `$OZG_ROOT` folder structure to a special OHI group which is also the primary group of the batch account. This requires manual action.
- Monitoring security alerts (<http://otn.oracle.com/deploy/security/alerts.htm>).
- etc.

7.7. SPACE AND STORAGE MANAGEMENT

OHI Back Office supports table compression for the tables that contain the fact data related to the insured members. The “fact tables” are the tables that grow, as opposed to the ‘configuration’ tables. In this way the storage impact of the larger tables can be reduced. For further information, please read the Oracle Health Insurance Release Installation Guide.

Additional activities related to space and storage management are:

- Managing the available and required disk space;
- Managing storage settings for objects;
- Shrinking objects (e.g., based on Segment Shrink Advisor);
- Managing the size of various Oracle system software files, e.g., trace files, alert files, audit files, AS log files `error_log*`, `access_log*`, `default-web-acces.log`, `em-web-acces.log`, `OPMN log files` etc.
- Managing the size of OHI Back Office Batch scheduler log and output files in `$OZG_LOG` and `$OZG_OUT`.
- etc.

7.8. LICENSE CONTROL

- Checking license issues for the installation of new releases and/or patches or the use of certain options and/or tools;
- etc.

7.9. NETWORKING

- Making use of Oracle Net *dead connection detection* by means of configuring parameter `SQLNET.EXPIRE_TIME` in file `sqlnet.ora` with a value larger than zero. For more information, please see the Net Services Reference manual as being part of the standard database documentation.
- etc.

7.9.1. RAC

- For RAC environments it is crucial that the different nodes have the exact same system time; it is highly recommended to use NTP in this case (Network Time Protocol).
- If in an Oracle RAC environment an instance is put in restricted mode (= required for the installation of OHI (patch) releases), then the services for that instance will be stopped (by the Oracle Clusterware, only in case of *dynamic registration* with the listener).

Connection to the instance will still be possible via:

1. `SID`; this is to be used for connecting to the container database for executing DBA tasks; if tasks need to be executed for the pluggable database the session needs to switch to this PDB container.
2. `Service Name`; if this is used, which is necessary for `OHIPATCH.pl` to connect, then the `tnsnames.ora` entry must be modified (see also My Oracle Support note 301099.1).

Between the `CONNECT_DATA` and `SERVICE_NAME` entries the following has to be

added:
(UR=A)



Remark: Working with service names is preferred.

8. OHI BACK OFFICE MANAGEMENT

The management activities for OHI Back Office can be subdivided into tasks for System management/DBA and tasks for creating new application users.

8.1. SYSTEM MANAGEMENT/DBA

8.1.1. Oracle Environment

Typically, an OHI Environment runs on two types of servers:

- A Database Server running
 - a Pluggable Database (PDB) for each OHI BO environment
 - (optionally) a Pluggable Database (PDB) for each OHI DataMarts environment
 - a separate PDB for the OPSS schemas of the Fusing MiddleWare Infrastructure
- An Application Server running
 - An Oracle Forms Managed Server (from an ORACLE_HOME that has Oracle Forms on top of WebLogic FusionMiddleWare)
 - An Oracle HTTP Server (OHS), typically running from the same Home as the Forms Managed Server)
 - A FMW Managed Server for the OHI Web Services, possibly running from a different FWM Home. This Managed Server typically services only one OHI Back Office environment.
 - An OHI Batch Scheduler for each OHI Back Office environment

On the Application Server

- the OS account `oracle` typically is the owner of the Oracle FMW software and the OHI Back Office files.
- the OS account `batch` typically is the owner of the OHI Back Office batch scheduler.

8.1.1.1. Starting Application Server components

To start the WebLogic Node Manager, Admin Server, Managed Server and the Oracle HTTP Server of the Forms Domain on the Application Server, use your custom scripts.

The same goes for the FMW components in the Web Services Domain(s).

These will typically be run as `oracle` and set the right set of environment variables using `ozg_init.env` and then use the standard FMW scripts to start the components in order:

Use the OHI-supplied script and the account `batch` to start the OHI Batch Scheduler for each OHI BO environment.

Component	Script
FORMS Domain	<FORMSDOMAIN_HOME>/bin/setDomainEnv.sh
Forms Node Manager	nohup \$DOMAIN_HOME/bin/startNodeManager.sh &
Forms Admin Server	nohup \$DOMAIN_HOME/bin/startWebLogic.sh &

Forms Managed Server	nohup \$DOMAIN_HOME/bin/startManagedWebLogic.sh <WLS_FORMS> &
Oracle HTTP Server	\$DOMAIN_HOME/bin/startComponent.sh <ohs1>
OHI BO batch scheduler	\$OZG_BASE/sh/OZG_START_BATCH.sh - verbose -connect <ohienv>_batch <ohienv>
WEB SERVICES Domain	<WSDOMAIN_HOME>/bin/setDomainEnv.sh
Web Services Node Manager	nohup \$DOMAIN_HOME/bin/startNodeManager.sh &
Web Services Admin Server	nohup \$DOMAIN_HOME/bin/startWebLogic.sh &
Web Services Managed Server	nohup \$DOMAIN_HOME/bin/startManagedWebLogic.sh <WLS_WEB> &



Remark: The Admin Server of the Forms Domain is not needed at runtime.

Remark: The Admin Server of the Web Services Domain is not needed at runtime, only when deploying new versions of the OHI Web Services or changing the configuration.

8.1.1.2. Stopping Application Server components

To stop the Oracle HTTP Server, Managed Server, Admin Server and the WebLogic Node Manager of the Forms Domain on the Application Server, use your custom scripts.

These will typically be run as `oracle` and set the right set of environment variables using `ozg_init.env` and then use the standard FMW scripts to stop the components in order.

Use the OHI-supplied script and the account `batch` to stop the OHI Batch Scheduler for each OHI BO environment.

Component	Script
OHI BO batch scheduler	\$OZG_BASE/sh/OZG_STOP_BATCH.sh -verbose -connect <ohienv>_batch <ohienv>
FORMS Domain	<FORMSDOMAIN_HOME>/bin/setDomainEnv.sh
Oracle HTTP Server	\$DOMAIN_HOME/bin/stopComponent.sh <ohs1>
Forms Managed Server	nohup \$DOMAIN_HOME/bin/stopManagedWebLogic.sh <WLS_FORMS>
Forms Admin Server	nohup \$DOMAIN_HOME/bin/stopWebLogic.sh
Forms Node Manager	nohup \$DOMAIN_HOME/bin/stopNodeManager.sh
WEB SERVICES	<WSDOMAIN_HOME>/bin/setDomainEnv.sh

Domain	
Web Services Admin Server	nohup \$DOMAIN_HOME/bin/stopWebLogic.sh
Web Services Managed Server	nohup \$DOMAIN_HOME/bin/stopManagedWebLogic.sh <WLS_WEB>
Web Services Node Manager	nohup \$DOMAIN_HOME/bin/stopNodeManager.sh

8.1.1.3. Starting and stopping the database(s)

To stop the databases for one or more OHI Back Office environments, execute the standard actions for database management.

8.1.2. Clients

The user interface of the application OHI Back Office can be started on a client PC or 'Remote Desktop' via a browser by means of an URL like:

<http://<host>:<port>/forms/frmservlet?config=Prod>

It is recommended to configure a DNS server that enables all relevant clients in the network to access the relevant host. If a DNS server is used, then it is not required to manage the local host files on each client PC.

Additionally, it is recommended to position all relevant URLs for OHI Back Office on a company portal or homepage, in order to simplify the implementation and management.

The value of the parameter "config" refers to the configuration as it can be found in file \$DOMAIN_HOME/config/fmwconfig/servers/WLS_FORMS/applications/formsapp_12.2.1/config/formsweb.cfg.

OHI strongly recommends allowing only https communication, using a URL like:

<https://<host>:<port>/forms/frmservlet?config=Prod>

8.1.3. OHI Back Office Database accounts

The following provides an overview of the 4 required OHI Back Office database accounts with their usual usernames (you may use other usernames for some of these accounts, but initial installation scripts use the names below and this documentation will refer to these account names):

- OZG_OWNER/<specified during installation>
Owner account for the OHI Back Office data structure.
- OHI_VIEW_OWNER (mandatory name)
Account that holds duplicates of all view definitions owned by the OZG_OWNER account to support the Virtual Private Database (VPD) implementation, except for the views that are related to this VPD implementation itself.
- OHI_DPS_USER/<specified during installation> (mandatory name)
Account that executes the dynamic pl/sql code, using correct and allowed object privileges, that can be configured within the application.

- BATCH/<specified during installation>
Owner account for the OHI Back Office batch scheduler.

Several other, optional, accounts may be needed, depending on your use of OHI Back Office functionality:

USERNAME	PRESCRIBE D NAME	FUNCTIONALITY
C##DBV_ACCTMGR_ROOT, C##DBV_ACCTMGR_ROOT_BACKU P, C##DBV_OWNER_ROOT, C##DBV_OWNER_ROOT, C##DBV_OWNER_ROOT_BACKUP	No	Database Vault
OHI_REALM_AUTH_USER	No	OHI Realm for Database Vault
HSL_USER	No	HSL Web Services. Used in Data Sources to access HSL code.
OBD_SELECT_USER	Yes	OHI Data Marts access to OHI Back Office data. Used in Database Link in DM.
OHI_JMS_QUEUE_USER	Yes	Access to OHI JMS queues for Service Consumers and Message Based Output. ("Berichtgebaseerde uitvoer")
PSL_USER	No	PSL Web Services (for OHI JET). Used in Data Sources to access PSL code.
SVL_USER	No	SVL Web Services. Used in Data Sources to access SVL code.
SVS_OWNER	No	Owner of custom developments

OHI also requires an OHI user account ("Functionaris") named "MANAGER" that is used during installation of patches. This should NOT be a database account. Because OHI user accounts can only be created after a database account has been created, make sure you drop or lock the MANAGER database account after creation of the OHI user account.

It is best practice to set a specific user context (in the configuration files svl.properties, hsl.properties and psl.properties) for the OHI Web Services, that will be used in the audit columns of the records created or modified by the Web Services. This user context must also be an OHI user account WITHOUT a database account. A typical name would be WEB_USER.

8.1.4. OHI Back Office Batch Scheduler

The OHI Back Office batch scheduler handles script requests which are submitted by the end users of the OHI Back Office application.

8.1.4.1. Starting

Service Name Versus Environment Name

Under OS account `batch` enter the following command for the environment in question; the Environment Name of the environment for which the batch scheduler has to be started is provided as a parameter.

```
. ozg_init.env environment_name  
$OZG_BASE/sh/OZG_START_BATCH.sh environment_name
```

The script will start for 1 environment, 1 batch scheduler on 1 server.

It is possible to use a (database) service name that differs from the (OHI) environment name. This is typically useful when a Secure External Password Store is used with a name which is different from the environment name, and which identifies the batch username/password combination. This service name can be passed as 'connect' parameter:

```
. ozg_init.env environment_name
$OZG_BASE/sh/OZG_START_BATCH.sh -connect service_name environment_name
```

Verbose Mode

By means of parameter `-verbose` (optional) it is possible to start the batch scheduler in *verbose* mode. In this mode, extra information is written in the log file, which can help when you are investigating issues:

```
. ozg_init.env environment_name
$OZG_BASE/sh/OZG_START_BATCH.sh -verbose -connect service_name environment_name
```

Retention Period for Log Files

When starting the batch scheduler, the last (optional) parameter is the retention time (in days) of the log files of the batch scheduler.

By default, this is 7 days.

```
. ozg_init.env environment_name
$OZG_BASE/sh/OZG_START_BATCH.sh environment_name 14
```

8.1.4.2. Stopping

Under OS account `batch` enter the following command for the relevant environment:

```
. ozg_init.env environment_name
$OZG_BASE/sh/OZG_STOP_BATCH.sh environment_name
```

Or when a separate service name is used to connect to the user batch:

```
. ozg_init.env environment_name
$OZG_BASE/sh/OZG_STOP_BATCH.sh -connect service_name environment_name
```

The script will stop all batch schedulers on all servers for the given environment.

In the overall scripts `ozg_batch_start.sh/ozg_batch_stop.sh`, calls are included to start/stop *all batch schedulers for all environments on 1 server*.



The OHI Back Office batch scheduler runs on the Middleware Tier.



See [Appendix A - configuration multiple batch schedulers](#) for the implementation of *multiple* batch schedulers in case multiple application servers are allocated.



For a RAC environment we advise to have a specific service identifying the node(s) on which the batch scheduler (and the batches) should run.

8.1.4.3. OHI Back Office batch scheduler Output and Log Files

OHI Back Office recommends to regularly clear the OHI Back Office batch scheduler output files in the directory `$OZG_OUT` and the OHI Back Office batch scheduler log files in the directory `$OZG_LOG`.

8.1.5. Changing Settings

8.1.5.1. Switching Software Trees

In a standard configuration, after the installation of OHI Back Office, OHI will use the Oracle software in the Application Server software tree.

The OHI Batch Scheduler needs additional software from another software tree. It will use either the database software tree (if the Database Tier and Middleware Tier are installed on the same server) or the database client tree (the Database Tier and Middleware Tier are installed on different servers).

In order to switch your OS session to the database or client software tree, the following script can be used:

```
. ozg_init.env $OZG_ORATAB_DB19
```

In order to reset the environment for the 12c R2 Application Server software tree, the following calls can be used:

Example:

```
. ozg_init.env $OZG_ORATAB_FRS122119
```

Note that both these variables are set in `ozg_init.env`.

8.1.5.2. Switching OHI Environments

In order to switch your OS session to a specific OHI environment (using the Application Server software tree), the following commands can be used, e.g.:

```
. ozg_init.env prod # configuring the OHI prod environment settings
. ozg_init.env test # configuring the OHI test environment settings
```

In order to switch your OS session to a specific OHI using the Database or Client software tree, (e.g. for startup/shutdown of the instance), the following commands can be used, e.g.:

```
. ozg_init.env $OZG_ORATAB_DB19 # configuring the 19c database software
. ozg_init.env prod              # configuring the prod service_name.

. ozg_init.env $OZG_ORATAB_DB19 # configuring the 19c database software
. ozg_init.env test              # configuring the test service_name
```

In the situation above, the first call will recognize that the parameter (`$OZG_ORATAB_DB19`) identifies a database Oracle Home and will clear the `$TWO_TASK` variable (whatever it is) while retaining the `$ORACLE_SID`). The second call will set both `$ORACLE_SID` and `$TWO_TASK` to the passed environment name. After that a call like `sqlplus ozg_owner/<password>` will use the environment name in `$TWO_TASK` to connect to the specified service name set by the second call to `ozg_init.env`.

If you need to connect to the root container of the container database, for example to execute instance management actions like starting the instance, the commands above should be used in the opposite order and specify the name of the instance of the container database:


```
. ozg_init.env CDB01                # configuring the SID name
. ozg_init.env $OZG_ORATAB_DB19 # configuring the 19c database software
```

In the above situation the first call will set both \$ORACLE_SID and \$TWO_TASK to the passed instance name. The second call will recognize the parameter identifies a database Oracle Home and will clear the \$TWO_TASK variable (while retaining the \$ORACLE_SID).

A call like `sqlplus / as sysdba` will connect to the root container. You can then issue the command `alter session set container=<pdb_name>` to switch to the specified pluggable database.

8.1.6. Adding New OHI Back Office Application Environments

When a new OHI Back Office application environment within an existing software environment is required, the following steps have to be performed in the Middleware Tier:

- Add a batch scheduler start command for the new environment in `$OZG_ADMIN/ozg_batch_start.sh`.
- Add a batch scheduler stop command for the new environment in `$OZG_ADMIN/ozg_batch_stop.sh`.
- Configure a new environment `ENV_NAME` in `$DOMAIN_HOME/config/fmwconfig/servers/WLS_FORMS/applications/formsapp_12.2.1/config/formsweb.cfg` by means of the Enterprise Manager control application for the Forms and Reports services.
- Add an `ozg_servlet_wls_<ENV_NAME>.env` file with appropriate settings in `$OZG_ADMIN` (if you referenced this location in the variable `envFile`).
- Create a new application directory `$OZG_ROOT/environment_name`.
- Create log and out directories for the users, including the batch user. See the next paragraphs for details.
- Install the OHI Back Office application software for the relevant environment.

8.2. CREATING NEW APPLICATION USERS

The following steps have to be performed when configuring a new end user for the OHI Back Office application:

8.2.1. Creating an Oracle Account

First of all, an Oracle account has to be created in the pluggable database for the new end user. Subsequently, the required system privileges have to be assigned to the new account. This step can be performed in SQL*Plus as user SYS or SYSTEM while connected to the pluggable database. Issue the `CREATE USER` command:

Example

```
create      user          scott
identified by          tiger
default      tablespace  users
temporary   tablespace  temp
quota       unlimited on users
/
```

```
grant create session to scott
/  
grant alter session to scott
/  

```



If OHI Back Office is configured with Single Sign-On (SSO) the user must be created in the SSO Server repository too.

8.2.2. Creating Directories for the New User

For each new end user, directories have to be created in the Middleware Tier. The OHI batch scheduler will create output files and log files there. These files need to be downloadable from the Forms application via the users' browser.

This has to be performed under OS account `batch` :

```
mkdir $OZG_OUT/<new account>
mkdir $OZG_LOG/<new account>
```

<new account> is the name of the Oracle account of the new end user, in *lower case*.

Example

```
mkdir $OZG_OUT/scott
mkdir $OZG_LOG/scott
```



If multiple Middle Tier servers are used, the directories have to be created on a shared file system so all the nodes/ have access.

8.2.3. Storing Application Authorization

Now the new end user can be authorized for the application OHI Back Office. In order to create the authorizations, start the application and log on as `MANAGER`.

- Create a “Functionaris” (OHI user account) for the new end user in screen “Systeem/Beheer/Autorisatie/Gebruiker” (system/management/authorization/user).

Now assign the required roles by means of menu authorization in the same screen.

The standard menu role `MANAGER_ROL` has only limited menu authorization to be able to set up the authorization model; other roles should be created and set up as desired.

Close the application, restart it and log on as the new end user in order to test the newly created account.

If desired, the user settings can be modified by means of screen “Bewerken/Instellingen” (Edit/Settings).

It is optional to configure the required functional application authorization (company access, authorization for administrative organizations, brands etc.)

For the tables under the module and user authorization the mutation logging is enabled by default. For more information about mutation logging see:



ORACLE HEALTH INSURANCE BACK OFFICE – CUSTOM DEVELOPMENT

8.3. INSTALLATION OHI (PATCH) RELEASES

The installation process of OHI (patch) releases is described in the OHI Release Installation Manual and is performed on the Middleware Tier:



ORACLE HEALTH INSURANCE BACK OFFICE – RELEASE INSTALLATION
GUIDE

For new customers this and any other relevant details will be shown by Oracle in a knowledge transfer session.

8.4. CLONING AN OHI BACK OFFICE ENVIRONMENT

During the lifecycle of an OHI Back Office environment the wish may come up to copy an environment to a new environment or to overwrite an existing environment with another existing environment. We refer to this as ‘cloning’ an environment.

As technical setup and configuration may differ a lot between customers and technical infrastructures there is no standard way to describe this. Additionally, the impact on interfaces with other applications and systems is customer and environment specific.

For these reason, only some generic guidelines can be given regarding what should be done to clone an OHI Back Office environment.

8.4.1. Definition of an ‘Environment’

When an OHI Back Office environment should be cloned we expect the pluggable database, application server configuration and accompanying file structure should be ‘cloned’: a copy should be made so it can run under a different ‘name’ (alias).

8.4.2. Identical Technology Stack

For cloning to succeed the ‘OHI contents’ must be moved to a technical infrastructure that is identical, meaning it contains identical database and application server software, including the patches installed for this technology stack.

8.4.3. Database ‘copy’

The pluggable database should be cloned (physically or virtually using for example storage technology or snapshot cloning) and started using a different name (export and import might be an option but requires a special approach and is much more time consuming; cloning a pluggable database is fairly easy).

8.4.4. Clone ‘approach’

The following actions should be executed to clone an OHI Back Office environment:

1. Determine a source environment that is stable (stopped or frozen) and can be ‘copied’.
2. Choose an appropriate new name for the new environment (use meaningful naming conventions).
3. Make sure you have a container database to plug the pluggable database in to.

4. Choose the pluggable database name and create it by cloning the source PDB, using the new name (typically based on the new environment name).
5. Copy/clone the application server file system with the OHI software. This is typically the \$OZG_BASE folder structure. Make sure it is copied to an environment where \$OZG_ADMIN is based on the same or newer releases of OHI Back Office.
6. Copy/clone additional file system structures as used by your environment (typically locations identified by \$OZG_OUT and \$OZG_LOG and file system locations you have defined in database directories), for as far as you need historic input and output files.
7. Create at least a service to connect to the pluggable database and consider deleting the services that were used for the old name. Do not forget to change the startup trigger that starts these services, if needed (depends on the way the trigger is created).
8. Adapt the servlet .env files (it is assumed you store them in \$OZG_BASE): change the name when it contains the environment name; change the settings in the .env file where they contain environment specific settings such as TWO_TASK or a file system location.
9. Change the WebLogic Server configuration for Forms (formsweb.cfg) so that new environment entries are present for all environment specific variables.
10. Adapt/add tnsnames.ora entries and wallet entries so they are present for the new environment.
11. Update startup/shutdown scripts that contain environment specific entries.
12. Update ozg_init.env if you have environment specific settings in there.
13. [Update/add IP addresses](#) if the IP address of the application server that will be used is unknown.
14. When you are able to startup the pluggable database, make sure you update the environment specific entries in tables ALG_SYSTEEM_PARAMETERS and ALG_BO_PARAMETERWAARDEN (this can be done through the System Parameter and Back Office parameter screen when you have configured the application server for the new environment; when you update the database you may script this for efficiency reasons). Typically the settings for the batch scheduler and file locations may have to change.
15. Update the database directories you may have configured for input or output, so they reflect the different output location if that is environment specific.
16. Deploy the Service Layer web services (SVL), Use Cases web services (HSL) and the Jet application (including PSL web services) if they are needed for this environment. Make sure the Data sources connect to the new PDB.
17. Change database links to the old PDB and from the new PDB if relevant.
18. Implement any additional changes for custom code (using dynamic code within OHI or using custom code schema owners or other interfaces).
19. Lock or remove end user database accounts you cloned from Production. Also lock the “Functionaris”-records. Change passwords for database account that must remain available. Optionally add accounts for testers.
20. Thoroughly check whether you are complete. Especially when you made a clone from a production environment, in which case you should be sure there is no interaction with any production functionality in your application landscape.

Make sure you understand the process above and identify any possible omissions. Test whether all functionality is working correctly. After this publish the name, URL's, WSDL's, etc. of this environment to your 'user community'.

8.5. ALLOWED SCHEMA MAINTENANCE

For a reliable and supported operation of the application it is not allowed to alter any of the standard OHI object definitions within one of the OHI database schemas. This means that only OHI provided scripts and/or processes may be used to alter the definition or the contents of any of the objects within such a standard OHI account.

This means that explicitly the following actions are not allowed:

- Changing the definition of table (column definition, constraints, indexes, compression or partition structure). Such changes should always be implemented by OHI provided scripts.
 - It is allowed to increase the PCTFREE or INI_TRANS setting or move a table temporarily to a different tablespace for reorganization purposes provided all impact on dependent objects (unusable indexes, invalidated pl/sql) is solved before the environment is being used again.
- Adding non OHI objects to an OHI object owner other than OHI dynamically maintained objects.
- Changing the contents or structure of other objects (view definitions, trigger definitions, stored pl/sql code, sequence definitions).
- Also, removal of OHI objects is not allowed except when explicitly specified.

Of course, maintenance to implement responsible ownership is allowed. What does this mean?

- Meaning that for reorganization purposes not only tables may be relocated but also indexes may be rebuilt and storage clauses also for indexes may be adapted.
- Also, incorrect NLS length semantics may be corrected and a schema may be exported and imported provided it is checked that all owner objects are moved including the relevant roles and object privileges as well as the system privileges granted to OHI schemas. Robust checks to compare the situation 'before' and 'after' should be implemented.
- Retention periods for OHI provided queues may be increased (within reasonable limits, to prevent the queue table contents grows too much), not decreased.

When it is not clear whether a maintenance action may impact the support status of OHI please contact your OHI representative before acting and do not proceed.

9. REFERENCES FOR ACTIVE MANAGEMENT OF OHI BACK OFFICE

In this chapter a possible approach is proposed to 'actively manage' the OHI Back Office. This has to be used at the time that OHI Back Office is successfully configured and operational.

Active management entails the undertaking of preventive actions to prevent performance problems, in particular as a consequence of passive management. It also strives to simplify the search for the cause of performance problems when they do occur. Additionally, a number of matters are tackled to apply active space management.

In case performance issues are encountered, a suggestion is made for handling the issue.

The target group for this chapter is the OHI Back Office database manager, the DBA for an OHI Back Office environment. For Unix managers, the chapter could be useful to get the hang of the database management section of this chapter in particular. For functional application managers with a highly technical interest, certain sections could function as an illustration with respect to the cause of performance problems (and probably also possible solutions).

This chapter does not focus on the background of the techniques used. Enough documentation regarding this matter can be found elsewhere. Main goal is to provide enough information to handle the fundamental issues, even if not much expertise is available regarding the subject. Experts in certain subareas would want and can collect better or more detailed information at times.

The setup has been selected so that acquired practical experiences are shared. As expected, the chapter is an evolving section of this document which will be further elaborated in the future (based on numerous acquired experiences, both internally at Oracle and externally at OHI Back Office customers). In that sense it will never be finished and always be open for improvement (and therefore also for comments and possible suggestions).

The setup is based on an OHI Back Office database. The content cannot simply be applied to an OHI Data Marts environment given that additional or different matters may be applicable.

ATTENTION: No rights can be derived from this chapter. It is just a matter of advice (unless specifically indicated differently). This is to be used at your own risk.

9.1. PERFORMANCE - GENERAL

In this chapter some general sizing guidelines are given regarding the OHI Back Office application.

The advises itself are quite outdated and mainly provide a historical insight. Over the past decade an updated sizing document has been delivered with major releases that contains high level indexation numbers for growth in CPU and Memory resources.

9.1.1. Memory and CPU

The following graphs provide minimum guidelines for the required CPU and memory capacity for OHI Back Office.

The guidelines are based on internal and particularly external key figures of used equipment and test results. No computed values are involved. The values, such as the ones of the equipment of various customers in the first months of 2009 can be found in a graph. Based on additional customer-specific information and an estimate of the trend to be recognized, subsequently, a certain trend line can be defined. When using the customer key figures, it is important to understand that each customer runs customer specific additional workload on the server in

question, in addition to the OHI Back Office application. The graph figures should therefore not be followed blindly, but they could be used as an extra tool for sizing a server configuration.

The figures are based on combined database and application server allocation on a single server. However, for a separated database and application server, both servers will have to meet almost all of these criteria as the daytime work will primarily focus on the application server and the nighttime work primarily on the database server.

9.1.1.1. Memory

In terms of memory, for each OHI Back Office user *at least* 20MB has to be available, but in reality users sometimes clearly require more when they use multiple sessions and screens.

RAC

In case of RAC Oracle recommends setting up the following extra allocations per node:

- 10% extra buffer cache
- 15% extra shared pool

9.1.1.2. CPU

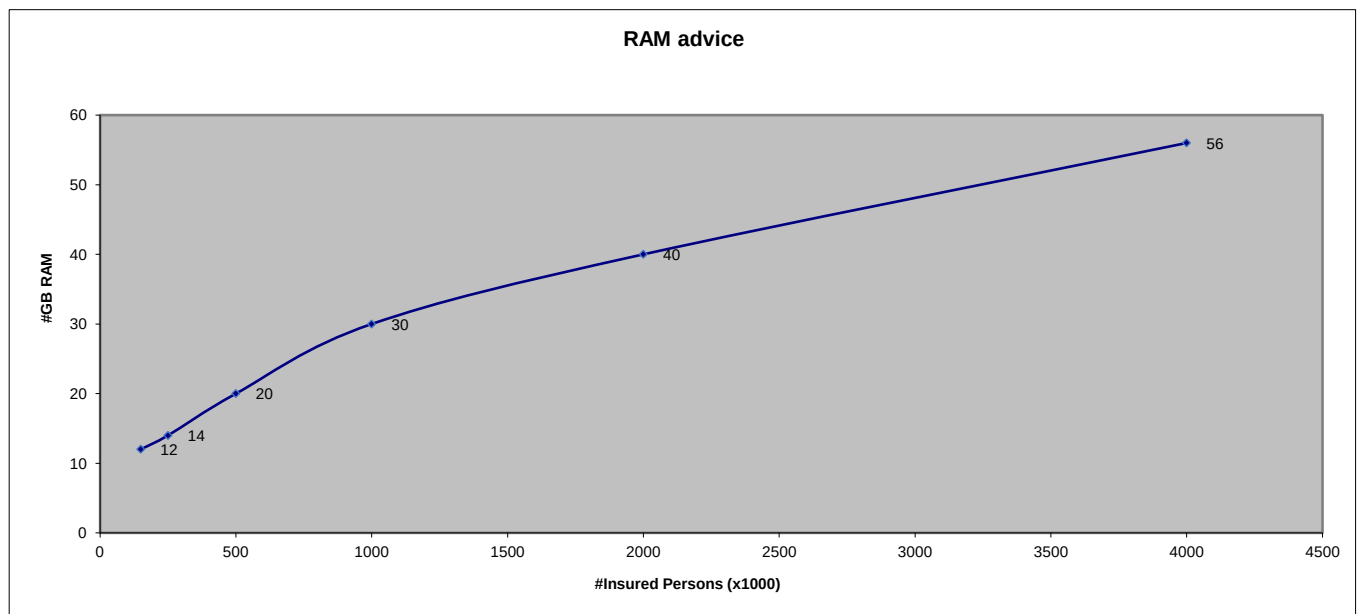
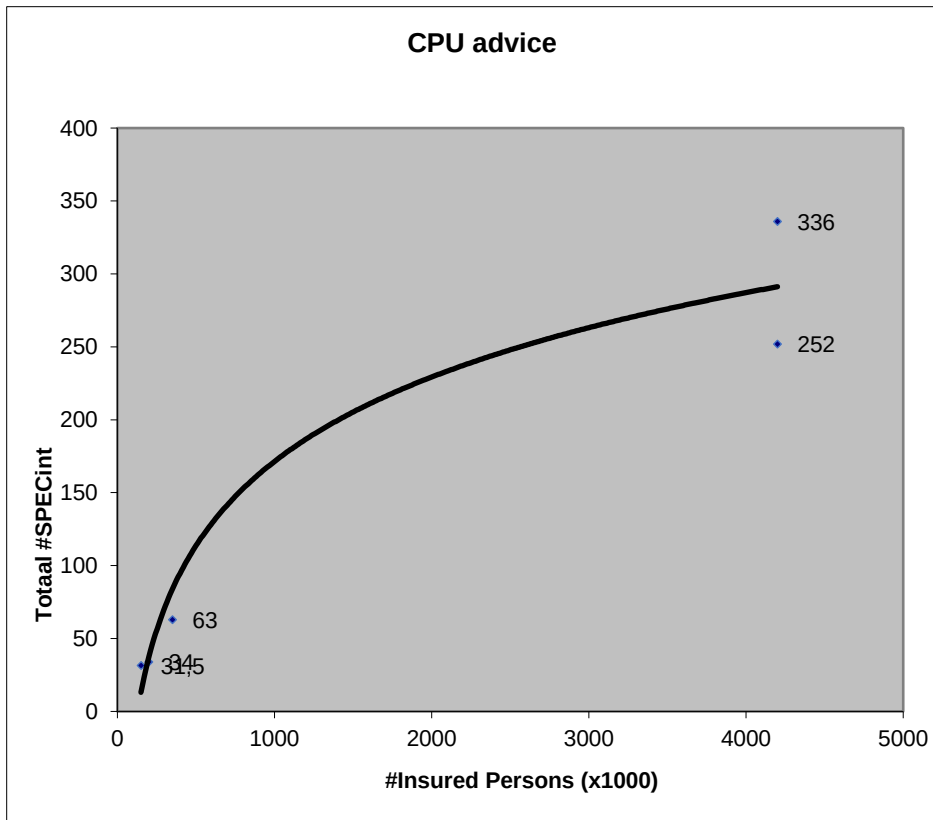
With respect to the number of CPUs, historically a *maximum* of 100 users has to be considered for each CPU. However, considering the multi core and multi-threading techniques nowadays available exceeding this number is possible.

The CPU use is expressed in SPECint2006 benchmark values, which indicates the CPU capacity based on certain integer actions. This is not really a benchmark considered as representative for an application such as OHI Back Office, but it provides something to hold on to when comparing the CPU capacity of certain servers. Given that this value is available for a rather large diversity of equipment, this benchmark is applied. Transaction-oriented benchmark values would be preferable, but this is not available for sufficient equipment in order to establish a thorough comparison of the equipment as is the case for the various customers.

On site <http://www.spec.org/cpu2006> information can be found regarding the benchmark applied.

By means of <http://www.spec.org/cgi-bin/osgresults?conf=cpu2006> it is possible to search for the benchmark values for certain types of servers. The figures used in the following CPU graph are the result of the value of field CINT2006 *rates*. For example Sun Fire E6900 with 16 dual core processors. The SPECint_rate2006 value for this server is 288.

If RAC is used, Oracle recommends counting for each OHI Back Office RAC platform (i.e. all nodes together] with at least 10% extra CPU (SPECint).



9.1.2. I/O

With respect to I/O throughput it is possible to establish that a random I/O on a data file for OHI applications may take on average at the most about 5ms. More than 5ms can be considered as a potential bottle neck and cause unwanted delay in the application response times.

In AWR output this is the column `Av Rd(ms)` in the `Tablespace IO Stats` overview

9.2. PERFORMANCE & MANAGEMENT OF THE OHI BATCH SCHEDULER

The OHI Back Office ‘batch scheduler’ is the component that starts and monitors ‘batch requests’. Its role is vital in processing the important batch and background processes of the application. Good tuning of the settings is required in order to maximize the use of available system capacity while preventing response times degradation during online usage. This needs quite some attention and asks for a good insight in available system capacity, requirements of the ‘business’ and the impact of changing certain database and application settings that influence this processing.

9.2.1. Parallelism used in batch requests

An important goal of the batch scheduler is to run the jobs that execute batch requests in parallel with each other. Over the years a new mechanism has been introduced for batch requests. Batch module definitions that are based on this ‘Renewed’ mechanism (this can be recognized by looking up this property in the batch module definition screen SYS1008F) do implement a standard way of workload distribution, over parallel processes, which consists of 2 phases:

First phase - Workload gathering

During the first phase the main process starts up and determines a smaller or larger amount of work units. Normally this is the ‘serial phase’ of processing a batch request.

Second phase - Workload execution

When all work units have been determined by the main process, it creates a number of subprocess batch requests to execute the workload in parallel. These subprocesses are started for the same ‘main’ batch request and pick up units of work as long as there are units left and a possible maximum time limit is not exceeded. The main process monitors this until all parallel subprocesses are stopped.

The first phase, the workload gathering, is typically the lighter part of the batch request execution and is normally processed by a single process (‘serial’) in quite a limited amount of time after which the real ‘worker subprocesses’ start doing the actual work in parallel.

Parallelizing the first phase

However, for some batch requests this first phase of workload gathering can be time consuming because quite complicated conditions might need to be checked to determine which units of work need to be executed. For these jobs, functionality has been introduced since release 10.13.2 of OHI Back Office to ‘parallelize’ the workload gathering. Therefore, for some batch request definitions parallel execution of the workload gathering ‘query’ is supported. Whether this is supported is defined by the ‘Parallel Supported’ property in the SYS1008F screen.

If this parallelism is supported, it can be activated providing that the database settings have been set in such a way that this can be supported. Make sure activating this for a batch definition is only done when the database parameters described below are configured to support parallelism and that these settings have proven, during representative tests, to speed up the processing.

To support this parallelism during the first phase, starting with release 10.13.2 a setting of PARALLEL_MAX_SERVERS larger than zero is supported. Since that release a database trigger is introduced which disables parallel execution by default, so parallel execution (in fact ‘parallel query’) needs to be activated explicitly.

Before activating the parallel execution, please make sure IO calibration inside the database has been performed. This is described in the ‘Oracle Database Performance Tuning Guide’ in the chapter about ‘I/O Configuration and Design’, the paragraph ‘I/O Calibration inside the Database’.

Also take notice of paragraph ‘Tuning General Parameters for Parallel Execution’ as present in the chapter ‘Using Parallel Execution’ within the ‘Oracle Database VLDB and Partitioning Guide’.

When you want to use this functionality, please follow these recommendations:

- Specify a value larger than zero for `PARALLEL_MAX_SERVERS` but make sure not more than 25 to 50 percent of the original maximum number of parallel processes (specified as system parameter, in table `ALG_SYSTEEM_PARAMETERS`, column `BATCH_MAX_PROCESSEN`) for the OHI batch scheduler is specified. And be sure to decrease the value of this system parameter that specifies this number of parallel processes.
This parameter does not have to be reduced with an equal percentage because it is expected that the parallel execution by the database will be used a lot less than the parallel sub processes functionality of the batch scheduler. But beware not to overload the system with the combination of parallel query execution processes and parallel batch request processes and start with conservative settings.
- If you are using RAC set `PARALLEL_FORCE_LOCAL` to `TRUE` (this will be checked during installations and the ‘OHI object validation’).
- Specify a value `MANUAL` or `AUTO` for `PARALLEL_DEGREE_POLICY` (`LIMITED` is not supported) and know what this means. Setting to `AUTO` activates the automatic degree of parallelism (auto DOP) functionality.
- Specify the value `TRUE` for `PARALLEL_ADAPTIVE_MULTI_USER`.
- Optionally change the `PARALLEL_MIN_TIME_THRESHOLD` parameter.

The above settings enable you to use the 19c automatic degree of parallelism functionality, which enables statement queuing and in-memory parallel execution, or to stick with pre 11gR2 parallel execution functionality. Our advice is to only use automatic degree of parallelism functionality if you have tested that this does not have negative impact on other processes. In case of doubt, do not enable it, as only a few batches may benefit from it.

By enabling adaptive parallelism, the system can reduce the degree of parallelism to prevent an overload of the system. Consequently, response times may vary.

To prevent parallel execution processes from flooding the system when auto DOP is used, the parameter `PARALLEL_DEGREE_LIMIT` may be set to a non-default value. By default, the limit will be CPU based, but as OHI environments have been seen to be sometimes read IO constrained, a setting to IO may be preferred in your environment.

9.2.2. Batch scheduler settings

The batch scheduler settings are maintained with screen module SYS1010F (Menu: System/Management/General/System parameters, in Dutch: Systeem/Beheer/Algemeen/Systeemparameter).

These are the most important parameters:

- Max parallel subprocesses
The maximum number of subprocesses allowed to start for executing a task. Many high-volume batches consist of a dispatcher which distributes its work to worker processes. The maximum number of subprocesses per request applies to the dispatcher and its worker processes.
Note that the dispatcher creates the sub processes by creating a batch request for each worker process.
- Max parallel batch requests
The maximum number of batch requests that are allowed to execute simultaneously. It limits the total number of running requests and includes the subprocesses (see above).

The corresponding system parameter is
ALG_SYSTEEM_PARAMETERS.BATCH_MAX_PROCESSEN

9.2.2.1. Sizing BATCH_MAX_PROCESSEN

The value of this parameter is very important since it contributes to the throughput of the batch processes. If this parameter is set too low, not enough work will be executed. If set too high, the database performance may suffer to the point that end users can no longer use the OHI Back Office application.

A rule of thumb is to calculate the initial value for BATCH_MAX_PROCESSEN as 2 x the number of CPU threads in the database nodes used for batch processing.

Examples:

- Single database node with 2 quad-core CPU's and 2 threads per core.
Number of threads: 16 (2x4x2).
Initial value for BATCH_MAX_PROCESSEN: 32 (16x2)
- RAC configuration with 2 database nodes for batch processing.
Each node has 2 hex-core CPU's and 2 threads per core.
Number of threads for 2 servers: 48 (2x2x6x2)
Initial value for BATCH_MAX_PROCESSEN: 48 (24x2)

The reason to limit BATCH_MAX_PROCESSEN to twice the number of CPU threads is to reduce the amount of time waiting for IO.

Note that these are initial values and expect to revise these values as you get more insight in the actual use and performance of your configuration.

Also keep in mind that database resource management may be implemented to throttle the database instance. In that case the calculated initial value for BATCH_MAX_PROCESSEN may be on the high side.

9.3. DATABASE PERFORMANCE

This section elaborates on the various aspects of performance of OHI Back Office. Important settings are assigned but utilities are also offered that may be of assistance when zooming in on possible performance issues.

The objective is to provide an overview as to how it is possible to monitor and possibly improve the performance of OHI Back Office.

By getting familiar with a normal performance and by collecting data about it, in case of actual performance problems it is easier to zoom in on the cause and nature of the problem. Especially if reference measures of a representative period with a normal/acceptable performance are available.

9.3.1. Database Settings

A smooth performance of OHI Back Office starts with optimally setting up and configuring a number of database settings.

9.3.1.1. Initialization Parameters

It is important to configure some crucial parameters in a correct manner. We list the parameters that are particularly important and for some cases we recommend possible values. Many other parameters can be configured, but at least the following are important for the performance of OHI Back Office.



With respect to the installation of OHI (patch) releases, different settings apply. See the following document for more details (a complete listing of database parameter requirements is present in an Appendix):



OHI BO Release Installation Guide

The Most Crucial Parameters

There are 2 parameters that are very significant as they determine the greater part of the SGA:

- `DB_CACHE_SIZE` (=Ym)
Use this instead of old parameter `DB_BLOCK_BUFFERS`. Specify the size of the buffer cache in multiples of 16MB, with a minimum suggestion of 128MB. As a guideline use a value of about 100MB per 50,000 insured persons in OHI Back Office. For environments with smaller numbers of insured persons, a clearly higher value has to be applied. For very large numbers - millions - this number can be adjusted.
- `SHARED_POOL_SIZE` (=Ym)
Therefore it is best to also specify multiples of 16MB. A nice guideline to start with is 5 MB per user in case of 100 users, however, with a minimum of 512 MB. For several hundreds of users it is possible to take a somewhat lower value.

Use the 'Advice' button in *Oracle Enterprise Manager (OEM)/Cloud Control* to get a picture of the correctness of the configuration after the database has been used for a couple of hours (!) with a typical load. A good time is e.g. by the end of the morning of a typical weekday.

Currently complete use of Automatic (Shared) Memory Management is not advised. If activated make sure sufficient minimum values are set for `DB_CACHE_SIZE`, `SHARED_POOL_SIZE` as well as `SGA_TARGET` (and also for `PGA_AGGREGATE_TARGET`, but more about that parameter follows).

Parameters for SQL Tuning

The parameters in `v$sql_optimizer_env` are of importance in order to activate the *Cost Based Optimizer* properly for OHI Back Office.

Parameters for Session Memory Allocation

There are various parameters to activate the allocation of memory for sessions, the allocation of the working memory. In the past it was possible to reserve a certain amount of memory per type of operation. However, it is possible to specify a target for all sessions together depending on the size of the total volume of working memory. The latter is normally preferred and is performed by means of the following PGA parameter.

- `PGA_AGGREGATE_TARGET` (=Ym)
It is recommended to use this setting to keep the total volume of working memory within certain limits. An automatic conversion takes place to the working memory allocation per session. The total target volume can become larger temporarily, but the memory is kept within certain limits. This has as the advantage that an objective can be set for all sessions together. A possible drawback is that specific optimization for a certain objective is not possible. This ordinarily is not required anyway. As a guideline, a reservation of 5 MB per user can be kept for e.g. about 200 users. A multiple of 16 is not required.
- `PGA_AGGREGATE_LIMIT` (=Ym or Zg)
We recommend using this setting to specify an explicit maximum limit. Without explicit limit, a default will be used. This default may either be too low for certain actions (like loading the AGB file) or it may be too high and hit operating system limits. For typical production workloads in OHI Back Office we advise a minimum limit 12 GB, mainly because of the process that loads the AGB file.

For several of these parameters it is also possible to obtain some advice for optimization of this configuration by clicking the 'Advice' button. Advice can only be used after a representative load of the database has taken place. It is also possible to simply check if the PGA 'cache hit percentage' lies above 90% and if the maximum PGA volume and current volume do not exceed the objective by too much. Otherwise the value has to be revised upward.

Instead of automatically assigning working memory, it is possible to assign specific memory for certain operations by means of the so-called `....._AREA_SIZE` parameters. These parameters are used only if the parameter `WORKAREA_SIZE_POLICY` is set to `MANUAL`. If the last parameter has not been set, then this will be set to `AUTO` by default when `PGA_AGGREGATE_TARGET` has a value that is greater than zero. It is recommended to use the following parameters only for certain purposes and to activate them by temporarily setting `WORKAREA_SIZE_POLICY` to `MANUAL`. For OHI Back Office it is no longer needed to set this parameter to `MANUAL`, you can leave it to the default value `AUTO`.

- `WORKAREA_SIZE_POLICY (=manual)`
Set to manual in order to have the following parameter values be used. Otherwise, set to `auto` or do not configure it at all.
- `SORT_AREA_SIZE (=Y)`
Configure this parameter explicitly in case of very heavy sorting operations that have to be optimized extra. During the installation of an OHI Back Office major release, sometimes e.g., indexes are created on very large tables that benefit in case of a large value for this parameter. This could lead to a difference in throughput time of several factors when creating index (e.g., 2 hours may become 30 minutes). It is recommended to set the value to e.g., 50MB or 250MB (specified in bytes) in case very heavy sorting operations have to be expedited temporarily. The relevant quantity of memory is allocated per session in case this is required, so a value of 50 MB is not desirable when users log on (but in that situation `WORKAREA_SIZE_POLICY` normally will be set again to `AUTO`). Otherwise, for certain sorting operations an unexpected high volume of memory could be allocated.
- `HASH_AREA_SIZE (=Y)`
Configure this parameter explicitly as this may for example benefit heavy join operations during OHI release installations.

Both the sort and hash area size parameters can be retained during normal runtime operation of the applications as the workarea size parameter setting at `auto` will result in ignoring these parameters during normal use.

Parameters for Rollback (undo) Space

Management of the rollback space has to be automated by Oracle. The quantity of rollback space rendered available determines the throughput time for some batches as well. It is recommended to allocate some GBs of rollback space (or to allow for having this much space available by means of e.g., auto-extendable data files for the relevant table space).

The following parameter values are recommended:

- `UNDO_RETENTION (=10800)`
This value indicates the time to Oracle as to how long the rollback space should not be overwritten to provide a long-term statement - read 'consistent' - image. This value is provided in seconds. 3 hours would be an appropriate value. If sufficient rollback space is available (to be assessed via Oracle Enterprise Manager/Cloud Control) and based on one-hour settings (or more) 'snapshot too old' messages pop up, then the software has to be adapted to it (an incident message has to be reported).

Parameters for parallel execution

If you want to enable parallelism for the initial phase of batches please consult the previous paragraph about Batch Scheduler Performance and the optional use of parallel execution by enabling this through certain parameter settings.

Please note that parallel execution is disabled by default when an OHI related session logs on. Parallel query, DDL and DML are disabled. This is done by the OHI specific logon trigger for the following accounts:

- The OHI table owner and the OHI batch account
- The OHI user accounts (presence in `ALG_FUNCTIONARISSEN` is sufficient)
- The accounts that have received a grant on an OHI table owner object (except for `SYS` & `SYSTEM` when inappropriate granted) or that are granted a standard OHI role.

9.3.2. Statistics

The statistics definitions have been changed since 10g. The 19g standard job is recommended for collecting statistics. This job handles collecting statistics of all schemes and for the dictionary statistics. The job will collect the statistics in the *maintenance window* for all objects in the database. The job determines the order of the database objects for which the statistics require altering. In that way, the most important statistics have been defined when a *maintenance window* ends.

9.3.2.1. System Statistics

For determining the optimum way to perform an SQL statement, the database uses a so-called execution plan. Depending on the postulated optimization objective for performing SQL statements - providing the first part of the result as soon as possible (`FIRST_ROWS_10 optimizer_mode`) or minimizing the time to determine all results (`ALL_ROWS optimizer_mode`), a different plan could be preferred.

In order to determine these ‘execution plans’, the ‘Cost Based Optimizer’ (CBO) will consider the speed of I/O of the system and the speed of the processors (the CPUs) of the server. To this end so-called ‘system statistics’ are required.

ATTENTION: The Cost Based Optimizer (CBO) is used by default to optimize the SQL statements as performed by the database. This is arranged via the aforementioned instance parameter.

9.3.2.2. What are System Statistics?

As of the launch of Oracle 9i, not only does the CBO have to keep account of the I/O costs, but also the CPU costs and the estimated required temporary table space (see the relevant columns in table `PLAN_TABLE`: `CPU_COST`, `IO_COST` and `TEMP_SPACE`).

The CBO since 9i can deal with CPU costs that are based on actual system characteristics, that allows for cost estimation, for more realistic comparison of the execution plans.

A statement that requires relatively little (physical) I/O, but that uses a lot of CPU because many database blocks have to be accessed in cache, could clearly benefit from system statistics because the outcome of the required time for a certain plan is closely related to reality (in the past CPU and I/O used to be as expensive in a calculation, while in reality I/O is much more expensive, or more slow than CPU, although this might differ per machine used).

Therefore, the criteria for making the choice for most likely best execution plan can be established more reliably by the CBO.

9.3.2.3. Collecting System Statistics

Introduction

In order to optimize the performance of the CBO, the system statistics have to be collected again if the configuration or the use of the system undergoes clear changes (e.g. because the data is increasing and/or the database files are moved to a different location).

Collecting system statistics refers to an analysis of the database activity by Oracle during a certain period of time that is representative for the average database load (and the system as a whole).

Not having system statistics could result in choices for incorrect (less well performing) execution plans (if there are no records in table `AUX_STATS$` under user `SYS`, there are no system statistics).

Standard Values:

It is possible to immediately create direct system statistics if these do not exist yet, even though no representative load is involved. A number of standard settings are used (which are better than no statistics at all). To this end it is possible to indicate that statistics have to be determined without there being a representative 'workload'.

Since DBMS 10g this is performed by default during instance startup.

It is possible to perform this manually by means of the following (SQL*Plus) command, under `SYS` (just like all other commands that follow):

```
exec dbms_stats.gather_system_stats('NOWORKLOAD')
```



Attention: For OHI Back Office environments the `NOWORKLOAD` option (method *Standard values*) is the recommended setting for collecting system statistics, until further notice. This relates to negative experiences with respect to the actual collecting of system statistics.

Measured Values

Subsequently, when OHI is executing a representative load, actual values can be collected. Collecting can be activated by entering the following command e.g. at 10 a.m.:

```
exec dbms_stats.gather_system_stats('START')
```

To end the collection period (e.g. around 3 p.m.), execute the following command:

```
exec dbms_stats.gather_system_stats('STOP')
```

ATTENTION: During this period, the instance should not be stopped. Otherwise you have to restart collecting the statistics all over.



Attention: Until further notice, the method of actual collected values should not be used, unless this is requested explicitly by OHI Development or Oracle Support Services.

Status Definition

The following SQL*Plus commands show the status of the system statistics collection job:

```
col sname format a20
col pname format a15
col pval2 format a30
```

```
select * from aux_stats$;
```

If the statistics are defined without a representative work load, this results in:

SNAME	PNAME	PVAL1	PVAL2
SYSSTATS_INFO	STATUS		NOWORKLOAD
SYSSTATS_INFO	DSTART		10-26-2004 13:59
SYSSTATS_INFO	DSTOP		10-26-2004 13:59
SYSSTATS_INFO	FLAGS	1	

If afterwards collecting is activated, this provides the following output:

SNAME	PNAME	PVAL1	PVAL2
SYSSTATS_INFO	STATUS		MANUALGATHERING
SYSSTATS_INFO	DSTART		10-26-2004 14:01
SYSSTATS_INFO	DSTOP		10-26-2004 14:01
SYSSTATS_INFO	FLAGS	1	
SYSSTATS_TEMP	SBLKRDS	125167	
SYSSTATS_TEMP	SBLKRDTIM	391710	
SYSSTATS_TEMP	MBLKRDS	97238	
SYSSTATS_TEMP	MBLKRDTIM	73860	
SYSSTATS_TEMP	CPUCYCLES	467124	
SYSSTATS_TEMP	CPUTIM	1434067	
SYSSTATS_TEMP	JOB	0	
SYSSTATS_TEMP	MBRTOTAL	759624	

In that case initial values have been entered for certain values that are defined. If afterwards collecting is stopped then this could result in the following output:

SNAME	PNAME	PVAL1	PVAL2
SYSSTATS_INFO	STATUS		COMPLETED
SYSSTATS_INFO	DSTART		10-26-2004 14:01
SYSSTATS_INFO	DSTOP		10-26-2004 14:04
SYSSTATS_INFO	FLAGS	1	
SYSSTATS_MAIN	SREADTIM	8.381	
SYSSTATS_MAIN	MREADTIM	.626	
SYSSTATS_MAIN	CPUSPEED	36	
SYSSTATS_MAIN	MBRC	4	
SYSSTATS_MAIN	MAXTHR	-1	
SYSSTATS_MAIN	SLAVETHR	-1	

What can be derived from this is that collecting statistics is based on a period of 3 minutes. In case the statistics collection is not successful the line with value STATUS for column PNAME could have value BADSTATS for column PVAL2, as represented below:

SNAME	PNAME	PVAL1	PVAL2
SYSSTATS_INFO	STATUS		BADSTATS
SYSSTATS_INFO	DSTART		10-25-2004 22:41
SYSSTATS_INFO	DSTOP		10-25-2004 22:41
SYSSTATS_INFO	FLAGS	1	
SYSSTATS_MAIN	SREADTIM	-1	
SYSSTATS_MAIN	MREADTIM	-1	
SYSSTATS_MAIN	CPUSPEED	221	
SYSSTATS_MAIN	MBRC	-1	
SYSSTATS_MAIN	MAXTHR	-1	
SYSSTATS_MAIN	SLAVETHR	-1	

In such cases, the collection of statistics has to be restarted.

Deleting System Statistics

For disabling/deleting system statistics

```
exec dbms_stats.delete_system_stats
```

is sufficient.

Committing is not required for the above actions.

System Statistics and Execution Plans

Collecting system statistics does *not* nullify the execution plans in the cache, as opposed to *editing the table statistics*. For existing SQL statements for which an execution plan was available, after the creation of system statistics no new execution plans will be defined again right away. These have to be deleted from the 'cache' first. For testing purposes, you can add an extra space to a statement to force the calculation of a new plan after the system statistics have been changed.

9.3.2.4. Table Statistics

In addition to having system statistics available, it is important to have up-to-date table (and index) statistics. The Cost Based Optimizer uses the statistics of tables and indexes (e.g. how many records are included in a table, what is the division of values in a column, how many records are there in an index) to identify the execution plan that best meets the optimizer requirements. To this end, these statistics have to be approximately the same as the actual content of the table (and index). The statistics could be 'estimated' based on a part of the content of a table or 'calculated' when going through the table content entirely.

In order to establish representative statistics, they must be updated on a regular basis if the table content changes. If the content changes often (insert, update or delete of many records) the statistics have to be gathered more often than for tables with more static content. When a new OHI release is installed, for the tables that are changed in structure or contents the gathering of statistics may need to be executed again.

In order to abstract from this issue, a 'table monitoring' functionality must be used in the database. Oracle keeps track of how many changes (inserts, updates, deletes) are performed on the data in a table. This data is used in certain utilities to determine statistics 'automatically' for the tables that require this. 'Temporary' tables (tables that are not permanent but that can store data only during a session) are dealt with in a different way.

When determining the statistics, Oracle decides what percentage of the data in table is used to gather statistics.

Considering that there are various complicating factors for adequately collecting table statistics, script `OZGISTAS.sql` is included in the reference software. This script is called during the installation of a new release.

The script implements required OHI settings (such as preventing and/or dropping column histograms, locking the statistics for certain tables, etc.) and uses spare runtime to collect statistics.

The script can only be run by the user who requires his objects to be analyzed. It contains two input parameters. Both of them are mandatory.

Parameter 1 involves the number of hours during which the script can run; providing a value is mandatory.

Parameter 2 involves the environment in which the script runs. Possible values include `OBD` (=OHI DATA MARTS) and `OZG` (=OHI Back Office).

When called by the installation menu (OHIPATCH) these are passed automatically.

9.3.2.5. Data Dictionary Table Statistics

Since a 10g database the creation of dictionary statistics is *mandatory*. To this end `dbms_stats.gather_dictionary_stats` procedure can be used under `SYS` account.

For a database with only OHI Back Office objects this takes about fifteen minutes. If the data dictionary has experienced a lot of changes since the previous statistics gathering e.g. by a schema import), you should gather the statistics again using the statement above.

9.3.2.6. Fixed Objects Statistics

Since the 10g database the creation of statistics is also mandatory for so-called “fixed objects”. Use the procedure `dbms_stats.gather_fixed_objects_stats` under `SYS` account.

Conditions for collecting these statistics include:

- The SGA size and the `init.ora` database parameters have to be stable.
- The system has to have a representative load.

9.3.2.7. Dividing I/O - if Possible

In the past it was common to divide the I/O over physical disks by creating multiple tablespace and data files and spreading those over the available disks according to the expected I/O loads.

Nowadays, we use mostly logical volumes and certain RAID levels, and leave the division of the I/O to the controller(s) (of which all sorts of variants are possible). In general, this should guarantee a suitable I/O division, certainly considering the availability of fast read and write caches with intelligent algorithms.

One of the few options that remain is the division across multiple controllers if these are available. For an I/O bottleneck problem this could offer a solution.

If it is at all possible, locate the redo log files on a separate and fast I/O device. It would be even better to put each redo log group on a separate device, or at least put each successive redo log group on a separate device (e.g. even groups on disk 1 and odd groups on disk 2).

The archived redo logs (if these are used) have to be written to a separate device as well, so that the other I/O do not form an obstruction.

These last two recommendations are not strictly necessary but should be your first action to consider if the transaction speed slows down and if statistics show that the redo log actions tend to reduce the speed.

9.3.2.8. Resource Management – Session Prioritization

The OHI Back Office database is used by online users, batch processes, web services, etc. Some of these batch processes run for long periods, requiring minutes or even hours of processing. Such processes can sometimes claim 100% of a processor for long periods of time.

If there are many long-running batch processes, they will keep the database resources busy and may leave insufficient resources for other sessions. As a result, for example the online performance or the web service performance may suffer.

It then becomes necessary to prioritize these sessions over the batch processes. There is no point in prioritizing sessions at the operating system level: if an Oracle session has acquired memory latches and its' associated process is given lower priority, other database processes needing these latches are forced to wait until the low-priority process has eventually finished.

The recommended way for prioritizing such sessions is to use Oracle's resource management options. The Oracle documentation describes in detail how to manage the many kinds of database resources, of which CPU is just one type of resource.

For the purpose of OHI Back Office our minimal recommendation is to prioritize online and other potential interactive sessions over (long-running) batch processes.

At this point you should be aware that Oracle distinguishes three groups of resource consumers by default:

- SYS_GROUP (sessions of SYS or SYSTEM)
- BATCH_GROUP (sessions of BACKUP, COPY)
- OTHER_GROUPS (any other session which does not map to an existing resource consumer group)

By putting the OHI batch user into BATCH_GROUP we can set its priority relative to the other resource consumer groups.

The prioritization of resource consumer groups is handled by resource plans.

Note that in Oracle 19c in multitenant mode, there are two types of resource plans:

- PDB resource plan
A resource plan for managing resources within a pluggable database (PDB). In multitenant mode, the OHI Back Office data and its resource consumer groups will reside in a PDB.
- CDB resource plan
This resource plan applies to the container database. It is used to manage resources between PDB's.

In other words, to prioritize BATCH_GROUP relative to OTHER_GROUPS we need a PDB resource plan.

The default PDB resource plan is MIXED_WORKLOAD_PLAN. Since this resource plan changes with different versions of the Oracle database you may want to create your own PDB resource plan.

Note that an Oracle 19c PDB resource plan has a number of restrictions:

- PDB resource plans cannot have a multilevel scheduling policy
- PDB resource plans cannot have more than 8 consumer groups
- PDB resource plans cannot have subplans.

The following example of a PDB resource plan limits BATCH_GROUP (containing the batch sessions) at 20% if there is contention for resources. This plan prioritizes SYS_GROUP (containing the SYS and SYSTEM sessions) over OTHER_GROUPS, which in turn has priority over BATCH_GROUP.

```
begin
  dbms_resource_manager.create_simple_plan
  (
    simple_plan      => 'OZG_PLAN'
  , consumer_group1 => 'SYS_GROUP'
  , group1_percent  => 100
  , consumer_group2 => 'OTHER_GROUPS'
  , group2_percent  => 80
  , consumer_group3 => 'BATCH_GROUP'
  , group3_percent  => 20
  );
```

```
end;  
/
```

CDB resource plans are used in Oracle 19c to manage the priorities of pluggable databases relative to one another. By default, each PDB is given 1 share. Allocate more shares to a PDB to give it more resources if needed.

However, in Oracle 19c it is not possible to manage the resources across multiple container databases on the same server. In that case you could use the Instance Caging functionality. See the standard Oracle database documentation for details.

Finally, if you are creating scheduling windows (for example a 'batch' window and a 'daytime' window) it is advisable to explicitly set the resource plan for these windows, because the default resource plan may not be suitable.

9.3.3. Collecting Performance Data

This section discusses the resources which can be activated to collect performance data.

9.3.3.1. Operating System Utilities

Simple use of some general operating system utilities (Unix based) will be discussed in some sections. These can be used to acquire insight into the load of the server(s). There are many more options, sometimes these are better and more advanced, but these tools allow for an initial impression of where the bottleneck is located in most cases.

vmstat

By means of e.g. `'vmstat 5'` it is possible to acquire insight into memory-related statistics every 5 seconds, amongst which page swapping (a value of 300 seconds, or 5 minutes, could also be useful to assess the average load in a screen during a longer period; an hour consists of 12 lines).

Whether or not page swapping takes place can be derived from the value in front of the column 'sr' (scan rate) in the group page. If this has a constant value in hundreds or higher, swapping is clearly involved (the value has to be compared with the number of available pages).

Additionally, for the CPU load the percentage of idle time is represented. If idle is nearly continuously close to 0, then the CPU is heavily loaded.

With regard to group 'procs', the number under r(un queue), b(locked) and (can be run but s)w(apped) is very interesting for an initial impression. Values not equal to zero indicate that processes are not serviced right away.

For more information, see the Linux manual pages.

iostat

By means of for example `'iostat 5'` the default output is given. In order to zoom in on I/O problems, specific options have to be used, but the four columns under CPU provide insight as to whether or not waiting for I/O is required. Column 'w' under CPU provides the percentage of the time required for I/O,

By means of `'iostat -xc 5 10'` it is possible to have overviews provided of the average service time 10 times during 5 seconds, the busy percentage etc. of all devices including how the CPU time is used. A busy device could possibly be relieved by distributing the database files better across the devices (if the database files are an explanation for the load of the devices).

sar

By means of the `sar` utility it is possible to register statistical data with a certain interval in a simple way. The data can be converted into readable reports by means of the same utility.

In order to optimize the use of this a set-up is required with an interval of e.g. 10 minutes of 'recording' (24 * 7). This allows reasonably well for zooming in on longer peak periods and in this way, data involving half an hour can be compared with statistical data on a database level (this will be discussed at a later stage).

By means of the following 2 lines in the `crontab` for `sar` (user `sys` on a Sun Solaris system) every 10 minutes recording can be made that are converted into a readable report at the end of the day (11:55 p.m.)

```
0,10,20,30,40,50 * * * * /usr/lib/sa/sa1
```

```
55 23 * * * /usr/lib/sa/sa2 -s 0:00 -e 23:50 -i 600 -A
```

CPU load for previous period (all 'snapshots' of the day):

```
sar
```

Reading memory use for previous period:

```
sar -r
```

Reading swapping of previous period:

```
sar -w
```

Reading the actual load five times every 10 seconds:

```
sar 10 5
```

Of course, there are advanced tools to implement monitoring of the system. However, because of a comparison with other customers and installations and providing available data to Oracle in case of performance problems, it is desirable to always have `sar` monitoring activated. The overhead is very limited, and the advantage is that this allows for tracing the cause of performance problems faster because sometimes certain causes can be excluded.

top/topas/prstat

By means of a tool such as `prstat`, `top` or `topas` (or other system utilities) it is possible to retrieve which processes are used by most CPUs rather fast. Especially if certain processes are slow or if the CPU use on the server level is high, then it is possible to zoom in on the processes that require most CPU.

In case of frequent performance problems, it could also be desirable for Oracle to use such a utility. Access to one of these utilities (available via the 'remote connect' account) could speed up the process.

swap

By means of the `swap` command, certain data can be retrieved involving the swap memory. This could be desirable in case swapping seems to be involved.

Listing out the swap memory:

```
swap -l
```

Summary of the swap use:

```
swap -s
```

OS Watcher

Oracle's OS Watcher (OSW) is a collection of UNIX shell scripts intended to collect and archive operating system and network metrics to aid support in diagnosing performance issues. Information on how to setup and use OS Watcher can be found in the following My Oracle Support notes:



My Oracle Support note 301137.1, OS Watcher users guide



My Oracle Support note 461053.1, OS Watcher graph users guide

9.3.3.2. Database Monitoring

Oracle Enterprise Manager

For monitoring the database for recent activities there are various options. Particularly, by using the Oracle Enterprise Manager (OEM) Cloud Control in combination with an OEM repository and configured agent processes, a lot of useful information can be retrieved.

Automatic Work load Repository

We recommend collecting data periodically in a relatively easy way, so that comparisons can be made between data of various customers. In case of more general performance problems, this data allows for the possibility to obtain a clear view as to the bottlenecks in the database (presuming that there are any).

Therefore, we use information from AWR, the Automatic Workload Repository, which is installed by default and configured since DBMS 10g. AWR creates periodic ‘snapshots’ of the database statistics automatically and stores them in the SYSAUX table space. Reports can be released on these snapshots in tables.

By default one snapshot is created every hour, only at the CDB level. Every night a job runs which automatically removes snapshots taken more than 7 days ago (the default retention setting).

OHI advises to create snapshots at the PDB level at least every 20 minutes, but we prefer each 10 minutes, and use a retention period of at least 45 days. This can be enabled through:

```
dbms_workload_repository.modify_snapshot_settings  
(retention => 45*24*60, interval => 10)
```

This consumes more PDB specific SYSAUX space but can be very helpful to investigate problems (and look for example whether these problems also occurred the previous month in the same way). The current settings for AWR at the PDB level can be queried using the dictionary view AWR_PDB_WR_CONTROL.

Beware for an interval value of +40150 00:01:00.0 in the column SNAP_INTERVAL as this means once in 40150 days. This is a problem in the database code occurring after plugging a PDB out and in and is/was still present in the database 19c RU Oct 2019 version.

To enable automatic snapshot creation at the PDB level the parameter AWR_PDB_AUTOFLUSH_ENABLED should be set to TRUE. This OHI-specific advice contradicts the standard Oracle advice given in the manual but does help a lot when investigating OHI database specific performance issues.

PL/SQL Hierarchical Profiler

For OHI batch processes it is possible to activate the PL/SQL hierarchical profiler for a specific running process (a script request). This needs to be done through the table owner account. To accomplish this, the script request ID needs to be specified as well as a database directory name for which the OHI table owner account has ‘write’ privileges.

To find a suitable database directory:

```
select TABLE_NAME  
from   DBA_TAB_PRIVS  
where  GRANTEE      = <<'OHI TABLE OWNER>>'  
and    TYPE          = 'DIRECTORY'  
and    PRIVILEGE     = 'WRITE';
```

The command to execute, with serveroutput enabled, is:

```
begin  
  alg_script_pck.start_plshprof  
  ( p_sav_id => 999999
```

```
, p_directory_name => 'VOORBEELD_DIR'
);
end;
```

Of course 999999 is a fictitious script request id.

Beware: the trace file that is created might grow very large in a few minutes.

To stop the profiling of the process, execute in the same way:

```
begin
  alg_script_pck.stop_plshprof
  ( p_sav_id => 999999 );
end;
```

The trace file that is created in the specified directory can be used to generate a set of very interesting HTML reports by means of the database Oracle Home based command `plshprof`. The name of the trace file will start with 'OHI_plshprof' followed by the CDB and PDB container name, the script request id and the timestamp of the start of the profiler session.

The following command will generate the output reports:

```
plshprof -output <outputfilename>.html <tracefilename>.trc
```

The actual database code that implements the profiling is available in database package `DBMS_HPROF`.

9.3.3.3. RAC

In a RAC environment, each AWR snapshot gathers information regarding all instances in the cluster. This data can be aggregated in a database wide AWR RAC report.

AWR – Reporting Statistics

When statistics are collected, they can be used to run reports about certain periods. These periods are characterized by a 'start' snapshot and an 'end' snapshot (identified by numbers; a list of snapshot periods with identifying numbers is displayed by each AWR utility that requires the snapshot numbers).

By means of report script `awrrpt.sql` it is possible to create a general report about a certain period. If a problem is encountered during a certain period, it is useful to limit the report period further. If a reduced performance was experienced from 9:45 a.m. to 11:55 a.m. it would be recommended to run the report for the period 9 a.m. to 12 p.m. (assuming snapshots were only made once an hour).

The script `awrrpt.sql` will suggest a name for the 'spooled' report file. Please accept the proposed name.

All sorts of statistic information are displayed in the report. Additionally, various lists are provided of the heaviest SQL and PL/SQL statements.

For statements suspected to be responsible for the performance degradation (typically statements in the top 10 lists), you can create a more detailed report by means of a separate script `awrsqrpt.sql`. In this case the identifying hash value (SQL ID) has to be provided. This SQL ID is reported for each statement in the report created with `awrrpt.sql`. Identifying 'suspect' statements will normally be performed by the development department of OHI Back Office (the development team) after an AWR report is delivered.

However, it is not necessary to examine problem statements that may be present in the OHI Back Office application as customer of OHI Back Office. The OHI Back Office development team should normally perform this task. By means of the AWR reports and any specific reports pertaining to certain statements, the development team can focus on pertinent solutions for any problems.

ADDM Report

When you deliver AWR reports to the development department for OHI Back Office, you should also deliver the ADDM (Automatic Database Diagnostics Monitor) report for the relevant snapshot range. This ADDM report provides advice regarding the AWR collected data. Run script `addmrpt.sql` to generate the ADDM report.

This script performs an ADDM analysis for the relevant range and creates a report about it.

In order to display the report of the *last* ADDM analysis, script `get_latest_addm.sql` can be used; this script does *not* first perform an analysis.

RAC

In a RAC environment, an ADDM report is specific for a node/instance. Separate ADDM reports must be collected for the different instances.

9.3.3.4. Tracing Sessions

Whereas AWR is used to collect all of the database data, in certain cases it may be desirable to zoom in on a specific process or a certain database session, e.g. when a specific session causes problems and no general problems are detected.

For such a session, creating so-called 'trace files' is recommended. Even though this should be a familiar task for a DBA, some indications are provided as to how to deal with tracing.

Setting a Session to Trace Mode

Activating tracing in a session is relatively easy by entering the following command:

```
alter session set sql_trace = true;
```

However, sometimes it may be required to set an ongoing session to trace mode. This is possible by entering command:

```
execute dbms_system.set_sql_trace_in_session(&sid,&serialnr, true)
```

This has to be performed with an account that has relevant rights.

While doing so, it is necessary to determine the values for 'sid' and 'serial#'. These can be retrieved from view `(G)V$SESSION`. Column `USERNAME` can be used as a filter to create a selection for the user sessions that are slow.

However, users who have opened multiple screens from the menu in OHI Back Office will have multiple sessions in the database. When a slow action in a screen has to be traced, it is possible to do as follows:

- Start the screen from the menu (or the calling screen once it appears in the start-up screen).
- As a first action, before performing the slow action, determine the 'sessionID': Start the Information window by means of the Help menu and the Info menu option. Determine the value of 'sessionID'. This value identifies a session by means of column `AUDSID` in `(G)V$SESSION`. As long as the 'problem screen' does not close, this value will remain constant and you can use it to determine `SID` and `SERIAL#` from `(G)V$SESSION`:

```
select sid
,      serial#
,      (select spid
        from   v$process p
```



```

        where p.addr = s.paddr
      )
    oracle_process
from    v$session s
where   audsid = &sessionid

```

- Use the `sid` and `serial#` values in order to perform the above command and to set the session to trace mode.
- Perform the slow action.
- Check the trace directory to see if a trace file has been created with the name `oracle_process` number as provided by the above query.
- Close the screen. The trace file will be closed, flushing any remaining info from the buffer.

If case a screen is already blocked by a long-term action and you want to trace the session, try to identify it in the following way:

- Determine which oracle process uses a lot of CPU in the relevant database (by means of a peak-like tool). If there is no such process, the problem is supposedly not located in the database but in the forms. It is possible that the forms process consumes a lot of CPU. The database session does not have to be traced in that case.
- By means of the following query, it is possible to determine which sessions the relevant user has in the database. The oracle process that consumes a lot of CPU should have the ID available in the list of process numbers in the 'SQL_Net' column as provided in this query.

```

select row_number() over (order by b.process)||'.' as row_number
,      b.osuser          osuser
,      b.username        user
,      b.sid             sid
,      b.serial#         serial#
,      b.program         program
,      b.process         UNIXpid
,      d.spid            SQL_net
,      a.sql_text        sqltext
from    v$sqlarea        a
,      v$session         b
,      v$process         d
where   b.sql_address = a.address (+)
and     b.paddr       = d.addr
and     b.username    = upper('&usr') /* Oracle database account of the user
*/
and     b.osuser      = 'oracle' /* f60webm processen are running under the
Unix account "oracle" */
order  by row_number() over (order by b.process)||'.'
/

```

- Use all `sid` and `serial#` combinations that match the 'heavy' oracle process in order to set the relevant sessions to trace mode.
- Check if a trace file appears (see above). If this is not the case, then it is possible that the session requires a lot of time with a single SQL statement. The trace file will only be created after the statement provides results or if the session starts with another statement. (This will typically be the case for heavy statements in batches, for screens this happens less frequently).

For running batch processes, searching the sessions based on a username is not a solution if multiple batches run at the same time. The batches all run under the same Oracle user, often referred to as `BATCH`. The batch scheduler also runs under this Oracle user and has to be recognized to be excluded. The batch scheduler has a value that starts with `SYSS004S` in column `MODULE` of `V$SESSION`. This session should therefore not be traced.

If multiple sessions are active in the user batch, it is possible to identify the correct session by selecting the ID process the session has under Unix. Usually, the session will return as a heavy

Oracle process in a tool that displays the top sessions (see above). The process ID that matches the Oracle process can then be used to select the correct values from the `V$SESSION` and `V$PROCESS` view by means of the following statement:

```
select sid
,      serial#
,      process
,      username
from   v$session s
where  s.paddr = (select addr
                  from   v$process
                  where  spid = &oracle_proces_id
                  )
```

Should several heavy Oracle processes run, request the start time of the heavy processes by means of `ps` command (`ps -eaf | grep <procesid>`) and compare this with the start time of the script requests as displayed in the screen. This will have to be about the same as the start time of the running script request. In this way it can often be determined which Oracle process matches which script request.

Trace File Points of Attention

Some points of attention

- If a session is set to trace mode then normally a trace file will be created in the location provided by means of the initialization parameter `DIAGNOSTIC_DEST` (within a subdirectory 'trace' deep within the directory structure designated by this location setting).
- The names of the trace files normally have the form `<$ORACLE_SID>_ora_<process_id>.trc`. In this case the `process_id` can be determined by means of column `SPID` from view `V$PROCESS` (see previous section). For MTS sessions the process id of the 'server' process will be used (a so-called `Sxxx` process, `S001` for server process 1, etc.). Such a server process will also write statements of other sessions to the trace file. An MTS configuration is not as usable for tracing sessions.
- By means of the initialization parameter `MAX_DUMP_FILE_SIZE` a limit can be set for the size of trace files (in system block processes or in KB or MB). If this limit is set rather low (some MBs e.g.), then it is possible that trace information can be registered only for the first few minutes of an action, because afterwards the maximum volume is reached for the trace file. In order to trace a slow action in a screen this is generally not a problem, for tracing a slow batch sometimes hundreds of MBs of data per hour could be written already. Therefore it is of importance that these parameters are set to e.g. unlimited or to a high value when tracing long-term batch processes (and of course sufficient space has to be available on the volume on which the trace file will be written; if necessary use a symbolic link for the relevant directory for writing the trace files in a different location which has sufficient space available).
- If a trace file is filled up then it is no longer possible to get the writing of the trace information started by setting the parameter value higher dynamically. Only once a new session is started trace information will be written again.
- However, tracing may also be activated while no trace information is written (see the previous remark about tracing a screen which has been blocked for a longer period). Often the cause is that the session is occupied for a longer period with one and the same SQL statement and does not retrieve records (yet in case of a select statement). No trace information will be written in that case. This could be the case for batches if an SQL statement is included which would have a result only after a couple of hours. Whether or not this is concerned can usually be verified by assessing if the session has been occupied with the same SQL statement for a long time in Oracle Enterprise Manager.
- If possible, it is desirable to have the trace file as complete as possible. Particularly for the slow screen actions it is possible to close the screen after 'reproducing' a slow action after which even the trace file is closed. This will also contain certain counting information that

could be useful for assessing the trace file. Also for the batch it is recommended to have a trace file about the entire process that is as complete as possible, but usually this is not possible due to the size. Or it is not possible because the batch has been running for too long and does not end.

Formatting a Trace File

When a session is set to trace mode and the trace file containing the trace information has been identified, then it is desirable to format it in a format what is easily readable and useful. It is possible to already format a trace file while it is still being created. If possible, it would be better to wait until it is entirely ready.

Formatting has to take place via utility `tkprof`. The version that is located in the database bin directory of `ORACLE_HOME` has to be used. Switch therefore in advance the environment settings to the correct `ORACLE_HOME`.

In order to get the heaviest statements of the trace file on top in the formatted output, it is of importance to apply good sorting techniques. This optimizes the output because in that case it is not necessary to look for the slowest statements very long and therefore slow statements are missed less easily.

Useful sorting can be done on the sum of the elapsed times of parse, executing and fetching. In that way most long-term statements will for sure end up on top in the output. For this, parameter value `'sort=prsela,exeela,fchela'` is very useful. An example of call:

```
tkprof ontw_ora_7222.trc trace.txt explain=ozg_owner/ozg_owner
sort=prsela,exeela,fchela
```

Some points of attention:

- Run `tkprof` under the account that has relevant rights for the objects used, for example the owner account of OHI Back Office. Or use an account that has `SELECT ANY TABLE` rights.
- Given that `tkprof` determines execution plans for the statements in the trace file at the moment on which `tkprof` is running, then it is important that nothing in the database is changed between the creation of the trace file and running `tkprof` (for example, the table statistics do not have to be updated in the meantime).

9.3.3.5. Statements – Determining Execution Plan

If certain SQL statements are slow it is desirable to examine the execution plan by consulting the explain plan output in which the execution plan is represented. By means of `tkprof` the execution plan is represented for the statements by default in the trace file. But it could also be desirable to assess a change in a statement in terms of execution plan.

In order to determine an execution plan, it is important to have access to the right `PLAN_TABLE`. For this, the global temporary `PLAN_TABLE` has to be used. To facilitate this, it is *mandatory* to remove any available `PLAN_TABLE` tables under all schemas. The following action list can be used to obtain this:

- Select the owners of the existing `PLAN_TABLE`s:

```
select owner from dba_tables where table_name='PLAN_TABLE'
```

- Remove the `PLAN_TABLE`s for the various owners.
- Remove any public and private synonyms for `PLAN_TABLE` if they refer to a `PLAN_TABLE`.

Assessing the execution plans of slow statements is something that has to be performed by the development team of the OHI Back Office application and will not be discussed further in this document.

9.3.3.6. Check Index Use

It could be desirable to check if certain indexes are used. For this, it is required to activate the relevant monitoring for an index:

```
alter index &index_name monitoring usage
```

Via `view v$object_usage` it is possible to trace if an index is used.

By default, this is not required and the development team of OHI Back Office will be asked to activate this if required for studying a certain performance problem.

9.3.3.7. Use of SQL Access Advisor

Script `OZGTUNES.sql` (available in `$OZG_BASE/sql`) will create a report for a (problematic) SQL statement.

This can be run if e.g. an AWR report indicates a SQL statement as problematic.

The script starts the *SQL Access Advisor* and hereby generates information that contains advice regarding the steps to be taken in order to optimize the SQL statement performance. The output can be sent along with the rest of the mandatory information for incident notifications of performance problems. The OHI Development can then probably solve the problem faster.

Parameters `OZGTUNES.sql`

- The ID of the SQL statement about which information/advice has to be printed.
- Optional: the logical name of the directory (= directory object in the database). If nothing is indicated then `OZG_TMP` is used.
- Optional: product with permissible values `OZG` (=OHI Back Office) and `OBD` (OHI Data Marts). Parameter influences advice: either OLTP or datawarehouse. If nothing is indicated the script uses `OZG`.

Examples of Possible Calls

```
@OZGTUNES ax0ufbmack2cg " "
```

This command writes file `ax0ufbmack2cg.out` to `OZG_TMP` with product `OZG` (OLTP).

```
@ OZGTUNES ax0ufbmack2cg " OBD
```

Writes file `ax0ufbmack2cg.out` to `OZG_TMP` with product `OBD` (OHI Data Marts).

```
@ OZGTUNES ax0ufbmack2cg OZG_DIR "
```

Writes file `ax0ufbmack2cg.out` to `OZG_DIR` with product `OZG` (OHI Back Office).

```
@ OZGTUNES ax0ufbmack2cg OZG_DIR OBD
```

Writes file `ax0ufbmack2cg.out` to `OZG_DIR` with product `OBD` (OHI Data Marts).

9.3.3.8. SQL Profiles

Using the *SQL Tuning Advisor*, it is possible to store SQL Profiles for specific statements. It is allowed to store SQL Profiles.

9.3.4. Assess Settings during 'Regular' Performance

This component will be developed in greater detail at a later stage.

9.3.4.1. Database

Components to be checked:

- Hit rate of the buffer cache. At least 90%, but preferably more than 95%.
- The same for shared pool.
- Checking the PGA settings

9.3.4.2. Preventive Application Management Actions

This component will be developed in greater detail at a later stage.

Issues to be considered:

- Max number of batches
- Max number of processes per batch
- Using parallel execution during the workload phase of a batch

9.3.4.3. Examining Performance Problems

If you receive complaints about performance, we recommended you zoom in on the problem before drawing conclusions as to a possible cause. In this section a description can be found about a possible approach that can be applied.

9.3.4.4. Finding out the Scope

When performance problems occur, it is important to first find out the extent of the problems at the moment they are encountered. The problem may only occur during certain periods; then determining these periods is important too):

- Does everyone experience the problems on all locations regardless of the operation?
- Is it possible for the problem to be encountered in various workstations but does it only occur when logging in on the system with a certain name?
- Is the poor performance related to the location (only certain workstations or PCs)?
- Does the poor performance occur only in certain sections of the application?
- Can the poor performance be detected only if a certain action is performed on the data?
- At the moment the problem occurs, is it just OHI Back Office that has a poor performance on the PC or are all of the actions in the relevant workstation(s) experiencing this problem?

If the problem is time-driven it is of importance to note down the exact time (up to the minute with a note specifying where the time is read; the clock of the system could differ by a couple of minutes).

With the answers to these questions the scope can be determined. This would probably lead to one of the following delineations:

- A PC-specific problem is involved.

In these cases the problem can be caused by the hardware, the network connection, the operating system, certain applications that run on the PC, alternative settings on the PC or, more tricky, a combination of these problems.

It is typical that the problem is encountered on one PC regardless of the user that logs in or the application and that the same user experiences no problem on other PCs.

- The problem is related to the network.

Only certain workstations in a specific section of the building or a specific building experience this problem.

Typically, the problem is not encountered when the same user performs the same operations on workstations in a different section of the building.

- The problem is related to the server (database and/or application server).

All applications on the server(s) experience the problem. In that case, via monitoring utilities it is possible to check that the server is overloaded at a certain point (too much swapping/paging is involved; the server is occupying nearly 100% CPU all the time or is waiting for I/O regularly; or a combination of the above).

Typically, all users experience problems with the slowness in the applications that run on the relevant server and all server applications are rather slow.

- It is either a database or a general application problem.

All actions on the relevant database are performed relatively slowly (generally speaking this will result in a server problem but in that case particularly the database actions will cause problems).

Typically, one particular application runs very slowly instead of other applications that run on the same server (if available).

From the outside, it is often hard to make a distinction between the server and the application-specific database problems.

- The problem is caused by a problem in the application.

The problem is only encountered during certain actions. Depending on the gravity of the problem, only the relevant action experiences problems in performance or other components of the application are burdened by it as well because the database and/or server is overloaded.

Typically, some of the actions in the application are performed at a normal speed, but certain actions are much slower than before (if there is a frame of reference).

The problem is caused by wrong application use.

Typically, new users are involved or users who perform certain actions for the first time. Or a function is involved that is used for the first time since its introduction. Chances for incorrect use of certain functions or wrong expectation are the highest.

The problem is caused by a certain bottleneck in the infrastructure of the application.

Typically, for these problems no clear general overloading can be observed. A possible cause could be that many processes change the same file (outside of the database). An example, which actually did occur, involved many user sessions that wrote many messages at the same time in the Apache log files.

9.3.4.5. Zooming in on Problems

In order to figure out the cause of the performance problems experienced, the following suggestions are made:

- Checking the workstation: What is the CPU load of the PC like (e.g. use a 'task manager' to monitor the CPU load; if it is near 100% and this is not caused by the application process then usually this could be a reason).
- Does the relevant action (execute it as similarly as possible) run comparably slowly in a different workstation? If not, does it make a difference if the same account is used in a different workstation?
- Is a peak or an overload encountered on the server?
 - Is there hardly or no idle CPU time?

- Is 'waiting for I/O' involved?
- Is there a high paging or swapping rate and do all active processes insufficiently fit in the work memory of the server?

In case of one or more of these phenomena, the cause has to be ascertained.

In case of a heavy CPU load the processes that cause the problems have to be defined as well as the application/database they belong to.

In case of a high I/O rate, the processes that cause the problem have to be checked (as well as the application they belong to).

If the memory use is too high, then what has to be checked is if there are certain processes that require large amount of memory or if the number of processes is higher than normally. The cause for that problem will then have to be ascertained.

- If no peak load is experienced, is there a clearly heavy process available at the moment the performance problems are experienced? Apparently the relevant operation is then very heavy and requires to be zoomed into.
- In case certain processes are heavy (a lot of I/O, CPU or memory usage) the kind of process has to be ascertained: For OHI Back Office only a few types of processes can be recognized:
 - an Oracle client process that handles the SQL and PL/SQL work;
 - a forms server process that runs on the server and that matches the applet screen processes that run on the workstations;
 - a batch 'client' process (SQL*Plus, SQL*Loader, import/export or shell process).

In almost all cases the Oracle client process, which matches a certain database, will be the culprit. At times a web forms server process may crash or spin (users should never have such processes run for more than few CPU percentages; if this happens, it has to be killed). If the Oracle client process is identified then by means of the process ID it is possible to determine which database session matches it, after which it is possible to examine the relevant database session further.

- If a certain process is identified as the culprit it can be examined further. For Oracle client processes additional tools are available without problems, for other processes even the source will have to be examined.

Oracle client processes can be set to trace mode so that the produced trace file can be assessed afterwards by formatting it with `tkprof`. Additionally, it is possible to zoom in to the SQL statements that perform the process and to have a look at the statistics in Oracle Enterprise Manager or by means of a different tool.

In the future zooming in on problems will be focused on what appear to be encountered in the database. If research shows that the problem is to be found elsewhere, then this section will probably not be of any further use.

Before continuing, the changes made since the problem has occurred will also have to be discussed further:

- Has the frequency increased?
- Is there a new version of a component of the total 'stack' technology (hardware and software) in use?
- Have multiple changes been implemented (this makes it more tricky to have a clear view as to the cause, which would call for making as many changes as possible successively rather than simultaneously).
- Is it for certain that the problem has not occurred before?
- Is the problem related to time periods? In other words, is the problem encountered during the beginning or the end of the week, the month or the year?

The more information available, the more plausible certain causes may be or the more easily others can be excluded.

9.3.4.6. Search the Oracle Database for a General Problem

At a certain moment a general database problem can be suspected. In that case it has to be clear that no other causes are involved:

- No additional users are involved which would stretch the limits of the machine.
- There is sufficient physical memory available or, not too much swapping is taking place.
- There are no processes that are more prominent in terms of consumption.

If this impression of having problems with the entire database performance persists, then the general database matters should be focused on. Particularly, the AWR output which reports on the slow period could be useful.

Now examine the following:

- Is the problem an I/O problem:
 - do the service times in AWR no longer correspond with the 'good' periods?
 - has the buffer hit percentage clearly decreased?
- Are there any unusual events in the top 5 of the wait events with a substantially higher waiting period than normally?

In case e.g. a latch problem is involved, then this is to be zoomed into. There could be numerous other problems like this. An initial approach is to search on My Oracle Support for similar problems.

Generally, general database problems - not caused by some specific processes - do not tend to cause application problems that easily. A parameter setting value that is too low would more likely be a cause to a problem.

9.3.4.7. Search Oracle Sessions for Problems

If it is clear that certain sessions are perceived as top processes, they require to be examined. At a later stage, more information will be provided about a possible approach, using the previously described possibilities.

9.3.5. Notifying Performance Problems

If performance problems occur which are caused by an OHI Back Office operation, you will need to supply the following information when registering a performance incident:

- ADDM report
- AWR report in HTML-format
- Output of Object check (see *Installation manual, Oracle Health Insurance (patch) releases* on Beehive Online; start the Object check via the *application menu* so that the *runtime* checks are performed!)
- In case of `ORA-00600/07445` error messages; the number of the Service Request logged with Oracle Support Services for the database product, and the corresponding alert- & tracefiles

Based on this information it is possible OHI development will ask for more information with respect to specific SQL statements. In that case a delivery of the following will be requested:

- AWR SQL report in HTML-format
- Output of OZGTUNES.sql

For more information, see [Collecting Performance Data](#)

9.4. PERFORMANCE APPLICATION SERVER

This section briefly discusses the performance of Oracle Forms Services and WebLogic Server.

9.4.1. Load Balancing Oracle Forms

In order to implement Load Balancing for Oracle Forms (the screen interface component of OHI Back Office) the following My Oracle Support note can be used:



Troubleshooting Forms Performance Issues (Doc ID 19895.1)

9.4.2. Investigating WebLogic Server issues

When there are doubts about the performance of the WebLogic JVM processes we advise to investigate this using Java Flight Recorder (JFR). This can give a very detailed insight into performance of objects running in the Java Virtual Machine because WebLogic Server has integrated the WebLogic Diagnostics Framework (WLDF) with JFR.

JFR can be activated in a production situation because the performance impact is quite minimal.

To use WLDF with JFR please read [this relevant paragraph](#) on docs.oracle.com.

NOTE: before you unlock the commercial features (see below) make sure you have a license. Consult My Oracle Support: **Support Entitlement for Java SE When Used As Part of Another Oracle Product (Doc ID 1557737.1)**.

These are the basic steps to activate JFR for a running Managed Server process:

- Obtain the process id (pid) for the running MS process.
- Make sure the JDK which is used for running the MS process is in your 'path'.
- Unlock the commercial features once:

```
jcmd <pid> VM.unlock_commercial_features
```

This normally responds with (when <pid> was 99999):

```
99999:
Commercial Features now unlocked.
```
- Specify to start the flight recording for a period of 600 seconds:

```
jcmd <pid> JFR.start duration=600s filename=JFRrecord10min.jfr
```

This will respond with something like:

```
99999:
Started recording 2. The result will be written to:
$DOMAIN_HOME/JFRrecord10min.jfr
```

You may use the JFR.stop option to stop the recording before the 10 minutes have passed, or use the JFR.check option to check on the current recording.

When the recording has finished, copy the recorded .jfr file to your desktop (or another location where you have access to a JDK). Start Java Mission Control (JMC) to analyze the recorded data in the .jfr file.

JMC can be found as jmc.exe in the bin folder of a local JDK installation, in Windows.

For complete details about the Java Mission Control interface, see Java Mission Control User's Guide at the following location:

<http://docs.oracle.com/javacomponents/index.html>

9.5. GETTING INFORMATION AFTER “ORA-12801”

Parallel Query is used by OHI in during the following activities:

- work gathering phase of OHI Back Office batches
- installation of OHI patches,
- actions in OHI Data Marts
- actions in Subsetting and Data Masking.

Unfortunately using Parallel Query may mask the cause of errors, because only the generic message “ORA-12801: error signaled in parallel query” is shown. If the alert log of the database does not show more information, you can temporarily set a database event. See document 1187823.1 “How to Analyze an ORA-12801” on My Oracle Support.

As a DBA, execute:

```
ALTER SYSTEM SET EVENTS '10397 trace name context forever, level 1';
```

Now execute the action that resulted in the ORA-12801 error again. After the error re-occurs, check the alert log. This should now contain the actual error.

Don't forget to turn of the event.

As a DBA, execute:

```
ALTER SYSTEM SET EVENTS '10397 trace name context off';
```

To check which events are enabled, execute:

```
SET SERVEROUTPUT ON
DECLARE
    l_level NUMBER;
BEGIN
    FOR l_event IN 10000..10999
    LOOP
        dbms_system.read_ev (l_event,l_level);
        IF l_level > 0 THEN
            -- note: requires exec permission for DBMS_SYSTEM
            dbms_output.put_line ('Event '||TO_CHAR (l_event)||
                ' is set at level '||TO_CHAR (l_level));
        END IF;
    END LOOP;
END;
/
```

9.6. PREVENT UNNECESSARY LOCK MESSAGES

The OHI Back Office application implements an intensive locking strategy to safeguard consistency of the data and a watertight business rule mechanism. This is a unique and distinguishing characteristic of the OHI Back Office application. But it may mean that you need to adapt table storage settings to prevent ‘preventable’ (unnecessary) lock messages that can occur due to high concurrency load.

What do we mean with this? First follows a short explanation of some concepts.

- A table consists of a number of database blocks.
- An active transaction on a database block (having a lock or posted update, delete or insert on a record in the block that is not yet committed or rolled back) requires a transaction entry in that block.
- A transaction entry requires approximately 23 bytes in a block (the exact nr of bytes is operating system dependent).
- In each block there are a number of initially reserved transaction entries in the block header.
- When there are more transaction entries needed than initially reserved the free space in a block can be used for additional ‘dynamically’ allocated transaction entries.
- When a new transaction on a database block cannot acquire a transaction entry due to a lack of free space this is handled in the same way as if a lock on a record cannot be acquired. Depending on the executed statement the statement fails or hangs/waits until the transaction entry can be acquired.

What does this mean?

- When you have a lot of concurrent sessions that execute transactions on the same block (this is table and process dependent) and the free space in the block is already quite exhausted by former updates of records that consumed the free space, it may be a next session raises an exception during an attempt to lock data with a NOWAIT clause.
- Even though the record to be locked is not actually locked the exception is thrown that the ‘resource is busy’, meaning the lock cannot be obtained.
- This is no ‘real’ lock but a message indicating there is transaction congestion within the block.
- It is very hard to distinguish these lock messages from actual lock messages, where the data was locked by another session.

When you expect processes suffer from these unnecessary lock messages and these messages do obstruct your processes, you should adjust storage parameters for the table where this occurs. You have the option to increase the INITRANS setting for the table involved and/or increase the PCTFREE setting.

The INITRANS value determines the initially reserved number of transaction entries in the block header of a block where PCTFREE determines the amount of space to keep free in a block for row length increase. Inserts are allowed as long as that PCTFREE is obeyed.

Currently within OHI Back Office minimum values of 4 or 16 are implemented for the INITRANS setting for ‘regular functional’ tables, depending on the table. You may increase this value; OHI Back Office will implement the 4 or 16 as a minimum value during release installations. These values are based on an assumption that the maximum nr of parallel sub processes that is used is in the range 16 to 20. If you clearly use more parallel sub processes it might be needed to increase your actual setting for certain tables proportionally when you expect ‘lock’ messages occur because of a too low initrans setting.

You should trade off whether you want to reserve additional space in a block for transaction entries, which cannot be used for actual data storage, against the chance and implications of an unnecessary lock message due to high number of concurrent processes.

Do not blindly set high values on all tables, this will cost you valuable storage space and decrease your IO rate. Only implement higher values on tables that clearly suffer from this issue.

If you are not sure whether a table suffers from this problem please contact OHI Support and ask for assistance in identifying potential problematic tables (this is dependent on your database block size, the order and way you run processes and the parallel degree you implement for batch processes).

An OHI batch process that encounters a record lock issue, will log a message that will - in most cases - show the table involved. This can help in identifying tables that may suffer from this issue. Typically this information is not present when a lock attempt fails due to an unavailable transaction entry but this does not mean that when this info is absent this is always caused by this problem. Other improvements and investigation may help in identifying the failing statement and whether your environment suffers from this problem.

Beware that when you adjust the INITRANS or PCTFREE setting the new values will only be applied to newly acquired blocks. All already acquired blocks do implement the original settings. A reorganization of the table (for example by moving the table) is needed to implement the new settings on existing blocks.

9.7. ANTICIPATE PREVENTIVE RESOURCE MESSAGES

If a certain type of resource is insufficiently available (memory, free disk space, free table space, reserved memory space for extra database sessions, etc.) error messages may be encountered in the application.

The objective of this section is to discuss several ways to undertake actions to prevent certain resource-related messages as much as possible.

In a later version of this document, this section will be worked out further based on practical experiences. The following indicates the type of issues involved.

9.7.1. Prevent Database Management Messages

By use of active management certain database messages can be prevented. Therefore, attention will be paid to the following subjects:

- Active space management for data and index storage
- Auto space management
- Redo and temporary space management
- Monitor the file system space (particularly swap space)
- Processes parameter
- Undo parameter settings
- Session cached cursors

9.7.2. Other Points of Special Interest

Additionally, attention will be paid to the following remaining related matters:

- Number of parallel processes
- Monitoring batch jobs
- Advice to application managers with regard to batches
- Error messages in batch scheduler log
- Compiling sources as a preventive measure
- Locks

10. OHI DATABASE VAULT - DESCRIPTION

Oracle Database Vault is a licensed option with the Oracle Database Enterprise Edition to implement more restrictive access to database objects and especially to implement an additional protection for the data stored in these objects.

This chapter describes the implementation of Oracle Database Vault for OHI. It is intended to help in understanding the OHI specific implementation, the choices made, and the consequences for your organization.

Once your organization has decided to implement Database Vault for OHI and has prepared the organization for the process changes that Database Vault requires, you can use the next chapter ‘Installing and Configuring Oracle Database Vault for OHI Back Office and/or OHI Data Marts’ to perform the installation and configuration. You can also find references to the standard Oracle Database Vault documentation in that chapter.

Starting with OHI release 10.20.7.0.0, OHI Back Office and OHI Data Marts support an OHI-specific implementation of Oracle Database Vault for production use.

Support for Oracle Database Vault for OHI Data Marts in an Autonomous Data Warehouse (ADW) is foreseen for a later release.

In this manual the implementation of Oracle Database Vault is described both for OHI Back Office and OHI Data Marts. Where the implementation is product-specific this will be noted.

10.1. DATABASE VAULT FOR OHI – ORGANISATIONAL IMPACT

Oracle Database Vault can be used to prevent privileged, often generic, database administration accounts (e.g. common accounts SYS and SYSTEM in the CDB\$ROOT container database or other accounts with the DBA role), from accessing or manipulating OHI data. This access may be undesirable because the OHI data is considered ‘sensitive data’.

Usually such ‘DBA’ accounts have privileges to access all data in all database objects of the database. As these more technical privileged accounts are often not tied to a single person but are used by a group of people in a database administration department, there is limited control over who can use these privileges and limited control over auditing. If the database administration is executed by an outside party, the OHI customer has even less control over who can access the OHI data with these accounts.

The OHI tables contain personally identifying data as well as sensitive health and financial information regarding these persons. Therefore, this data is classified as very privacy sensitive information, and special legal regulations apply. As such it may be that security policies and legal obligations cannot be fulfilled when such administrative accounts can fully access this sensitive data.

For these situations OHI supports the implementation of Oracle Database Vault, which means a clear segregation of duties will be supported and required:

- Database (and other technical) administrators will typically execute the day-to-day database and platform administration and management tasks and will have no access to OHI application data during these tasks.
- ‘Functional’ application administrators, who do not and should not have the privileges of database and technical administrators, will manage the access to the application data

and as such will also be responsible for account and access management. This means at least creating and dropping user accounts, granting privileges, and fulfilling unlock and password reset requests. As such only the application administrators can grant access to the application data and will know and manage the credentials of accounts that provide access to the application data. Typically, these ‘functional’ application administrators have no privileges for performing technical administrative tasks.

10.1.1. OHI Realm

Implementing Database Vault for OHI means a ‘realm’ is created in which the OHI related objects and accounts are placed that need extra protection. The realm is an extra protection zone and acts as an additional access gate. Besides the regular database privileges to access database objects, additional privileges for the realm are needed to actually access (the data of) an OHI database object.

Such a Database Vault implementation with a realm has quite some impact on database and application management tasks, due to the required segregation of duties. The following principles should be adhered to:

- The passwords of the Database Vault management accounts (‘DV owner’ and ‘DV account manager’ account, including backup accounts for these) need to be transferred to the ‘functional’ application administrators after the Database Vault implementation has been activated by the DBA’s. Typically, passwords have to be reset after these accounts have been created by the database administrators. (Technical note: this might mean the DV_OWNER role needs to be revoked temporarily, see the Database Vault administration manual, paragraph ‘Resetting the DV_OWNER User Password’).
- Only the functional application administrators create new database accounts and implement actions like password reset, unlock and lock of database accounts.
- Specifying passwords for ‘datasource’ accounts (for example for a connection pool for web services), wallet accounts, etc. must be done by the functional application administrators. Depending on which administrators implement application maintenance tasks this may mean that passwords have to be provided in ‘shared screen’ sessions, where the database or technical application administrator may execute technical configuration and the ‘functional’ application administrator needs to provide the passwords for accounts when prompted for.
- The OHI accounts that can be used through a wallet (a file which is restrictively accessible on application server level, and as such typically available for privileged operating system accounts) need extra protection. Having access to the wallet should not imply having access to the sensitive OHI data.
- Installing OHI Releases and patches will require combined actions and close cooperation from DBA’s and functional application administrators.
- Tasks that require involvement of both DBA and functional application administrator (four-eyes principle) should always be executed that way. No password handover should occur unless directly reset afterwards.
- A robust and secure auditing of all database vault related command executions is required so database vault cannot be disabled temporarily without being detected.

The implementation of Oracle Database Vault for OHI means that new and existing database accounts will not have access to the OHI Realm (and the data in it), unless extra realm privileges are granted to the account.

This means that for example the ‘almighty’ internal (common) SYS account of the database, that can normally query all the tables within all accounts in the database, is not able to access

the tables when they are additionally protected by the OHI realm. The same is true for accounts with the DBA privilege or the "SELECT ANY TABLE" privilege.

Please realize that this change in privileges will have a major impact on custom code, work processes, loading and unloading of data, ad-hoc querying, etc. Implementing Database Vault for OHI requires a project approach and is not (just) a technical exercise.

10.2. REALM COMPOSITION

The OHI realm is an additional "protection zone", additional to the regular database privileges that contains (and as such 'protects' by requiring realm authorization):

- The OHI table owner account with all its objects
- OHI BO specific: The OHI view owner account with all its objects
- OHI BO specific: The OHI Dynamic PL/SQL executing account with its objects
- All standard OHI delivered/required database roles:
 - for OHI Back Office:
OZG_ROL, OHI_ROLE_ALL, OHI_ROLE_BATCH,
OHI_ROLE_EXTRACT, OZG_ROL_DIRECT, OZG_ROL_SELECT,
OHI_REALM_ACCESS
 - For OHI Data Marts:
OBD_ROL_ADMIN, OBD_ETL_ROL, OBD_ROL_SELECT,
OHI_REALM_ACCESS

Note: the OHI_REALM_ACCESS role mentioned above is only created as part of the OHI realm implementation. When no realm is created the role is not needed.

Even an account that is protected by a realm and as such is 'contained' in the realm only has access to its own realm-protected objects when it has also received realm access.

The OHI realm grants access to the following accounts and roles:

- The OHI table owner (the account name is customer defined, typically OZG_OWNER for OHI BO and OBD_OWN for OHI DM)
- OHI BO specific: The OHI view owner (OHI_VIEW_OWNER)
- OHI BO specific: The OHI Dynamic PL/SQL executing account (OHI_DPS_USER)
- The batch account (name is customer defined, typically BATCH)
- DVSYS (this is the owner account for the Database Vault code objects). Access is necessary in order to be able to execute the OHI code that implements the access rules, for example a rule to allow OHI owner connections only when an authorizing session is present.
- Roles:
 - for OHI BO:
OHI_ROLE_ALL, OHI_ROLE_EXTRACT, OZG_ROL_DIRECT,
OZG_ROL_SELECT, OHI_REALM_ACCESS (OZG_ROL and
OHI_ROLE_BATCH do **not** provide access)
 - for OHI DM:
OBD_ROL_ADMIN, OBD_ROL_SELECT, OBD_ETL_ROL

10.3. REALM ACCESS

The previous paragraph describes what accounts (and their objects) are protected by the OHI Back Office or OHI Data Marts realm and what accounts and roles have access by default. This paragraph describes in more detail how access is arranged during usage and technical maintenance of OHI Back Office.

10.3.1. Limiting access for OHI owner accounts

The OHI accounts that own the OHI objects are in fact “schema owners”. These accounts are not required (i.e. nobody needs to log on with these accounts) during normal operation of the OHI application.

OHI has been recommending since many years to lock these accounts during normal operation of OHI, as a security measure. This must be implemented by the database administrator group. However, for a technical OHI or database administrator it is possible to (unlock and) logon to these accounts when no OHI realm for Database Vault is active.

Besides that, these accounts are required to use when the OHI objects need to be changed, e.g. during upgrades and application of OHI patches, so the accounts cannot be locked permanently.

To prevent access to these OHI owner accounts during normal operations the OHI realm contains two ‘realm owner access restrictions’:

- A database vault ‘connect’ rule that only allows logging on as an OHI object owner account (three accounts for OHI BO) when authorization is ‘active’. At other times, no connection is possible.
- A supplementary ‘session’ restriction that only allows sessions of OHI owner accounts to exist as long as the same authorization is ‘active’.

The connect and session authorization is active, meaning access is authorized, when at least one session in the same (pluggable) database is present, with an account name that starts with ‘OHI_REALM_AUTH’.

When such an account is logged on, connections by the OHI owner accounts are allowed and data access for OHI tables is possible. As soon as no such authorizing session is present anymore the last disconnecting authorizing session kills remaining OHI owner sessions.

The ‘authorizing’ accounts may only be created and used for this purpose. OHI Development strongly recommends creating ‘personal’ accounts for each person that can provide this authorization. By using personal accounts, it is always possible to track who granted access by logging on with his or her authorizing account. When e.g. Kevin Smith receives such an authorizing account it can be named OHI_REALM_AUTH_KEVIN_SMITH.

Presumably, the recipients of these authorizing accounts will be functional application administrators that already have privileged accounts for their daily work. Although it may be tempting to combine these accounts, you should never do so. Giving DBA’s access to the OHI data should always be a conscious decision, and not a by-product or unintended consequence of some other activity a functional administrator performs.

These new authorizing accounts must not receive object privileges or any database roles. They should only receive CREATE SESSION and RESTRICTED SESSION system privilege. They should also not be (!) an OHI user.

It is important that logons of these sessions are audited as these provide temporarily additional access to the sensitive OHI data!

10.3.2. Providing access for ‘interface’ accounts

Several “interface” accounts are used in OHI Back Office, e.g., for servicing web services or accessing the Advanced Queueing queues that implement the storage for Java Messaging Services (JMS) queues. These are accounts like:

- SVL_USER%
- HSL_USER%
- PSL_USER%
- OHI_JMS_QUEUE_USER

When the OHI realm is created these accounts do not have access to the realm by default. They need to be authorized. This is done by granting them the new role OHI_REALM_ACCESS.

After the new role has been created by a DBA, OHIPATCH step 120 will take care of granting this role to all database accounts that previously received direct object grants to OHI table owner objects.

10.3.3. OHI BO - Providing access for ‘custom code’ accounts

OHI customers have developed custom code that accesses the OHI software and data. The accounts that own this custom code have previously received privileges directly on the OHI objects (tables and software packages). These privileges are called “direct object grants” and are different from privileges granted via roles. These accounts now also need to receive access to the OHI realm.

As this situation is only supported for OHI Back Office, this paragraph is not relevant for OHI Data Marts.

Some OHI software packages are executed with “definer rights”, while others (the majority) are executed with “invoker rights”. This determines the privileges that the custom code accounts need.

When the custom code accounts only access the data and execute scripts by means of anonymous blocks or by custom “invoker rights” stored objects, in which only OHI PL/SQL objects with “invoker rights” are executed, it is enough to grant the OHI_REALM_ACCESS role in order to access the OHI objects protected by the realm. As with the interface accounts described in the paragraph above, this is automatically taken care of by step 120 of OHIPATCH.

When the custom code accounts also execute OHI packages with “definer rights”, which is more likely as the calling code may be stored as such, the OHI_REALM_ACCESS role is not enough. Execution will result in error “ORA-01031 (insufficient privileges)”.

It will be necessary to explicitly add participant authorization for the custom code account to the OHI realm by means of a routine `add_participant_to_ohibo_realm` in the `ALG_DB_VAULT_PCK` package (the next chapter describes this in more detail).

10.3.4. OHI BO - OHI role based custom code ‘runtime’ accounts

OHI customers also use accounts that do not own custom code and only have received privileges on OHI objects granted by pre-defined roles. These roles are OZG_ROL_DIRECT and OZG_ROL_SELECT. These roles already provide access to the realm by default, so no additional action is required for such accounts.

10.3.5. OHI BO - Regular user accounts

Regular OHI BO application users typically only have CREATE SESSION system privilege. When they log on and are identified as authorized functional application users, they dynamically receive an OHI secure application role, session based, which also provides them access to the realm.

10.4. ACCOUNT MANAGEMENT

As soon as Oracle Database Vault is configured and enabled this means the DBA accounts can no longer be used for creating and managing user accounts and granting these accounts system privileges.

Account management activities like unlocking accounts or resetting passwords are no longer possible with the DBA accounts. These responsibilities are transferred to Oracle Database Vault Account Manager accounts and additional accounts that receive this role (the DV_ACCTMGR role, see later). Of course, this means and requires a clear separation of duties which should not be combined by the same team or persons.

The impact on the organization and the work processes needs to be determined in advance. Changes in workflows and tooling will be required, e.g. for Helpdesk tools.

It is useless to implement Database Vault if the same persons will have access to the DBA accounts and the new account manager accounts.

10.5. OHI TECHNICAL APPLICATION MANAGEMENT

In most current OHI usage scenarios the DBA team also performs the OHI release installations (running 'OHIPATCH'). For this the DBA team needs access to the OHI object owner accounts, during the execution of release installation activities.

During such periods close cooperation will be needed between the DBA's, performing this work, and the persons who can authorize temporary access to the OHI realm, by logging on with their OHI_REALM_AUTH% account. Such an authorizing person needs to setup a database session during the release installation activities and needs to end it as soon as these are finished.

Note that these realm authorizers are not necessarily the same persons as the account managers mentioned above. Even if they are, they will and must use different accounts for their account management activities than for their realm authorization activities.

10.6. SUPPORTED SITUATIONS

The OHI code offers support for Oracle Database Vault in such a way that its use is completely optional and the following situations are all supported, provided the installation instructions in the next chapter are followed. The situations apply to the CDB\$ROOT container as well as the pluggable database that contains OHI.

- No Database Vault software installed (database option not present).
- Database Vault software installed but not configured.
- Database Vault software installed, configured but not enabled.
- Database Vault software installed, configured and enabled but no OHI realm in the pluggable database.
- Database Vault software installed, configured and enabled as well as an OHI realm in the pluggable database created and enabled.

The use of the Database Vault option “Operations Control” is optional (but requires extra configuration attention, ‘local accounts’ in the PDB should be used).

The use of ‘local accounts’ for Database Vault in the pluggable database (instead of using the common accounts in both container and pluggable databases) is optional (as long as “Operations Control” is not enabled).

The impact of both choices is discussed in the next paragraphs.

10.6.1. Local versus common accounts

Database Vault gives you a choice between using common accounts and local accounts for the DV configuration and management in the pluggable database.

Common accounts are created in the container database during DV configuration in the container database. They normally can be used to configure and administer Database Vault in all PDBs within the container database. It is an option though to create local accounts in a PDB for DV configuration and administering DV. Or to grant DV administration privileges to additional local accounts after configuring DV with common accounts in the PDB.

Another option during DV configuration in the container database is to enforce the use of local accounts for DV configuration and administration in the pluggable database. In such a situation the common accounts cannot be used for administration in the pluggable database and a set of local accounts in the pluggable database is required.

To use local accounts, you need to create additional accounts in each PDB and use these to configure Database Vault in the PDB.

Local accounts are stored within the pluggable database and move with the PDB when it is moved or cloned to a different container database. You can still use the same local DV accounts and passwords as before, even if the passwords for the common DV accounts are different in the other CDB.

When the use of local accounts in the container database is enforced during configuration, common accounts with DV_OWNER privileges cannot disable DV in the PDB; only local accounts with DV_OWNER privileges can do this.

When in such a situation DV is disabled in the CDB\$ROOT (by a common DV account) while it is enabled in the PDB, the PDB will only be available in RESTRICTED MODE and errors will be thrown when OHI tables are accessed.

If you plug in a PDB with local accounts into a container database where common DV accounts are used with enforced local DV accounts, the common accounts cannot disable or enable DV.

Another reason for using local DV accounts is the necessity to grant some privileges to the DV accounts in the OHI PDB. These grants might be lost when common accounts are used and a PDB is moved to a different CDB\$ROOT container with different named common DV owner accounts.

When you enforce the use of local accounts this impacts the installation of database Release Updates (RU) as you cannot grant a necessary DV_PATCH_ADMIN role only on container database level; you need to do this separately for each DV enabled pluggable database using local accounts, which might impose an unwanted technical maintenance impact.

The OHI Database Vault implementation supports both common and enforced local accounts.

10.6.2. Operations Control

Database 19c offers new Database Vault functionality named “Operations Control”. When enabled, this limits the privileges of common accounts in general for use in pluggable databases. This means that both the common DV accounts and other common accounts (like SYS and SYSTEM) are limited in their PDB access. When enabled all common accounts cannot access application data in PDB’s and cannot execute certain commands.

Just to be clear, Operations Control only prevents common DBA accounts from accessing data in a pluggable database. It does not offer any protection from local accounts in a pluggable database that have DBA privilege, or e.g. “SELECT ANY TABLE”. These local accounts can still access OHI data. Therefore, Operations Control can only be an addition to, and not be a replacement for the OHI realm.

When Operations Control is activated in a CDB and its PDB’s, this does have further impact on regular DBA tasks:

- As mentioned earlier, this influences the way the DV_PATCH_ADMIN role is granted. This role is required for a database Release Update installation. Therefore, it also needs to be granted through local accounts in each DV-enabled PDB.
- A PDB that was running in an Operations Control enabled state and is plugged out to plug it in into a different CDB also requires Operations Control to be enabled in that target CDB. Otherwise, it will remain in restricted mode due to plug-in violations.

The OHI Database Vault implementation supports configurations with and without Operations Control.

Be aware that the combination, of using DV common accounts and Operations Control, will require extra actions during OHI maintenance tasks:

- When DV actions need to be executed in the OHI PDB that require the DV root account this means only the common DV account is available. But that account has been limited by Operations Control. In those situations, Operations Control will first need to be disabled (at least for the OHI PDB) by the common DV root account in the CDB.
- After the actions in the OHI PDB have been completed, Operations Control needs to be re-enabled in the CDB.
- The common DV account manager account is also impacted: it cannot be used for local account management in the PDB. For that a local account is needed with DV account manager privileges.

Given this impact, the combination of Operations Control with only common DV accounts is not advised. When Operations Control is enabled, we strongly advise to also use local DV accounts in the PDB.

10.7. EFFECTIVE USE OF THE OHI SPECIFIC REALM

As mentioned earlier a clear segregation of duties is needed for a safe and secure implementation of restricted access to the OHI data.

Without this segregation the implementation of Database Vault and the OHI realm introduces complexity without adding much security.

The following roles need to be clearly distinguished and assigned to persons:

1. Technical Database Administration (backup/restore, availability, upgrades, release updates, monitoring, tuning, etc.)
2. Technical OHI Application Administration (OHI release installations, technology stack changes, configuration changes, space management, object management, etc.)
3. Database Vault (DV owner) and Database Account management (creating/dropping database user accounts, assigning/resetting passwords, assigning roles, unlock accounts, etc.)
4. Temporary authorizing OHI realm access for (mainly OHI) technical maintenance goals

Some roles may be combined but typically the same person should not combine roles from the first two and the last two.

Restrictive and structural password management is one of the key elements in this, passwords should never be shared over groups that execute different roles.

When Database Vault for OHI is being implemented make sure you also give attention to:

- Privileged OHI accounts versus regular functional user accounts, make sure you know what accounts have what privileges.
- Password policies.
- Auditing
- Tracking which person uses which account for technical maintenance tasks

11. INSTALLING AND CONFIGURING ORACLE DATABASE VAULT FOR OHI BACK OFFICE AND/OR OHI DATA MARTS

Oracle Database Vault is a licensed option with the Oracle Database Enterprise Edition to implement more restrictive access to database objects and especially implement an additional protection for the data stored in these objects.

Starting with OHI release 10.20.7.0.0, OHI Back Office and OHI Data Marts support an OHI-specific implementation of Oracle Database Vault for production use.

Support for Oracle Database Vault for OHI Data Marts in an Autonomous Data Warehouse (ADW) is foreseen for a later release.

This chapter describes the installation and configuration actions that are needed to implement Database Vault both for OHI Back Office and OHI Data Marts.

For some background on the actions described in this chapter, see chapter OHI Database Vault - Description.

11.1. IMPLEMENT DATABASE VAULT

This paragraph will describe how to install, configure and enable the Database Vault option for your OHI database. These three generic steps are necessary before the OHI specific Realm can be created and activated. The OHI Realm protects the OHI data against access by regular (local) DBA and other privileged accounts.

The steps need to be executed for both the root container database CDB\$ROOT and the pluggable database that contains the OHI objects.

11.1.1. Install Database Vault components

Check the [Oracle Database Documentation](https://docs.oracle.com/en/database/oracle/oracle-database/19/security.html) for the installation instructions as present in the Database Vault Administrator's Guide, which can be found through the following link:

<https://docs.oracle.com/en/database/oracle/oracle-database/19/security.html>

The database components of the Database Vault option need to be installed on both the root container as well as in the pluggable database. You need the components Oracle Label Security (OLS) and Database Vault (DV). You do not need a license for OLS when it is only used for DV.

Check whether these components are present in your root container and pluggable database(s) by querying view CDB_REGISTRY from within the CDB\$ROOT container with opened pluggable database(s):

```
select r.comp_name
,      r.status
,      r.schema
,      r.other_schemas
```

```
,      r.con_id
,      p.pdb_name
from    cdb_registry r
        left outer join
        cdb_pdbbs p
on       r.con_id = p.con_id
where    r.comp_id in ('OLS','DV')
order by
        r.con_id
,        r.comp_id;
```

A possible result looks like below, be sure you do not forget to install these components in your target PDB):

COMP_NAME	STATUS	SCHEMA	OTHER_SCHEMAS	CON_ID	PDB_NAME
Oracle Database Vault	VALID	DVSYS	DVF	1	
Oracle Label Security	VALID	LBACSYS		1	
Oracle Database Vault	VALID	DVSYS	DVF	3	OHIDEV01
Oracle Label Security	VALID	LBACSYS		3	OHIDEV01

When DV is not present in your target pluggable database or not valid, run the scripts below to create these components in the relevant database (container, OHI pluggable database or both).

The scripts to create the OLS and DV components are respectively:

`$ORACLE_HOME/rdbms/admin/catols.sql`

`$ORACLE_HOME/rdbms/admin/catmac.sql`

Before running these scripts determine the number of invalid objects to check if you need to recompile objects after running these scripts. You can use the following query for this:

```
select con_id, owner, count(1)
from    cdb_invalid_objects
group by
        con_id, owner order by 1,2
```

Run these scripts as SYS while connected to the relevant pluggable database.

After installation, the status of OLS can be queried through CDB_OLS_STATUS while the status of DV can be queried through CDB_DV_STATUS. Note that these views only exist after the components have been installed.

If you created your CDB\$ROOT container in 19c, it will probably have OLS and DV installed and you only need to install these options in the PDB.

Run both scripts in sqlplus while connected to the target database (CDB and/or PDB) and make sure you spool the result so you can investigate any errors.

In the example below, parameter 1 and 2 are explicitly undefined before starting `catmac.sql`. This is to avoid issues if a previous script or your logon script sets these parameters.

Parameter 1 specifies the default tablespace for the users that will be created (usually tablespace 'SYSAUX') and parameter 2 specifies the temporary tablespace (usually 'TEMP'). The

question mark notation is used to identify the database 19c Oracle Home, requiring your shell environment is correctly initialized.

The dialog of the catmac.sql script will look like this:

```
Enter value for 1: SYSAUX
Enter value for 2: TEMP
```

You may run the scripts with the commands as shown below:

```
spool dv_install
@?/rdbms/admin/catols.sql
undefine 1
undefine 2
@?/rdbms/admin/catmac.sql
spool off
```

Or, as non-interactive (scriptable) alternative:

```
spool dv_install
@?/rdbms/admin/catols.sql
undefine 1
undefine 2
@?/rdbms/admin/catmac.sql SYSAUX TEMP
spool off
```

Rerun the CDB_REGISTRY query above to check OLS and DB are installed and have status 'VALID'.

11.1.2. Configure Database Vault for CDB\$ROOT

When the components for Database Vault have been installed the configuration can be started. Check the [Oracle Database Documentation](#) for the registration instructions, meaning the configuration and enablement for the root container.

As part of the registration process, the Database Vault administrative accounts must be created.

These are user accounts that are granted the Database Vault roles DV_OWNER and DV_ACCTMGR. As a safety measure, Oracle recommends that you create backups of these user accounts, so you can reset the password of the primary account if it is lost.

ATTENTION: Passwords should be specified/changed by the 'future' Database Vault manager team, typically a different team than the DBA team that will manage the database, or the passwords must be handed over to that team after installation and immediately changed by them.

The accounts must be created in the root container by creating common accounts. You may choose your own usernames for the Database Vault administrative accounts. You can use the example commands below.

```
GRANT CREATE SESSION, SET CONTAINER TO c##dbv_owner_root
IDENTIFIED BY &password_root CONTAINER = ALL;
```

```
GRANT CREATE SESSION, SET CONTAINER TO c##dbv_owner_root_backup
```

```
IDENTIFIED BY &password_root_bkp CONTAINER = ALL;

GRANT CREATE SESSION, SET CONTAINER
TO c##dbv_acctmgr_root
IDENTIFIED BY &password_am_root CONTAINER = ALL;

GRANT CREATE SESSION, SET CONTAINER
TO c##dbv_acctmgr_root_backup
IDENTIFIED BY &password_am_root_bkp CONTAINER = ALL;
```

After these accounts have been created, configure the backup Database Vault users using the command below.

Choose between using local or common accounts for the Database Vault administration:

- `force_local_dvowner=false`: the common accounts you create for Database Vault in the container database CDB\$ROOT can also be used in the PDB's.
- `force_local_dvowner = true`: the common accounts you create for Database Vault in the container database CDB\$ROOT cannot be used in the PDB's. They have no DV_OWNER or DV_ACCTMGR privileges in the PDB(s). You will need to create local accounts in the PDB and use them to configure Database Vault in the PDBs, in the next paragraph.

Using local accounts is optional, see the previous chapter for the additional impact this has on some administrative tasks. In the instructions that follow it is assumed common accounts are used for administering DV also in the PDB's.

Execute this command while connected to the CDB as SYS:

```
BEGIN
  CONFIGURE_DV
  ( dvowner_uname      => 'c##dbv_owner_root_backup'
  , dvacctmgr_uname    => 'c##dbv_acctmgr_root_backup'
  , force_local_dvowner => false
  );
END;
/
```

If you run into the error below, close the pluggable database (shown below as <pdb_name>) and re-execute the command:

```
ORA-65048: error encountered when processing the current DDL
statement in pluggable database <pdb_name>
.....
ORA-65092: system privilege granted with a different scope to
'DBA'
.....
```

The roles DV_OWNER and DV_ACCTMGR must be granted manually to both primary users:

```
grant DV_OWNER to c##dbv_owner_root with admin option;
grant DV_ACCTMGR to c##dbv_acctmgr_root with admin option;
```

Recompile invalidated objects in the root container:

```
@?/rdbms/admin/utlrp
```

The next query:

```
select * from cdb_dv_status;
```

should return 'true' for the record with name 'DV_CONFIGURE_STATUS':

NAME	STATUS	CON_ID
-----	-----	-----
DV_APP_PROTECTION	NOT CONFIGURED	1
DV_CONFIGURE_STATUS	TRUE	1
DV_ENABLE_STATUS	FALSE	1

11.1.3. Enable Database Vault in CDB\$ROOT

Next we need to enable Database Vault in the root container. Connect as user `c##dbv_owner_root` and execute the following command to enable Database Vault:

```
exec dbms_macadm.enable_dv;
```

Continue with a restart of the root container database.

After this the view `CDB_DV_STATUS` should also show 'TRUE' as status for the `DV_ENABLE_STATUS` record for the root container.

The query:

```
select * from cdb_dv_status;
```

should return:

NAME	STATUS	CON_ID
-----	-----	-----
DV_APP_PROTECTION	NOT CONFIGURED	1
DV_CONFIGURE_STATUS	TRUE	1
DV_ENABLE_STATUS	TRUE	1

11.1.4. Configure Database Vault in the PDB

The view `CDB_DV_STATUS` will show Database Vault is not yet configured and enabled in the pluggable database. Ignore the record for `DV_APP_PROTECTION` for now. This concerns Operations Control and will be discussed later.

The output of 'select * from cdb_dv_status' while connected as SYS in the PDB will look like this (your `CON_ID` may differ, of course):

NAME	STATUS	CON_ID
-----	-----	-----
DV_APP_PROTECTION	NOT CONFIGURED	5
DV_CONFIGURE_STATUS	FALSE	5
DV_ENABLE_STATUS	FALSE	5

11.1.4.1. Create local accounts

If you chose to use local DV accounts in the previous paragraph, you first need to create those accounts in the PDB. You are free to choose the names of these local accounts. The commands below are examples. We will use these names for the local accounts in the remainder of this document.

Connect to the OHI PDB as SYS and run the example commands below:

```
GRANT CREATE SESSION TO dbv_owner_root
IDENTIFIED BY &password_root;

GRANT CREATE SESSION TO dbv_owner_root_backup
IDENTIFIED BY &password_root_bkp;

GRANT CREATE SESSION TO dbv_acctmgr_root
IDENTIFIED BY &password_am_root;

GRANT CREATE SESSION TO dbv_acctmgr_root_backup
IDENTIFIED BY &password_am_root_bkp;
```

11.1.4.2. Configure DV

We now have the accounts we need, and can use either the common or the local backup accounts to configure DV in the PDB.

Connect to the OHI PDB as SYS and run the command below. Beware that the square brackets should not be taken literal, they stand for the optionality of the c## prefix when you have created local DV accounts and are used in the rest of this chapter:

```
BEGIN
  CONFIGURE_DV
    ( dvowner_uname    => '[c##]dbv_owner_root_backup'
    , dvacctmgr_uname => '[c##]dbv_acctmgr_root_backup'
    );
END;
/
```

This will invalidate quite some objects (we experienced 40+) within the SYS schema and some within other schemas. Recompile invalidated objects in the pluggable database:

```
@?/rdbms/admin/utlrp
```

When this fails (we have experienced this) first execute this in the CDB\$ROOT and afterwards reconnect to the PDB and repeat the action.

The roles DV_OWNER and DV_ACCTMGR must be granted manually to both primary users:

```
grant DV_OWNER to [c##]dbv_owner_root with admin option;
grant DV_ACCTMGR to [c##]dbv_acctmgr_root with admin option;
```

After this action querying the view DBA_DV_STATUS (or CDB_DV_STATUS) in the PDB

should return:

NAME	STATUS
-----	-----
DV_APP_PROTECTION	NOT CONFIGURED
DV_CONFIGURE_STATUS	TRUE
DV_ENABLE_STATUS	FALSE

11.1.5. Enable Database Vault in the PDB

To conclude we need to enable Database Vault also for the pluggable database.

Only do this when you actually are going to implement the OHI Realm for OHI Back Office or OHI Data Marts in the next step.

Before you enable the Database Vault option in the pluggable database where OHI resides, you need to grant some object and role privileges to the OHI table owner to facilitate some checks within OHI.

These privileges can only be granted as long as Database Vault is not enabled within the pluggable database!

Grant the privileges below using an account like **SYS, connected to the pluggable database**, and specify the OHI Back Office or OHI Data Marts table owner (whichever is applicable):

```
-- set account once
-- copy+paste+run the line below separately
define ohi_table_owner = &my_ohi_table_owner

-- now copy+paste+run the lines below
-- these are needed to be able to query the DV configuration
grant select on DVSYS.DBA_DV_REALM to &ohi_table_owner;
grant select on DVSYS.DBA_DV_REALM_AUTH to &ohi_table_owner;
grant select on DVSYS.DBA_DV_PROXY_AUTH to &ohi_table_owner;
grant select on DVSYS.DBA_DV_DATAPUMP_AUTH to &ohi_table_owner;
```

For OHI Back Office only:

connect to the DV owner root account in the pluggable database and grant proxy authorization privileges for the OHI table owner (in most cases 'OZG_OWNER') to use the two other OHI Back Office schema accounts:

```
begin
    dbms_macadm.authorize_proxy_user('&ohi_table_owner','OHI_VIEW_OWNER');
    dbms_macadm.authorize_proxy_user('&ohi_table_owner','OHI_DPS_USER');
end;
/
```

These are needed for the period when Database Vault is enabled but the OHI Back Office realm has not yet been created, to prevent "ORA-01031" errors during specific OHIPATCH steps. When you immediately proceed with creating the realm you can skip these proxy authorization commands because they are also added during the OHI BO realm creation.

connect to the DV owner root account in the pluggable database and grant system privileges for the OHI table owner (in most cases 'OZG_OWNER').

```
begin
  dbms_macadm.add_auth_to_realm
  ( realm_name      => 'Oracle System Privilege and Role Management Realm'
  , grantee        => '&ohi_table_owner'
  , rule_set_name   => null
  , auth_options    => then dbms_macutl.g_realm_auth_owner
  , auth_scope      => dbms_macutl.g_scope_local
  );
end;
/
```

These privileges are needed for the period when Database Vault is enabled but the OHI Back Office realm has not yet been created, to prevent “ORA-47410: Realm violation” errors during the creation of System Views. When you immediately proceed with creating the realm you can skip this command because it is also executed during the OHI BO realm creation.

Connect to the DV owner root account in the pluggable database and execute:

```
exec dbms_macadm.enable_dv;
```

Next close and reopen the pluggable database.

After this action querying the view DBA_DV_STATUS (or CDB_DV_STATUS) in the PDB should return

NAME	STATUS
-----	-----
DV_APP_PROTECTION	NOT CONFIGURED
DV_CONFIGURE_STATUS	TRUE
DV_ENABLE_STATUS	TRUE

11.2. ACTIVATE THE OHI REALM

This paragraph describes the installation of the OHI realm. The OHI Realm uses the Database Vault functionality that was configured and enabled in the previous paragraphs.

Make sure there are no OHI related sessions connected to the database while you execute the actions in this paragraph. Running sessions are affected by the realm actions and will throw errors. All sessions need to reconnect after the realm creation actions have been performed. For OHI Back Office this includes the sessions used for executing the web services pl/sql and the access of the AQ queues for offering the JMS Queues in WebLogic.

11.2.1. Create OHI realm access role

Note: This must be done before the OHI realm is created!

Connect to the OHI PDB as SYS or a (local) DBA, and execute:

```
create role ohi_realm_access;
grant ohi_realm_access to &ohi_object_owner with admin option;
```

This role is necessary to provide access to the OHI Realm to other users. These are typically interface supporting users (connection pool users for WS's and queue access) and custom

development users (especially in the OHI BO environment) that are not granted access by means of an OHI role. See OHI BO - Providing access for 'custom code' accounts.

Execute privileges for a few OHI objects need to be granted to Database Vault related accounts. After this, the role OHI_REALM_ACCESS needs to be revoked from the DBA account that was used to create it. In the code below, we will assume SYS was used and common DV accounts are used.

Connect as the OHI object owner and execute the following commands:

```
grant execute on alg_db_vault_pck to [c##]dbv_owner_root;
grant execute on alg_dv_allow_owner_access_fnc to dvsys;
grant ohi_realm_access to [c##]dbv_owner_root;

revoke ohi_realm_access from sys;
```

11.2.2. Create users that authorize OHI owner realm access

As soon as the OHI Realm is created in the next paragraph, the OHI owner account will no longer be able to log on, unless another user with a specific account (with a name starting with OHI_REALM_AUTH) is logged on to authorize the OHI owner session.

See Limiting access for OHI owner accounts for more background.

Create at least one such account, according to the example below.

Connect to the OHI PDB as [c##]dbv_acctmgr_root, and execute a command like:

```
grant create session to ohi_realm_auth_kevin_smith
identified by &somepassword;
```

This OHI_REALM_AUTH% account should not receive any other database role or system privilege and should not (!) be registered as an OHI user account ('functionaris'). OHI performs checks on this.

11.2.3. Grant inherit privilege and create the OHI realm

Beware, after you have created the OHI specific realm using the instructions in this paragraph it is no longer possible to connect to the OHI BO or OHI DM object owners in the usual way. For OHI BO this also applies to the OHI_VIEW_OWNER and the OHI_DPS_USER accounts. This means OHIPATCH.pl can only be used when an additional session is active for an OHI_REALM_AUTH% account, as created in the previous paragraph.

Connect to the OHI PDB as the Database Vault root owner [c##]dbv_owner_root and execute the following command:

```
grant inherit privileges on user [c##]dbv_owner_root to
&ohi_object_owner;
```

This allows the OHI-specific procedure alg_db_vault_pck.cre_ohi_realm to create the OHI Realm using DV functionality.

If the execution of this command results in an ORA-47506 error stating Database Vault is not activated in the CDB\$ROOT, while it is, you are running into database bug 30255143. The work-around is to disable DV in the PDB, restart the PDB, execute the grant, enable DV and restart the PDB. In short:

- As DV owner root in the OHI PDB:
`exec dbms_macadm.disable_dv;`
- As DBA in the OHI PDB:
`alter pluggable database close immediate;`
`alter pluggable database open;`
- As DV owner root in the OHI PDB:
`grant inherit privileges on user [c##]dbv_owner_root to &ohi_object_owner;`
`exec dbms_macadm.enable_dv;`
- As DBA in the OHI PDB:
`alter pluggable database close immediate;`
`alter pluggable database open;`

Connect to the OHI PDB as the Database Vault root owner **[c##]dbv_owner_root** and execute the following command to create the OHI realm:

```
set serveroutput on
exec alg_db_vault_pck.cre_ohi_realm;
```

When the last call has finished the OHI specific Database Vault realm with name ‘OHI Back Office Data Protection’ or ‘OHI Data Marts Data Protection’ has been created. You can check the presence by querying the view DBA_DV_REALM.

Again: from this moment on it is no longer possible to connect to the standard OHI BO or OHI DM schema’s without a session of a OHI_REALM_AUTH% account.

When the realm creation fails for some reason you can delete the realm, resolve the problem and execute the creation again.

To delete the realm, connect to the OHI PDB as the DV owner root and execute:

```
set serveroutput on
exec alg_db_vault_pck.del_ohi_realm;
```

If this does not succeed please read the paragraphs about updating and deleting the realm for what you might be able to do about this.

11.2.4. OHI BO - Grant additional realm access

This paragraph only applies to OHI Back Office.

See Providing access for ‘interface’ accounts and OHI BO - Providing access for ‘custom code’ accounts for background information.

After the realm has been created successfully, make sure you now grant access to the accounts that need it.

The interface accounts (e.g. the accounts that are used in the connection pools for the web services) get realm access in this way:

Connect as the OHI owner to the OHI PDB and execute the command below.

NOTE: This requires a session of a OHI_REALM_AUTH% account.

```
set serveroutput on
exec alg_security_pck.grant_realm_access_role;
```

The custom code owners that have direct object privileges and contain definer rights objects should get realm access in this way:

Connect to the OHI PDB as the Database Vault root owner **[c##]dbv_owner_root** and execute the following command for each of the custom code owners:

```
exec alg_db_vault_pck.add_participant_to_ohibo_realm(pi_grantee
=> '&custom_code_account');
```

When you need to revoke the authorization a 'remove' routine can be used:

```
exec alg_db_vault_pck.rem_participant_from_ohibo_realm
(pi_grantee => '&custom_code_account');
```

NOTE: When a new realm definition is applied, by deleting the old definition and creating the new realm definition, a manual action to be undertaken by means of a Database Vault owner account (having the DV_OWNER role), the already manually added participants are lost and need to be added again. Before deleting a realm you can query the existing participant through view DBA_DV_REALM_AUTH.

11.2.5. Enable Operations Control

Database 19c offers new Database Vault functionality named Operations Control. See Operations Control in this document for more detail.

When Operations Control is enabled, this limits the privileges of common DV accounts and generic common accounts like SYS and SYSTEM in the PDB's. The common accounts are limited in their access to data in the PDB and cannot execute certain commands.

The OHI Realm does not require Operations Control and will not be affected by Operations Control.

If you decide to enable Operations Control, it is easiest to do this after the OHI Realm has been enabled.

The instructions described earlier in this chapter assume Operations Control is not (!) yet activated. If Operations Control has been enabled, some of the previous instructions may fail with error message ORA-01031, especially if common DV accounts are used. Operations Control will have to be disabled temporarily in those cases.

To activate Operations Control in the CDB connect as the (common) DV owner root account and execute this command in SQL*Plus:

```
exec dbms_macadm.enable_app_protection;
```

To disable it in all containers in the CDB, execute:

```
exec dbms_macadm.disable_app_protection;
```

Operations Control can be disabled and re-enabled for a specific PDB, but only when it is active at CDB level, and only from within the CDB, by an account that has been granted the DV_OWNER role in the CDB:

```
exec dbms_macadm.disable_app_protection(pdb_name => '<OHI PDB>');
```

```
exec dbms_macadm.enable_app_protection(pdb_name => '<OHI PDB>');
```

You would typically need this when you would like to use a common account like SYS to grant additional object privileges in the OHI PDB to the OHI object owner.

Whether Operations Control is active or not can be queried through view DBA_DV_STATUS or CDB_DV_STATUS. The record for 'DV_APP_PROTECTION' specifies whether it is NOT CONFIGURED, ENABLED or DISABLED.

11.3. UPDATING, DEACTIVATING AND REMOVING THE OHI SPECIFIC REALM

11.3.1. Updating the OHI realm

In future OHI releases, the definition of the realm may be changed and the existing realm may need to be adapted. These actions will be part of the Specific Installation Instructions (step 20) of OHIPATCH. Each adaptation of the OHI Realm will result in a new version number of the Realm. This can be found in the Realm description:

```
select description
from   dba_dv_realm
where  name like 'OHI%';
```

will return e.g.

```
"Realm to protect all OHI data Version 7"
```

OHIPATCH will check the actual version of the created OHI realm (if present) against the expected version of that release.

To do a manual update of the OHI realm, simply delete it and then create it again.

NOTE 1: When you run into an error during the delete of the existing realm definition this may be caused by an incompatibility of the old realm definition and the newer version of the realm application code. In most cases this can be solved by temporarily disabling Database Vault in the pluggable database, restarting the (pluggable) database, deleting the realm definition as described above, enabling Database Vault again and restarting the database. After this you can create the realm again using the new application code.

NOTE 2: As mentioned in the paragraph about granting additional realm access (paragraph 11.2.4) you need to add again the participants you added previously to the realm, be sure you know them before deleting the realm (they are also reported by the 'object check' as run by OHIPATCH or can be queried through DBA_DV_REALM_AUTH).

To proceed, connect to the OHI PDB as the DV owner root account and execute these commands:

```
set serveroutput on
exec alg_db_vault_pck.del_ohi_realm;
exec alg_db_vault_pck.cre_ohi_realm;
```

This will also update the version number in the realm description.

Operations Control (discussed earlier) can remain enabled during this action if local DV accounts are used. It needs to be disabled if common DV accounts are used.

To check the expected and the actual version of the OHI realm, connect to the OHI PDB as the OHI table owner and execute:

```
select alg_db_vault_pck.get_expected_realm_version
,      sys_install_pck.get_current_realm_version
from    dual;
```

Alternatively, you can use the Info screen of the OHI BO Forms application.

11.3.2. Deactivating the OHI realm

It is technically possible to disable a realm. Whether a realm is enabled is shown in the column DBA_DV_REALM.ENABLED. For the OHI realm we do not support situations with a disabled realm, as this results in a hybrid protection with unpredictable behaviour.

When you do want to deactivate the realm you should delete it, using the instructions from the next paragraph. Deleting and creating a realm is a routine that takes a very limited amount of time, certainly less than a minute.

11.3.3. Deleting the OHI realm

When the realm needs to be removed for some reason the first step is to connect as the DV owner root account in the OHI PDB and execute these commands in SQL*Plus:

```
set serveroutput on
exec alg_db_vault_pck.del_ohi_realm;
```

If your intention is to stop using the OHI realm completely (and maybe start using it again at a later stage) more configuration actions should be reversed.

If you are also planning to disable Database Vault itself you should remove proxy authorizations and the extra system privileges granted to the OHI application owner. This can be done by calling the delete realm procedure and passing 'true' as parameter value. This will also remove the proxy authorizations and extra system privileges which were created during the preparation phase in paragraph 11.1.5:

```
set serveroutput on
exec alg_db_vault_pck.del_ohi_realm(pi_delete_proxy_auth => true);
```

In order to be able to follow the instructions again from the previous paragraph Activate the OHI Realm you need to do the following:

Connect as the OHI table owner in the OHI PDB. This will now be possible without a OHI_REALM_AUTH% session. Execute:

```
drop role ohi_realm_access;
```

```
revoke execute on alg_db_vault_pck from [c##]dbv_owner_root;
revoke execute on alg_dv_allow_owner_access_fnc from dvsys;
```

Beware: the last command will fail when you have enabled DV Operations Control, because DVSYS is a common account.

Next, connect as the DV owner root account in the OHI PDB and execute:

```
revoke inherit privileges on user [c##]dbv_owner_root from
&ohi_object_owner;
```

You should also consider dropping the OHI_REALM_AUTH% accounts.

If you intend to run OHI without the OHI Relam, but with Database Vault enabled, you need to reapply the privilege that was removed with the OHI Realm:

connect to the DV owner root account in the pluggable database and grant system privileges for the OHI table owner (in most cases 'OZG_OWNER').

```
begin
  dbms_macadm.add_auth_to_realm
  ( realm_name      => 'Oracle System Privilege and Role Management Realm'
  , grantee         => '&ohi_table_owner'
  , rule_set_name   => null
  , auth_options    => then dbms_macutl.g_realm_auth_owner
  , auth_scope      => dbms_macutl.g_scope_local
  );
end;
/
```

11.3.4. Disabling Database Vault in the PDB

To revert the Database Vault configuration in the PDB and disable Database Vault, execute the step in this paragraph. This will revert all actions described in Enable Database Vault in the PDB.

Note that it is not possible to “unconfigure” Database Vault in the PDB or in the CDB.

Connect to the OHI PDB as the DV root owner, and execute:

```
exec dbms_macadm.disable_dv;
```

Restart the PDB.

For OHI Back Office, additionally connect to the OHI PDB as the DV root owner, and execute:

```
begin
  dbms_macadm.unauthorize_proxy_user('&ohi_table_owner','OHI_VIEW_OWNER');
  dbms_macadm.unauthorize_proxy_user('&ohi_table_owner','OHI_DPS_USER');
end;
/
```

Connect as SYS or a DBA to the OHI PDB and execute:

```
revoke select on DVSYS.DBA_DV_REALM from &ohi_table_owner;
revoke select on DVSYS.DBA_DV_REALM_AUTH from &ohi_table_owner;
revoke select on DVSYS.DBA_DV_PROXY_AUTH from &ohi_table_owner;
```

```
revoke select on DVSYS.DBA_DV_DATAPUMP_AUTH from &ohi_table_owner;
```

11.4. DATABASE VAULT AND DB MAINTENANCE TASKS

In this chapter some database maintenance tasks are discussed that are impacted by the use of the Database Vault functionality.

11.4.1. Installing database Release Updates

The installation of quarterly database “Release Updates” temporarily requires more access to the database than for normal operations. The DV_PATCH_ADMIN role exists for this purpose.

You should temporarily grant this role to SYS in the CDB and PDB, by means of the DV owner account, in order to apply the RU by means of datapatch. Be sure to revoke it directly afterwards.

Please consult the DV manual for further information regarding the use of the DV_PATCH_ADMIN role when you will implement database Release Updates with DV.

11.4.2. Using DataPump for exporting data

When Database Vault and the OHI realm are enabled, exporting OHI objects with datapump is not possible by default, regardless of the account used (SYS, SYSTEM or <OHI Table Owner>).

If you try, you will encounter errors like:

```
ORA-39006: internal error
ORA-31632: master table "OZG_OWNER.SYS_EXPORT_TABLE_01" not found, invalid, or inaccessible
ORA-01031: insufficient privileges
ORA-39097: Data Pump job encountered unexpected error -1031
```

To allow datapump export, you need to grant temporary access to the relevant account.

As the DV owner root in the OHI PDB, execute:

```
EXEC DBMS_MACADM.AUTHORIZE_DATAPUMP_USER('<OHI Table Owner>');
```

After the export has completed, revoke the privilege again.

As the DV owner root in the OHI PDB, execute:

```
EXEC DBMS_MACADM.UNAUTHORIZE_DATAPUMP_USER('<OHI Table Owner>');
```

Of course, to export with the '<OHI Table Owner>', you need a OHI_REALM_AUTH% user to be logged in.

Operations Control, discussed in the next paragraph, does not impose extra restrictions on datapump export actions.

12. UNINSTALLING OHI BACK OFFICE

The de-installation of OHI Back Office consists of the following activities:

- Remove the OHI Back Office pluggable database(s) and potentially also the containing container database;
- Uninstalling the Oracle Database and Application Server software on the Database, Middleware and Client Tiers (if this Oracle system software is not used by other software any longer);
- Deleting directory `$OZG_ROOT`;
- Deleting the OS account `batch`;
- Deleting the OS account `oracle` (if this is no longer used by other software);
- Cleaning the file system.

APPENDIX A – CONFIGURATION OF MULTIPLE BATCH SCHEDULERS

Multiple batch schedulers can be launched for the same OHI (database) environment, provided they run on different application servers. For each OHI database, no more than one batch scheduler can be active per application server. If for example four applications servers are available, then a maximum of 4 batch scheduler processes can be started for the same database, one on each server.

The work between the different batch schedulers is not divided. This means that the first batch scheduler will be the active and subsequently started schedulers will only become active if the first scheduler crashes. The additional schedulers act as ‘standby’ schedulers to take over the work only when needed. As the load on the application server of the OHI batch work is quite limited this is a pragmatic way of offering higher availability.

In case of multiple batch schedulers, it is important that the output and log information is written to a central location, so that when consulting the log/output from the OHI Back Office application it does not matter on which application server the batch process has run. For this purpose, a SAN can be used. An alternative is the use of an NFS share.

Consult your Network Department or Linux Administrators to set up one or more shared directories between the Application Servers and (optionally) between the Application Servers and the Database Server.

STARTING / STOPPING THE BATCH SCHEDULER

Starting the Batch Scheduler

On each Application Server only one batch scheduler can be active for a given database. If the batch scheduler process is activated, the batch scheduler reserves a user lock for the given Application Server. If a batch scheduler process is active on the given server, the user lock is already occupied, and the newly requested batch scheduler process will abort starting.

If successful, the batch scheduler registers as an active process. This means that in table `ALG_BATCHSCHEDULERS` flag `IND_ACTIEF` of the row matching the application server is set to J.

In the following example the batch schedulers are active on servers `myhost1` and `myhost2`.

The query below...

```
select server
,      ind_actief
,      pid
from   alg_batchschedulers
/
```

...provides in that case the following result:

SERVER	I	PID
myhost1	J	17015
myhost2	J	11098

Stopping the Batch Scheduler

The active batch scheduler checks the row matching the process in table `ALG_BATCHSCHEDULERS` periodically. If column `IND_ACTIEF` is set to N, then the batch scheduler will stop.

The active batch scheduler is stopped on server `myhost` by means of the following SQL statement:

```
update alg_batchschedulers
set    ind_actief = 'N'
where  server     = 'myhost'
/
```

Script `OZG_STOP_BATCH.sh` notifies *all* batch scheduler processes of a certain environment to stop, by performing the following statement:

```
update alg_batchschedulers
set    ind_actief = 'N'
where  ind_actief != 'N'
/
```

PROCESS LOAD BALANCING MECHANISM

Functionality

The active batch scheduler process uses a process load balancing mechanism in order to determine how many new script requests for main processes and subprocesses can be processed.

For this reason, the following information is required:

- The maximum number of (main- and sub) processes that may be started (as configured in system parameter `BATCH_MAX_PROCESSEN`, `#batch_max_processes`).
- At least a quarter of this total capacity is 'reserved' for 'main' processes (rounded upwards to the first whole number, `#batch_max_main`).
- The number of active main processes started by the batch scheduler (`#active_main_processes`)
- The number of active subprocesses started by the batch scheduler process (`#active_sub_processes`)

The mechanism returns the available number of process 'slots' for main- and total nr of processes.

- $\text{Main} = \text{\#batch_max_main} - \text{\#active_main_processes}$
- $\text{Total} = \text{\#batch_max_processes} - (\text{\#active_main_processes} + \text{\#active_sub_processes})$

Example

Imagine the following situation:

- The value of `BATCH_MAX_PROCESSEN` is 100
- 10 main processes running, 70 sub processes active (total 80 running)
- 20 main requests and 30 sub request want to start

...then we can see the following nr of processes can be started.

Main capacity available: $25 - 10 \rightarrow 15$

Total capacity available: $100 - 80 \rightarrow 20$

Depending on the request order of the requests at most 15 main and 5 sub requests will be started or 0 main requests and 20 sub requests. Reason for limiting the ‘main process’ capacity is that the sub processes normally do most of the work and for these always capacity should be available to prevent the batch scheduler starts 100 main processes and there are no ‘slots’ for doing the actual work, meaning the batch work will actually come to a halt.

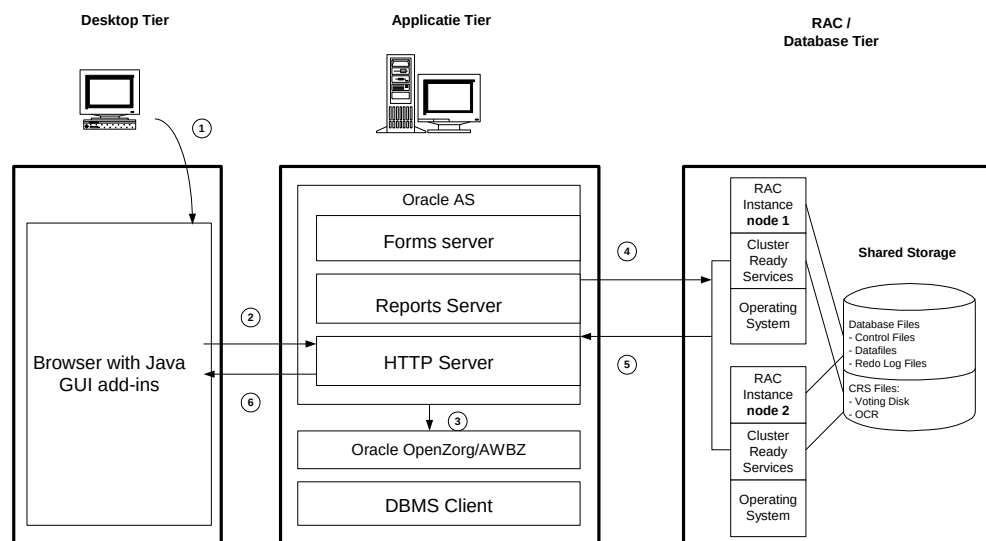
APPENDIX B - INSTALLATION & CONFIGURATION OF OHI IN A RAC ENVIRONMENT

INTRODUCTION

This chapter describes some specific points of attention for the installation and configuration of OHI Back Office in an Oracle RAC environment.

ARCHITECTURE

The following provides a general overview of the architecture.



Some remarks about the architecture:

- Deploying RAC alone is **not** a complete High Availability solution. If e.g. the storage breaks down, RAC does **not** offer protection and using e.g. DataGuard and a standby installation could provide a solution.

For an overview of the Oracle Maximum Availability Architecture see documentation that can be found through the link below:

<http://www.oracle.com/technetwork/database/features/availability/oracle-database-maa-best-practices-155386.html>

BATCH PROCESSING

It is best practice to run the batch processing on a different database instance/node from where the Forms sessions and web services run, to avoid performance degradation of the Forms and web services processes.

In an Oracle RAC environment, it is possible to run the batch processing across multiple nodes. This applies both to running separate batch types on different nodes and to running subprocesses of a single batch run on several nodes at the same time.

Some OHI Back Office batches run best when all subprocesses run on the same node. This has to do with unnecessary 'block pings' between multiple instances.

Other OHI Back Office batches can be run with their subprocesses spread across multiple nodes to improve throughput. These are the batches that do a lot of data access (reads) and few changes (writes). Examples are the claims processing batches.

You can use database services to influence on which node(s) the batches run. The desired service can be registered (as default) with the batch definition and overruled for each individual run.

FAILOVER

If all conditions for RAC Installation and Configuration are met, the failover test is the last step that must be performed.

Be aware that OHI Back Office does not support a full Transparent Application Failover for Forms sessions and batch processing.

This has to do with the fact that the PL/SQL variables and package states used by Oracle Forms are lost in case of instance failure.

Therefore, OHI Back Office will provide the following messages in case of an instance failure:

FRM-21011 PL/SQL Unhandled Exception Program Error

ORA-03113: End of File on Communication Channel

FRM-40655 SQL error forced rollback : clear form and re-enter transaction.

This is expected behavior. The Forms user will have to start a new connection with an active instance. Batches will have to be restarted.



OHI Back Office recommends **not** configuring Transparent Application Failover (TAF) for Forms considering that TAF offers no added value in a Forms environment.

TAF can be used in the Data sources c.q. Connection Pools of the Web Services because they are stateless.

MANAGEMENT

Backup and Recovery

See [Backup & Recovery/RAC](#).

Startup and Shutdown

See [Startup & Shutdown/RAC](#).

Networking

See [Networking/RAC](#).

Performance Tuning and Monitoring

See [Performance Tuning & Monitoring/RAC](#).

APPENDIX C - INSTALLATION & CONFIGURATION OF SSO IN OHI BACK OFFICE

INTRODUCTION

This chapter describes the installation and configuration of Single Sign-On (SSO) for OHI Back Office and applies only to customers who would like to enable Single Sign-On authentication for the OHI Back Office Forms and OHI JET application.

Authorization, meaning assigning roles or other privileges, is NOT part of the functionality described here. Authorization still needs to be applied within the OHI Back Office application using the OHI Back Office screens: assign application roles, access privileges for brands or financial units, etc.

As of release 10.21.7.0.0 OHI Back Office is certified (again) to work with SSO. This Appendix has been updated to use the functionality of Oracle Access Manager (OAM) to implement SSO.

Oracle Access Manager can be used for “simple” SSO scenarios and does not require Oracle Internet Directory (OID). Instead, it connects to your company LDAP server and stores its own data in OPSS (Oracle Platform Security Services), in a database schema created by RCU (Repository Configuration Utility). This is the architecture that is described in this chapter.

It is possible to use OID instead of the direct connection and OPSS, but that is beyond the scope of this chapter. More information can be found in the Forms documentation and on My Oracle Support.

Note:



Support for SSO authentication in the OHI JET application is available from release 10.22.8.0.0.



Attention: From release 10.24.3.0.0 on, two versions of SSO are supported:

- Connections to personal database accounts per user, using a proxy account. This will be referred to as proxySSO in this chapter.
- Connections using a single shared database account without ‘personal’ schemas being present in the OHI BO database. This will be referred to as commonSSO in this chapter.

Both versions can be used in parallel, and both do not necessarily conflict with the classical local logon method. This may be useful during a transition period, or as a fallback.



Attention: The Single Sign-On implementation described here only covers central authentication (is the person who he/she says he/she is) based on the centrally managed Single Sign-On account of the user. What other access and authorization is provided by that same account (besides OHI) is organization specific and determines the effectiveness of the actual single sign-on behavior. Main goal is to re-use the centrally managed account definition ('enterprise accounts') with its associated authentication password (or other form of authentication, which may include Two Factor Authentication).

After implementing this, you should be able to centrally disable or enable an account for access to an OHI environment, provided you have disabled the "classical" local logon using the Forms dialog.

Additional authorization (e.g. is the user allowed to access the OHI Back Office Application in the requested OHI environment?) and authorization (what privileges does the user have in OHI Back Office?) is outside the scope of this chapter. Activities like manual or automated provisioning of privileges and new accounts, locking or removing local database accounts (for proxySSO), actions after cloning the database to a non-production environment (changing privileges in OHI, adding test accounts for proxySSO, etc.) all need to be executed in the same way as when you are not using SSO.



Attention: The description in this chapter is for installing and configuring the minimally required components for providing SSO functionality. This is not a complete SSO solution. Additional directory services (e.g. account provisioning) should be provided by additional products, such as Oracle Identity Manager. As a customer is free in his/her choice of a directory product that offers the additional services integration, the implementation of such services will be customer specific.



Attention: The description in this chapter is not sufficient for use in a production environment. For a production environment you need to address aspects like high availability of OAM and of the connection to your Active Directory services, and the security of the communication between WebGate and OAM, and between OAM and your LDAP server.



It is possible to use SSO for only one of several Forms applications running on the same server, but the OAM Managed Servers need to be running for the non-SSO Forms applications too. All requests to OHS are sent to OAM by WebGate for evaluation.



The Single Sign-On configuration described here can only be used for end users of the OHI BO Forms and the OHI JET application. It can NOT be used for:

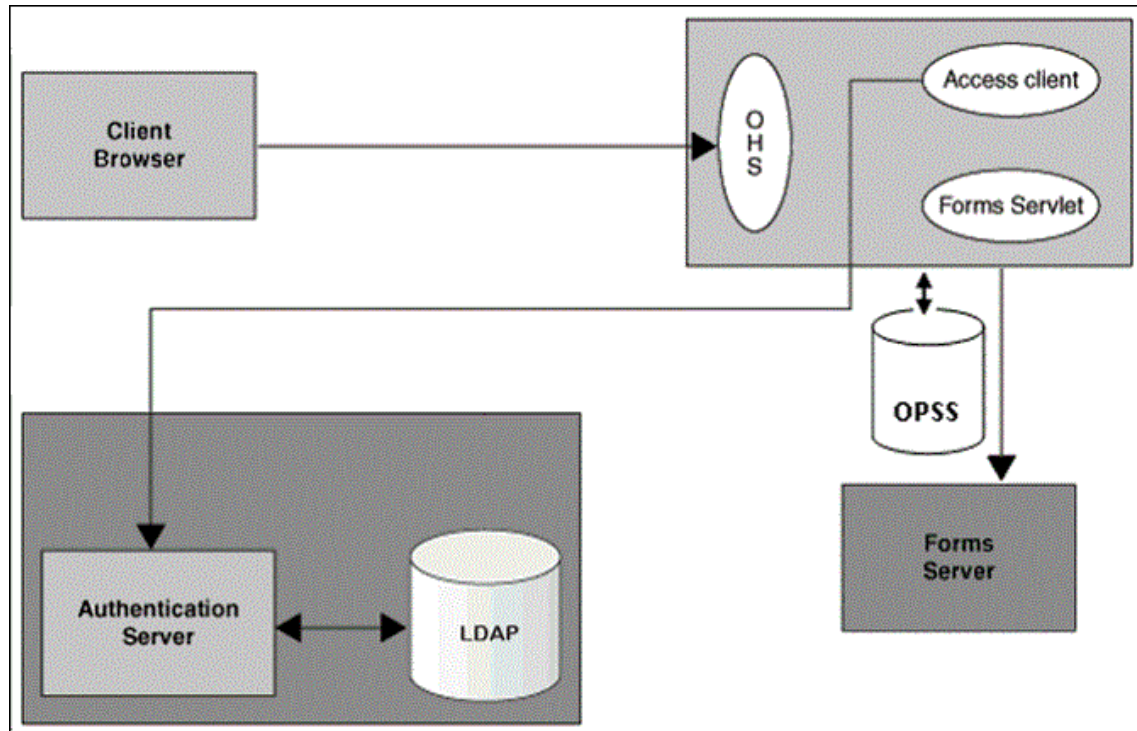
- OHI schema owners and the OHI BATCH account
- database accounts used in Data Sources to access Web Services or JMS Queues
- WebLogic accounts used to access Web Services
- database accounts used for OHI Data Marts
- database accounts used for custom database schemas, administration, read-only access, etc.

RELATED PUBLICATIONS

See [#Related Publications SSO](#)

ARCHITECTURE

The following picture provides a general overview of the architecture (it is taken from the standard forms documentation).



Global explanation of the architecture:

- The user starts OHI Back Office in a Browser Session by accessing a URL that is serviced by Oracle HTTP Server (OHS). Eventually, OHS will contact the Forms Servlet to fulfill the request.
- The Access Client (WebGate, installed as a plug-in of OHS) intercepts the request. If the user is not yet validated, the request is redirected to the Authentication Server (Oracle Access Manager, OAM, probably on a different server) *)
- The Authentication Server sees that the requested URL is protected and presents a login screen.
- The user fills in his username and password. This is the centrally defined username/password combination registered in the LDAP Server. OAM checks the username/password combination with the LDAP Server. The LDAP Server usually is the Active Directory server of your company.
- If authentication has succeeded, the Authentication Server provides a cookie for the user's browser and redirects the request back to OHS. The response contains an identifier for the authenticated user. In a proxySSO setup, this must match the user-specific database account. In a commonSSO setup, this can be any customer-specific identifier (e.g. the email address of the employee).
- WebGate again intercepts the request, now sees the cookie and passes the request on to the Forms Servlet, including the user identifier.
- The Forms Servlet retrieves the database credentials for the common user from OPSS. OPSS fulfills the role of "Forms Identity Store". The Forms Servlet then starts the Forms Server process. The Forms Server process logs on to the database, using the database credentials from the Forms Identity Store. For proxySSO configurations, the

common user is a generic proxy database account that can log on on behalf of the user-specific database account as indicated by the user identifier (but not using a user-specific password). For commonSSO configurations, the common user is used for all end user sessions.

- The OHI Back Office application checks if the user identifier has a valid registration in OHI BO (table ALG_FUNCTIONARISSEN) and if so, allows the user access. The OHI Back Office application then activates, when the Forms based UI is used, the secure application role and sets the user context based on the user identifier from the Authentication Server.
- *) In reality, more interaction between OHS, WebGate, Forms Servlet and OAM takes place, e.g. to add the parameter ssoMode=webgate to the request. This signals to OAM that the URL is protected.

Note that the above situation is the simplest one. Federation may also be needed; this is dependent on the way you have configured your network and authorization procedures.

SECURITY

The setup described in this chapter introduces two new communication channels: between OHS/WebGate on the Forms Server and OAM, and between OAM and your company LDAP server. Both channels need to be secured.

Securing communication between OAM and LDAP server

The HTTP communication between the OAM server and your company LDAP servers should be secured, using SSL (more precise: TLS). This requires an exchange of server certificates. The configuration is beyond the scope of this chapter.

Securing communication between WebGate and OAM

OAM has three communication modes:

- OPEN: Use this unencrypted mode if communication security is not an issue in your deployment.
- SIMPLE: Use this Oracle-signed certificate mode if you have some security concerns, such as not wanting to transmit passwords as plain text, but you do not manage your own Certificate Authority (CA). This will use a self-signed certificate provided by OAM.
- CERT: Use if you want different certificates on OAM Servers and WebGates and you have access to a trusted third-party CA.

You will probably want to use CERT mode for your production environment, but that setup is beyond the scope of this chapter. This chapter will describe SIMPLE mode.

For more details, see [About Communication Between OAM Servers and WebGates](#). and [Securing Communication](#).

Be aware that SIMPLE certificates will be de-supported and expire in March 2024, as described in [March 2024 Expiration Of The Oracle Access Manager \(OAM\) Out Of The Box Certificates \(Doc ID 2949379.1\)](#).

SINGLE POINT OF FAILURE

When authentication for OHI is delegated to a central directory server by means of additional intermediate components like OPSS, the availability of these components determines whether the OHI Back Office functionality is available. This means that when you have implemented a high(er) availability setup of your infrastructure components this high availability setup also must apply to the authentication components.

SSO SYSTEM ADMINISTRATION

Managing Users

As stated before, implementing SSO as described in this chapter does not remove the need for activities you already perform, such as:

For proxySSO:

- Create the database account when the user receives access to OHI Back Office
- Drop or lock the database account when the user no longer needs access or access to OHI Back Office is revoked for a certain environment.
- Assign an 'OHI user account record (Dutch: "functionaris") to the database account, within OHI Back Office

For commonSSO:

- Assign an 'OHI user account record (Dutch: "functionaris") to the user identifier from the Authentication Server, within OHI Back Office

For proxySSO and commonSSO:

- Assign the appropriate application roles to the OHI user account within OHI Back Office
- Assign the appropriate authorization for administrative organizations, brands etc.

In addition, you may need to perform extra activities, especially when cloning OHI environments, for each environment. See the Installation Instructions below for more details.

- Synchronize the password of the proxy or common account stored in the "Group Resource" RAD with the database password.
- For proxySSO: Disable SSO and re-enable direct access (when SSO is not implemented in all environments) by removing `PROXY ONLY CONNECT` from the individual database accounts.

You will have to incorporate the appropriate activities into your existing account and environment management procedures.

Monitoring the Single Sign-On Server

With the addition of new components in your landscape, you must monitor these components too:

- Repository database of your Forms domain(s): these used to be required during startup of your WebLogic servers only, but now the OPSS schema needs to be available whenever the OHI BO Forms application needs to be available.
- Repository database of your OAM domain.
- Both OAM Managed Servers

- Your LDAP Server.
- Connection to your LDAP Server(s) from the OAM servers
- Connection from the OHS server(s) in your Forms Domain(s) to the OAM servers.

SOFTWARE REQUIREMENTS

OHI Back Office with SSO has been certified with the following software configuration:

- OHI Back Office release 10.21.7.0.0 or later
- Oracle Fusion Middleware Forms and Reports Services 12cR2 (12.2.1.4.0 and 12.2.1.19.0) in short FRS
- Oracle Identity and Access Management 12cR2 (12.2.1.4.0) – in short OIAM



SSO integration must be manually added to the existing Forms Services, see “[Using Forms Services with Oracle Access Manager](#)” in “Work With Oracle Forms”. This can be found in the Forms [documentation library](#).



In the user’s browser, the security settings must be configured to accept cookies.

LICENSES

You need to make sure you have the proper licenses for the use of OIAM. If you only use the minimum components required for Forms SSO (OAM and OPSS), this may be covered by the OAM Basic license, which is included in your Forms license. If you use OAM for anything else than Forms or use other OIAM components (e.g. for federation) you will need an additional license. Please contact your Oracle Account Manager to discuss your situation in all cases.

INSTALLATION INSTRUCTIONS

The paragraphs below describe how to setup a sample environment for implementing Single Sign-On in combination with OHI Back Office on Oracle Linux. As each customer infrastructure landscape is different this is only meant to illustrate a possible setup, and to highlight some of the configuration choices. This description does NOT replace the detailed installation guides of Forms & Reports and Oracle Identity and Access Management.

It is advised to use these instructions to setup a Proof of Concept environment to determine how to implement the actual production SSO configuration.

Configuration choices

For the certification, the following environment was set up as a starting point.

- Windows 2019 server as Domain controller, running Active Directory
- Linux server for OAM:
 - Oracle Linux Server release 7.9
 - Oracle Database EE 19.11
 - Java JDK 1.8.0_291 (JDK)
- Linux Server for Forms:
 - Oracle Linux Server release 7.9

Oracle Database EE 19.11
Java JDK 1.8.0_291 (JDK)
Oracle Fusion Middleware Infrastructure 12.2.1.4.0
Oracle Fusion Middleware Forms and Reports Services 12cR2 (12.2.1.4.0)
OHI Back Office release 10.21.7.0.0

The following software was used to set up an isolated SSO environment.

- Oracle Fusion Middleware Infrastructure 12.2.1.4.0:
patch 30188255, V983368-01.zip
OPatch 13.9.4.2.7
- Oracle Fusion Middleware Identity and Access Management 12.2.1.4.0:
patch 30188363, V983411-01.zip
patch 32971905: OAM BUNDLE PATCH 12.2.1.4.210607
- WebGate:
patch 31739169: OAM WEBGATE BUNDLE PATCH 12.2.1.4.200811

The remainder of these instructions will assume you are using at least the same software versions and install all software for the owner “oracle”.

These are links to the documentation of Oracle Access Management 12.2.1.4:

[Get Started](#)

[Administering Oracle Access Management](#)

[Oracle Access Management 12.2.1.4.0. - Install](#)

[Tutorial - Installing Oracle Access Management 12c](#)

[Tutorial - Configuring OHS WebGate](#)

[RCU Requirements for Oracle Databases](#)



Attention: Create a new separate WebLogic Server installation (ORACLE_HOME). Do not re-use a WLS environment that is used for the Forms and Reports Services for OHI Back Office. This results in irresolvable conflicts.



Attention: It is assumed you have already installed Oracle Linux and the JDK on the OAM server. For the other components, overview instructions are given.



Attention: Wherever a hostname is needed make sure you use fully qualified domain names (<host name>.<domain name>). Make sure each host is reachable with its fully qualified domain name. Unfortunately, there appears to be a limit to the length of the host name. We encountered issues with host names (for OAM and for WebGate) with a length of 56 and had to omit the domain name to avoid errors in the webgate.log file like “The Access Server has returned a fatal error with no detailed information.”

Create a separate database (PDB) to support OAM

Like any other FMW Infrastructure installation, OAM requires database schemas, including Oracle Platform Security Services (OPSS). They will be created using the Repository Creation Utility (RCU). OAM needs an extra schema.
Consider these schemas to be a part of the OAM Domain.

In an existing or new CDB create the PDB for the Repository, using the requirements from the link above.



It is possible to use an existing PDB, if you choose a unique prefix when running the RCU, and check that the PDB meets the RCU requirements. Do not use a PDB that contains OHI database objects and data.

Install Fusion Middleware Infrastructure software

Create a new Middleware Home by running the FMW installer.

1. Become “oracle”
2. Check your Java settings
3. Change directory (cd) to the directory where you unzipped the FMW Infrastructure software
4. Set up a VNC session or similar
5. Run the Installation Wizard:
`java -jar fmw_12.2.1.4.0_infrastructure.jar`
6. In the Wizard, specify the location of the new Middleware Home (e.g. /ohi/oraBase/product/idm12214)
7. Choose Installation Type ‘Fusion Middleware Infrastructure’

The specified Middleware home will be referred to as \$MW_HOME.

Install OIAM software

Install the Oracle Fusion Middleware Identity and Access Management software in the same Middleware Home.

1. Become “oracle”
2. Check your Java settings
3. Change directory (cd) to the directory where you unzipped the Identity and Access Management software
4. Set up a VNC session or similar
5. Run the Installation Wizard:
`java -jar fmw_12.2.1.4.0_idm.jar`
6. In the Wizard, select the Middleware Home you just created.
7. Choose Installation Type ‘Collocated Oracle Identity and Access Manager (Managed through WebLogic server)’
8. Upgrade OPatch to 13.9.4.2.5+
9. Install the latest Patch Set Bundle for OAM. See My Oracle Support document “OAM Bundle Patch Release History (Doc ID 736372.1)”.

Create supporting database schemas

Use the Repository Creation Utility to create the required schema structures.

1. Become “oracle”
2. Check your Java settings
3. Change directory (cd) to \$MW_HOME/oracle_common
4. Set up a VNC session or similar
5. Run the Installation Wizard:
`./bin/rcu`
6. In the Wizard, choose ‘Create Repository’ and ‘System Load and Product Load’

7. Database Connection Details:
Database type – Oracle Database
Hostname/Port/Service name – the details of the newly created PDB
Username/Password – use SYS with its password, connect as SYSDBA
8. Select Components:
Create a new prefix, e.g., OIAM1
Within “AS Common Schemas”, select
 Common Infrastructure Services *
 Oracle Platform Security Services
 Audit Services
 Audit services Append
 Audit Services Viewer
 Metadata Services
 WebLogic Services *
Within “IDM Schemas”, select
 Oracle Access Manager
9. Set passwords for the schemas
10. Leave the tablespaces default or change them using the button “Manage Tablespaces”
11. Run the actions.
12. Check the status.

Create the OAM Domain

Create the WebLogic Domain. This will create an Admin Server, and two Managed Servers.

1. Become “oracle”
2. Check your Java settings
3. Change directory (cd) to \$MW_HOME/oracle_common/common/bin
4. Set up a VNC session or similar
5. Run the Configuration Wizard:
 ./config.sh
6. In the Wizard, choose “Create a new domain” and specify a Domain Location (e.g., /ohi/domBase/OIAM1). This will be referred to as the \$DOMAIN_HOME
7. Templates:
8. Create Domain Using Product Templates
Select:
 Basic WebLogic Server Domain [wlserver] *
 Oracle Access Management Suite [idm]
 Oracle Enterprise Manager [em]
 Oracle JRF [oracle_common]
 WebLogic Coherence Cluster Extension [wlserver]
9. Application Location:
Specify a new directory (e.g., /ohi/domBase/applications/OIAM1)
WARNING: if another WebLogic Domain exists on this server, you may overwrite that Domain’s Enterprise Manager jar file if you specify the default location.
10. Administrator Account: specify username and password
11. Domain Mode: Production
12. Database Configuration Type:
Specify Autoconfiguration Options using: RCU Data
Vendor: Oracle
Driver: Oracle’s Driver (Thin) for Service connections
Connection Parameters:
 Hostname/DBMS Service/Port – the details of the newly created PDB
 Schema Owner: the prefix you have specified + _STB (e.g. OIAM1_STB)

Button “Get RCU Configuration”

13. JDBC Test:

14. Advanced Configuration:

Select:

Administration Server

Node Manager

Topology

If you plan to make this a Highly Available installation, you may also have to select “Domain Frontend Host Capture” and “Deployments and Services”

15. Administration Server:

Specify a Server Name, a Listen Port, Enable SSL and an SSL Listen Port

16. Node Manager:

Node Manager Type: Per Domain Default Location

Node Manager Credentials: specify username and password

17. Managed Servers:

Modify Server Names and Ports to your standards. Enable SSL and enter an SSL Listen Port

18. Clusters: No action

19. Server Templates:

Remove the 2 templates (if you do not plan to add Managed Servers dynamically)

20. Coherence Clusters:

Modify Cluster Name and Cluster Listen port to your standards

21. Machines:

Remove the machine under the Tab “Machine”

Add a machine under the Tab “Unix Machine”

Specify a Name, Node Manager Listen Address (localhost) and the Node Manager port.

22. Assign Servers to Machines:

Move all three Servers to the right, under your UnixMachine

23. Virtual Targets: No action

24. Partitions: No action

25. Configuration Summary:

Review the configuration and press button “Create”

Start the components of the domain:

1. Become “oracle”

2. Set the Domain environment variables:

. <\$DOMAIN_HOME>/bin/setDomainEnv.sh

3. Start the Node Manager:

4. nohup \$DOMAIN_HOME/bin/startNodeManager.sh &

5. Start the AdminServer:

6. \$DOMAIN_HOME/bin/startWebLogic.sh

Enter username and password when asked.

Wait until you see

<webLogicServer> <BEA-000365> < Server state changed to RUNNING.>

Type <CTRL + Y> to stop the server and then start it in the background with the username and password that was stored:

\$DOMAIN_HOME/bin/startWebLogic.sh &

7. Logon to the WebLogic console and start the Managed Servers:

8. <http://<hostname>:<OAM AdminServer port>/console>

Configure the User Identity Store

Execute the following steps to let Oracle Access Manager use Active Directory as a backend Identity Store.

This is where you specify the details of your company’s LDAP Server (Active Directory).

OAM needs to know where to find your Active Directory Server, and how to find the users defined in it. You will have to confer with the people who manage your company's LDAP Server to agree on an account to use for the connection from OAM and to find out which properties in the LDAP Server contain the username and password of the users.

You will need the following details:

Property	Explanation	Example
Server name	The server name of your Active Directory server	mspdcad1.mycompany.com
Ports	The HTTPS (and HTTP) port of your Active Directory server	
Store Name	Any name. This does not have to match an entry in AD	MyCentralAD
Bind DN	The account to use for the connection, in LDAP notation. This account needs to have appropriate Read and Search privileges for the user and group base DNs	CN=Administrator,CN=Users,DC=mydepartment,DC=mycompany,DC=com
Password	The password of the account specified in the Bind DN	
Login ID Attribute	The LDAP field that holds the central account name of the users	sAMAccountName
User Password Attribute	The LDAP field that holds the central password of the users	userPassword
User Search Base	The starting point for the search for users in the directory tree	DC=mydepartment,DC=mycompany,DC=com
User Filter Object Classes	LDAP Filter for object type User	person
Group Name Attribute	The naming attribute for a group container, if groups reside in a container	<empty>
Group Search Base	The starting point for the search for groups in the directory tree	DC=mydepartment,DC=mycompany,DC=com
Group Filter Object Classes	LDAP Filter for object type Group	<empty>
LDAP user for proxySSO		
username	Username of an LDAP user we want to grant access to the Forms application. This is the value of the "Login ID Attribute" for this user.	pboskabouter
password	Password of the LDAP user we want to grant access to the Forms application	WawWa123#
LDAP user for commonSSO		
username	Username of an LDAP user we want to grant access to the Forms application. This is the value of the "Login ID Attribute" for this user.	paulus.boskabouter@donkerbos.nl
password	Password of the LDAP user we want to grant access to the Forms application	Gregorius123#

1. Open the OAM console in a browser and log on:
<http://<hostname>:<OAM AdminServer port>/oamconsole>

2. Click on the Tab “Configuration” and then on the Tile “User Identity Stores”.
3. Locate the “OAM ID Stores” table and click on the “+ Create”.
4. In the new window, fill in the properties you collected (see table above)
- 5.

Store Name	Any name describing your LDAP service, e.g. MYDEPARTMENT
Store Type	AD: Microsoft Active Directory
Enable SSL	Set checkbox to ‘Checked’.
Location	< The server name of your Active Directory server> e.g. mspdcad1.mycompany.com, optionally with the (SSL) port, e.g. mspdcad1.mycompany.com:636 Do not add http(s)://
Bind DN	account to use for connection to LDAP server, e.g. CN=Administrator,CN=Users,DC=mydepartment, DC=mycompany,DC=com
Password	password for the account in Bind DN
Login ID Attribute	e.g. sAMAccountName
User Password Attribute	e.g. userPassword
User Search Base	e.g. DC=mydepartment,DC=mycompany,DC=com
User Filter Object Classes	e.g. person
Group Name Attribute	e.g. <empty>
Group Search Base	e.g. DC=mydepartment,DC= mycompany,DC=com
Group Filter Classes	e.g. <empty>

6. Click on the button “Test Connection”. If you get an error message “Invalid Identity Store Configuration. The specified URL or credentials are invalid”, disable SSL (checkbox) and try again. You should get the message “Connection to the User Identity Store successful!” Make sure you press the button “Apply”.



Attention: You should enable SSL for the communication between OAM and AD. If you disabled SSL to get a working connection above, you should revisit this page and debug the SSL connection.



Note that we can only retrieve the LDAP username = “Login ID Attribute”. There is no way to register a separate field for a differing database account name to retrieve for the given “Login ID Attribute”.

7. Click on the Tab “Application Security” and then in the Tile “Plug-Ins” on the “+“ button. From the pulldown menu, select “Create LDAP Authentication Module”
8. In the new window:

Name	Any name, e.g. MYDEPARTMENTAD
User Identity Store	select the Store you just created> e.g. MYDEPARTMENT

Press the button “Apply”.

9. Click on the Tab “Application Security” (or on the subtab “Launch Pad”) and then in the Tile “Access Manager” on the “+“ button. From the pulldown menu, select “Create Authentication Scheme”
10. In the new window:

Name	Any name, e.g.
------	----------------

	MYDEPARTMENTscheme
Authentication Level	2
Default	unchecked
Challenge Method	FORM
Challenge Redirect URL	/oam/server
Authentication Module	<your module> e.g. MYDEPARTMENTAD
Challenge URL	/pages/login.jsp
Context Type	default
Context Value	/oam
Challenge Parameters	ssoCookie=disablehttponly

11. Click on the button “Apply” and then on the button “Set as Default”.
12. Restart the OAM Managed Servers and the OAM Admin Server.

For “Simple Mode” encryption, Access Manager includes a certificate authority with its own private key, which is installed across all WebGates and OAM Servers. For “Cert mode” encryption, a trusted CA needs to provide the necessary certificates.

The encryption mode on the OAM server side determines the lowest possible communication mode between OAM and its gateways.

To configure “Simple Mode”:

13. Click on the Tab “Configuration” and then on the Tile “Server Instances”.
14. Search and then Edit the Managed Server you created.
15. In the section “OAM Proxy”, select Mode “Simple”.
16. Optionally change the proxy port from the default of 5575
17. Click on the button “Apply” and confirm the warning “OAM Server instance MS_OAM12214 might be in use. Are you sure you want to edit it?”

Configure WebGate on the Forms Server

This section will configure WebGate on the Forms Server, so it can intercept requests and redirect them to OAM. This section needs to be executed for each Oracle HTTP Server in front of or part of a WebLogic Forms Domain where you want to implement SSO. WebGate is part of the Oracle HTTP Server (OHS) installation, which is installed on the Forms Server for the default architecture (collocated OHS as part of the WLS Forms Domain).

You may want to install the latest Bundle Patch for WebGate on the Forms Server, according to My Oracle Support document “OAM Bundle Patch Release History (736372.1)”. At the time of writing, this is Patch 31739169: “OAM WEBGATE BUNDLE PATCH 12.2.1.4.200811”.



We will use the OAM Remote Registration tool here, as implemented in the Configuration Helper Script. It is possible to do a manual registration of WebGate, copy the generated wallet, etc. and change the OHS configuration manually.

Copy the OAM Remote Registration tool from the OAM Server to the Forms Server.

On the OAM Server:

1. Become “oracle”
2. `scp $MW_HOME/idm/oam/server/rreg/client/RREG.tar.gz <Forms server>:/tmp`

On the Forms Server:

1. Become “oracle”
2. Set environment variables, so \$MW_HOME points to your Forms Software home (e.g. ohi/oraBase/product/frs12214)
3. Unzip the OAM Remote Registration tool:
tar -C \$MW_HOME -xvzf /tmp/rreg_client_12.2.1.4.tar.gz
4. Set the Domain environment variables for the Forms Domain:
. <\$DOMAIN_HOME>/bin/setDomainEnv.sh
5. Start the configuration helper for Forms.

```
$MW_HOME/forms/provision/frmconfighelper.sh enable_sso_ssl -
<OAM Server host name> <OAM Admin Server http port> -
<Forms Server host name> <OHS https port> <OHS http port> $DOMAIN_HOME -
<OHS_component_name>
```

Example:

```
$MW_HOME/forms/provision/frmconfighelper.sh enable_sso_ssl -
myoamserver1.mycompany.com 7001 myformsserver1.mycompany.com 8081 -
8080 DOMFRS12214 ohs12214
```

This will generate output like:

```
running frmconfighelper.sh enable_sso_ssl
using
log file /home.local/oracle/frmconfighelper/logs/frmconfighelper_2021_05_25_23
_10_01.log .....
JAVA_HOME=/usr/java/jdk1.8.0_latest
CLASSPATH= .....
OAM_REG_HOME=/ohi/oraBase/product/frs12214/rreg/bin/..
-----
Welcome to OAM Remote Registration Tool!
Parameters passed to the registration tool are:
Mode: inband
Filename: /home.local/oracle/frmconfighelper/FormsOAMRegRequest2Ports.xml
Enter admin username:weblogic
Username: weblogic
Enter admin password: <password>
Do you want to enter a Webgate password?(y/n):
n
Do you want to import an URIs file?(y/n):
n
-----
Request summary:
OAM11G Agent Name: myformsserver1.mycompany.com_OAM
Base URL:http:// myformsserver1.mycompany.com:8080
URL String: myformsserver1.mycompany.com_HostId
Registering in Mode:inband
Your registration request is being sent to the Admin server at: http:// -
myoamserver1.mycompany.com:7001
```

Inband registration process completed successfully! Output artifacts are created in the output folder.

The Configuration Helper Script registers the local WebGate with the OAM server and configures WebGate. It copies files generated on the OAM server to the WebGate configuration in `$DOMAIN_HOME/config/fmwconfig/components/OHS/<OHS_component_name>/webgate`. These files are then copied to the OHS instance directory `$DOMAIN_HOME/config/fmwconfig/components/OHS/instances/<OHS_component_name>/webgate` when OHS and the Admin Server are restarted.



If you encounter issues with the Configuration Helper Script, see this document on My Oracle Support:
[Troubleshooting Forms 12c SSO Issues after Running Forms Configuration Helper Script \(frmconfighelper\) \(Doc ID 2632507.1\)](#)

6. Stop the OHS server, the Forms Admin Server and the Forms Managed Server.



Restarting the Admin Server ensures that changes in the OHS configuration are propagated to the OHS instance configuration.



Be aware that the file `oblog_config_wg.xml` is required in the OHS instance configuration directory. Symptom: Internal Server Error message in the browser when accessing any OHS URL.
If this file is not propagated from the OHS configuration directory, you need to copy this file manually:

```
cp
$DOMAIN_HOME/config/fmwconfig/components/OHS/<OHS_component_name>/webgate/config /oblog_config_wg.xml
$DOMAIN_HOME/config/fmwconfig/components/OHS/instances/<OHS_component_name>/webgate/config
```



If your servers use official certificates for HTTPS, the next steps may not be necessary. Continue by [starting the servers](#).

7. Return to the OAM console at <http://<hostname>:<OAM AdminServer port>/oamconsole>
8. Click on the Tab “Application Security” and then on the Tile “Agents”.
9. Search and Edit the “<forms server>_OAM” SSO Agent that was created by the `frmconfighelper`.
10. In the next screen edit the “User Defined Parameters” and add two lines:
`SSLVerifyPeerCert=false`
`SSLVerifyHostname=false`
11. Press the button “Apply”.



Each “Apply” action (even when nothing has been changed) creates new output files in the directory `$DOMAIN_HOME/output/<forms server>_OAM`. This includes a new encrypted password in the file `password.xml`. These need to be deployed on the WebGate host.

On the OAM Server:

12. Become “oracle” and execute

```
scp $DOMAIN_HOME/output/<forms_server_OAM>/ <Forms server>:/tmp
```

On the Forms Server:

13. Become “oracle”

14. Set environment variables, so \$DOMAIN_HOME points to your Forms Software home (e.g. ohi/oraBase/product/frs12214)

15. Execute the following commands to distribute the new versions:

```
cp -r /tmp/fsgbu-lon-52_OAM/* -
$DOMAIN_HOME/config/fmwconfig/components/OHS/<OHS_component_name>/webgate/config
cp $ORACLE_HOME/webgate/ohs/config/oblog_config_wg.xml -
$DOMAIN_HOME/config/fmwconfig/components/OHS/<OHS_component_name>/webgate/config
rm -rf
$DOMAIN_HOME/config/fmwconfig/components/OHS/instances/<OHS_component_name>/webgate*
rm $DOMAIN_HOME/servers/<OHS component name>/cache/*
```

16. Start the Forms Admin Server, Forms Managed Server and OHS.

17. Test that your existing OHI BO Forms application (e.g. OHIDEV1) still starts up normally and shows the familiar logon screen. If you get an “Internal Server Error” message in the browser, that indicates a misconfiguration of the communication between WebGate and OAM.



We’ve noticed that applying SSL using the Configuration Helper Script breaks the possibility to run forms configurations (with or without SSO) with HTTP. They only work with HTTPS.

Symptom: During startup of the Forms application, the Java Runtime reports “java.lang.SecurityException: illegal URL redirect”

This happens in “Open Mode” as well as in “Simple Mode”. You may want to redirect all HTTP traffic to HTTPS using the OHS configuration.

18. Check the contents of the new WebGate log file \$MW_HOME/servers/<OHS server>/logs/webgate.log to make sure WebGate was activated and encountered no errors.

Test basic Single Sign-On

To test the basic Single Sign-On functionality, add a “named configuration” to your formsweb.cfg file with the ssomode flag:

As “oracle” on the Forms server:

1. Set the Domain environment variables for the Forms Domain:

```
. <$DOMAIN_HOME>/bin/setDomainEnv.sh
```
2. vi \$DOMAIN_HOME./config/fmwconfig/servers/<WLS Forms Server>/applications/formsapp_12.2.1/config/formsweb.cfg
 Copy the Named Configuration section of the OHI BO Forms application you used to test in step 17 above (e.g. OHIDEV1), give it a new name

For proxySSO:

(use e.g. OHIDEV1_SSO), and add the following lines:

```
ssomode=webgate
```

```
ssoproxyconnect=yes
```

```
ssodynamicresourcecreate=true (true is for testing only)
```

For commonSSO:

(use e.g. OHIDEV1_SSO2), and add the following lines:

```
ssoMode=webgate
```

```
ssoProxyConnect=no
```

```
ssoDynamicResourceCreate=true (true is for testing only)
```

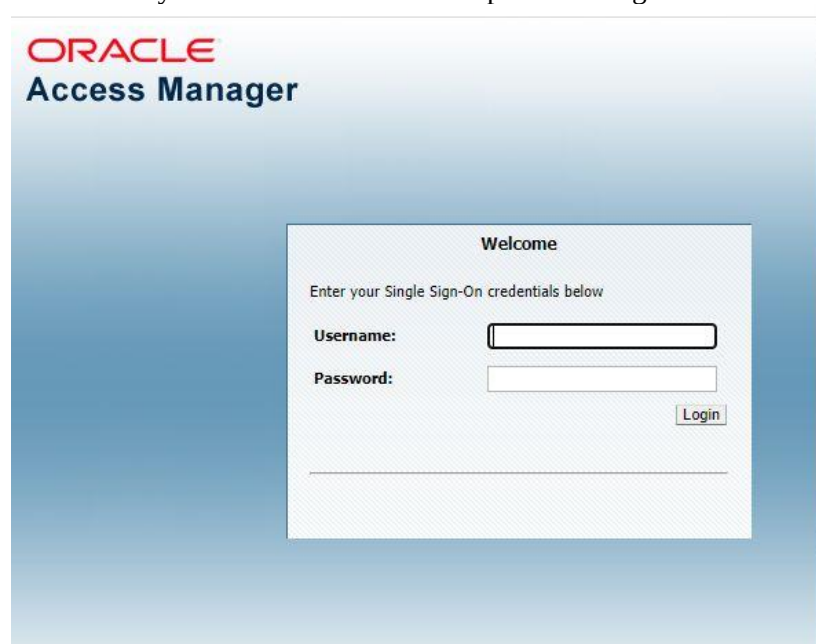


If you use the Forms StandAlone Launcher (FSAL), you need to add

```
ssoMode=%ssoMode%
```

to the basesaa.txt and webutilsaa.txt template files. Additional optional parameters are available. See [Using Forms Services with Oracle Access Manager](#)

3. Test SSO by entering the same URL as for OHIDEV1, but now with `?config=OHIDEV1_SSO ?config=OHIDEV1_SSO2` or at the end, instead of `?config=OHIDEV1`.
4. This should redirect you to the OAM server and present a Logon screen



5. Log on with the LDAP user you determined before [LDAP user]
6. This should redirect you to a page “Create Resource Access Descriptor” like the one below.

Create Resource Access Descriptor

Forms application BATEA16_SSO

Forms application database connection details

Userid	
Password	
Database	
Submit	Cancel

This is proof that the basic SSO functionality has been configured correctly. We now need to configure the link from the LDAP user to the existing OHI BO “functionaris” for the user, so the page “Create Resource Access Descriptor” can be skipped.

Configure database access

SSO access to the database from Forms requires a Resource Access Descriptor (RAD) that is registered in Oracle Platform Security Services (OPSS) of the Forms Domain. Data is stored in the OPSS database schema that was created by the Repository Creation Utility (RCU) when the Domain was created.

Forms uses the “Named Configuration” in the request URL (the value after `?config=`) to find out if Single Sign-On is used, by reading the file `formsweb.cfg` and then looks in OPSS for a RAD with the same name as the Named Configuration. Forms then uses the credentials of the RAD to log on to the database.

For proxySSO:

The LDAP username is used to create another session with that username. This session is used from then on. This means that the LDAP username must match the personal database account of the LDAP user and that account must be open.

For commonSSO:

The common database user from the RAD continues to be used. That means all Forms users will use database sessions for that same common user.

- Notice that the RAD does not “know” if it is used for proxySSO or commonSSO. That is solely determined by the setting for `ssoProxyConnect` in `formsweb.cfg`.

There are different ways to use RADs:

1. If no RAD is present for the requested Named Configuration, by default the user will be asked to create one (see the screen above). You probably do not want that. Users would have to create a RAD for each environment and have no way to reset their database password in the RAD. This is not easily managed. Set `ssoDynamicResourceCreate=false` to suppress the RAD screen.
2. For proxySSO:

If the LDAP username is the same as the database username, you can register a single “Group Resource” RAD per OHI environment, that will use a database proxy account to log on to the database for all users.

If the LDAP username is NOT the same as the database username, we need a mapping between the two. This requires individual RADs for each user for each OHI BO environment. You will have to create, update and delete this as part of your account provisioning (including database password reset actions). This is impractical and defeats the purpose of Single Sign-On.

In some Authentication Servers, you may be able to return an alias or alternate identifier instead of the (company-wide) LDAP username, to match the personal database account. In that case you can use the “Group Resource” RAD described below. If that does not work in your situation, we suggest using commonSSO instead of proxySSO.

3. For commonSSO:

The LDAP username (or alias) is not relevant for access to the database. As long as a RAD exists for the requested Named Configuration, a database session will be created for the common user from the RAD. The LDAP username or alias is then used within OHI BO to determine the correct “Functionaris” and set the correct privileges. If no matching “Functionaris” can be found, the Forms application and the database session will be terminated.

Actions for using Group Resource RADs for options 2 and 3 are discussed below. Using individual RADs is considered out of scope.

1. Modify the forms configuration.
As “oracle” on the Forms server:

Set the Domain environment variables for the Forms Domain:

```
. <$DOMAIN_HOME>/bin/setDomainEnv.sh
```

Modify the Named Configuration in formsweb.cfg

```
vi $DOMAIN_HOME./config/fmwconfig/servers/<WLS Forms  
Server>/applications/formsapp_12.2.1/config/formsweb.cfg
```

In the Named Configuration section you created earlier (e.g. OHIDEV1_SSO) set the SSO parameters as below:

For proxySSO:

```
ssoMode=webgate  
ssoProxyConnect=yes  
ssoDynamicResourceCreate=false
```

For commonSSO:

```
ssoMode=webgate  
ssoProxyConnect=no  
ssoDynamicResourceCreate=false
```

Optionally, add a line that points to your custom page with a custom error message when the RAD is not present for the requested OHI BO environment or the proxied account has issues.

```
ssoErrorURL=http://www.mycompany.nl/MyRADerrorpage.html
```

The user will now be redirected to your page instead of getting a page with the message:

Oracle Fusion Middleware Forms Services

FRM-93330: Fatal authentication error: User does not have proper credentials configured.

2. Create the database account for the RAD:

For proxySSO:

Create a proxy account in the OHI BO database with minimal privileges. As an example, we will use the name ohimdtierproxy.

As a DBA in the OHI BO database, execute:

```
CREATE USER ohimdtierproxy IDENTIFIED BY <midtierPW>;
GRANT CONNECT,CREATE SESSION TO ohimdtierproxy;
```

For commonSSO:

Create a common account in the OHI BO database with minimal privileges. This account **MUST** be named OHI_SSO_COMMON_USER.

```
CREATE USER ohi_sso_common_user IDENTIFIED BY <midtierPW>;
GRANT CONNECT,CREATE SESSION TO ohi_sso_common_user;
```



Make sure the password never expires or add the account to your routines for periodic password change.

For proxySSO:

3. Allow the existing OHI BO user “pboskabouter” to access the database through ohimdtierproxy. To demonstrate that user “pboskabouter” still has direct access to the database and via the Forms application, change his password to a known value. As a DBA in the OHI BO database, execute:

```
ALTER USER pboskabouter GRANT CONNECT THROUGH ohimdtierproxy;
ALTER USER pboskabouter identified by Oehoeboeroe;
```

Test that the proxy connection works:

```
connect ohimdtierproxy[pboskabouter]/<midtierPW>@OHIDEV1
show user
```

USER is "PBOSKABOUTER"

Test that the OHI BO user still has direct access to the database:

```
connect pboskabouter/Oehoeboeroe@OHIENV
show user
```

USER is "PBOSKABOUTER"

Be sure to reset the password of the user after the testing, as later on is described in this paragraph, is finished.

For commonSSO:

4. Link the LDAP username (or alias) to a “Functionaris” record in OHI BO. If you are providing SSO access for an existing OHI BO user, you can fill in the LDAP user name/alias in column SSO_USERID in table ALG_FUNCTIONARISSEN, using the screen SYS1017F (Menu option Systeem -> Beheer -> Autorisatie -> Gebruiker).

Assuming `pboskabouter` already exists as a “functionaris”, set `SSO_USERID` to “`paulus.boskabouter@donkerbos.nl`” to link the “functionaris” to that LDAP username.

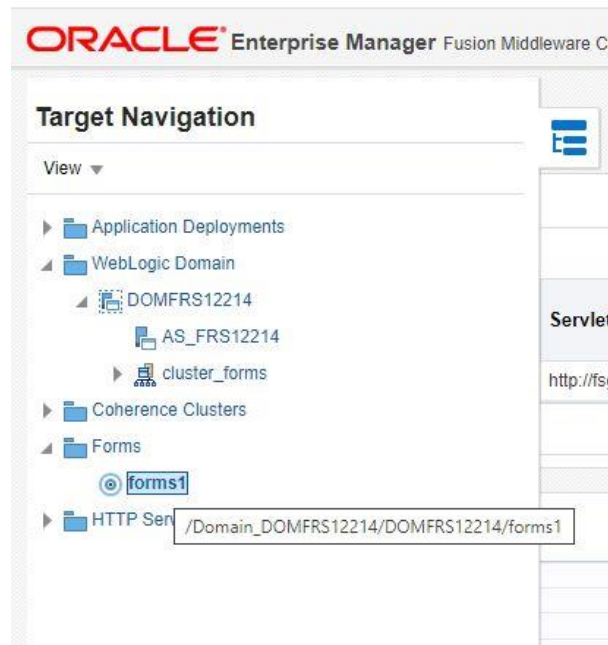
As long as the corresponding database account `pboskabouter` is active (and can connect through proxy account `ohimidtierproxy`), this same “functionaris”-record can be used for classical local logon, proxySSO and commonSSO, all at the same time.

If you decide to use commonSSO exclusively, drop the database account `pboskabouter` and no longer create database accounts for new users.

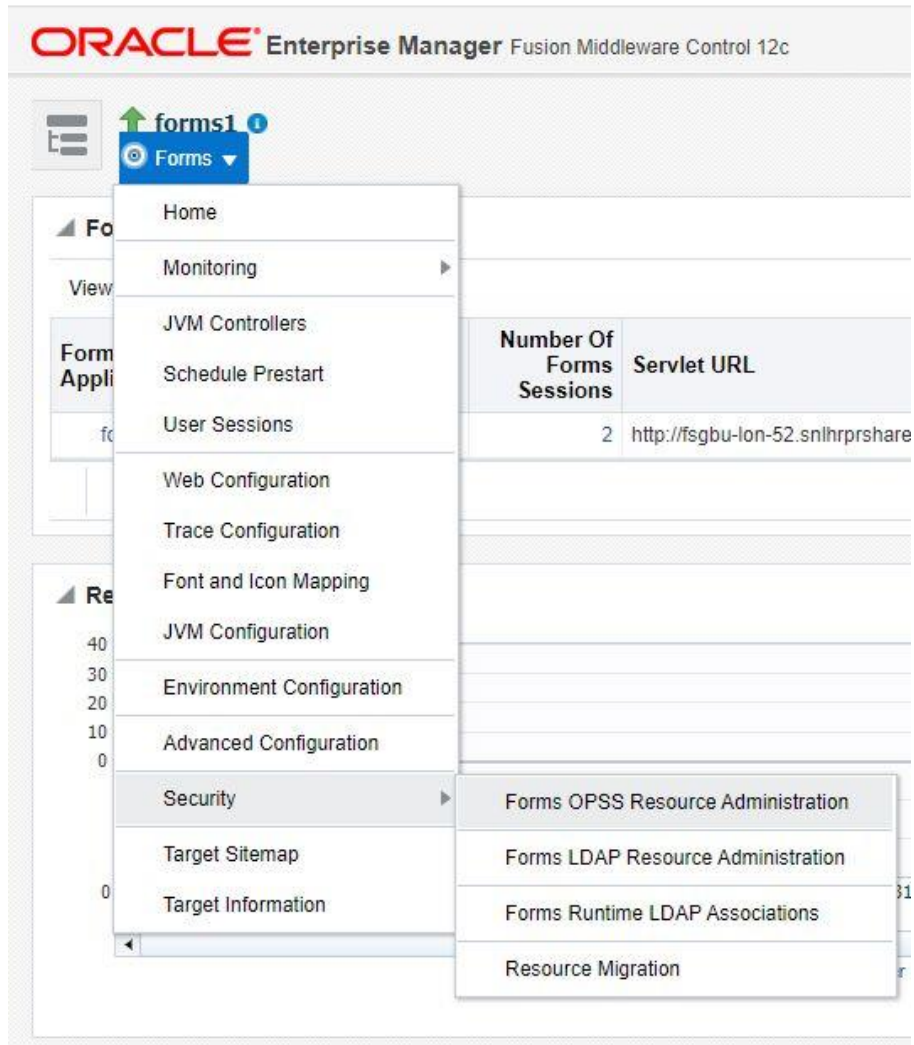
5. Create a Group Resource based on the proxy or common account. This will be used by all LDAP users to access the OHIBO database.

Start the Fusion MiddleWare Enterprise Manager Console of your Forms installation:
`http://<formsserver>:<adminserver port>/em`

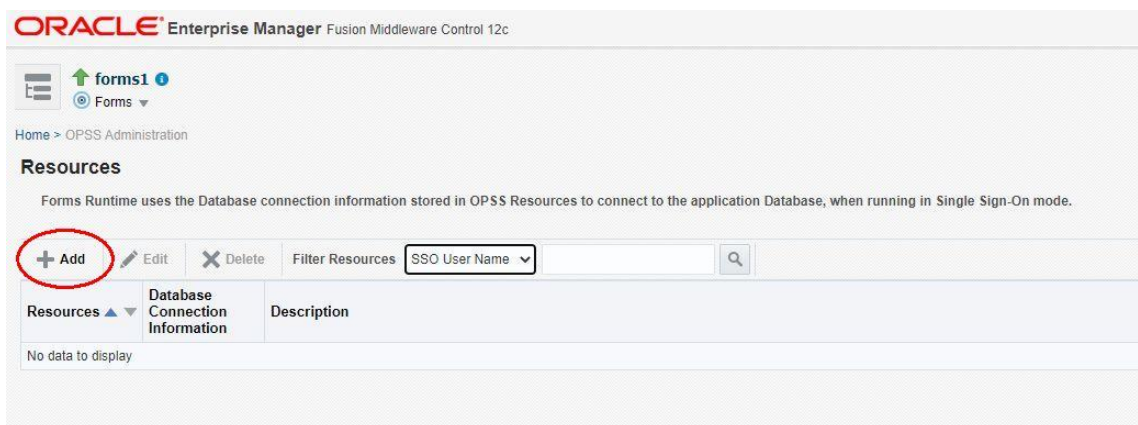
Open the Target Navigation sidebar and select Forms -> forms1.



From the `forms1` menu dropdown, select Security -> Forms OPSS Resource Administration.



In the screen "Resources", click on the button "Add"



In the screen "Add Database Connection Information", add:
For proxySSO:

Forms Application	Select your named config, e.g. OHIDEV1_SSO
Create Group Resource	Checked
Database Username	ohimidtierproxy
Database Password	midtierPW
Database Name	e.g. OHIDEV1

For commonSSO:

Forms Application	Select your named config, e.g. OHIDEV1_SSO2
Create Group Resource	Checked
Database Username	ohi_sso_common_user
Database Password	midtierPW
Database Name	e.g. OHIDEV1

Add Database Connection Information

* Forms Application: OHIDEV1_SSO ☒ Create Group Resource

SSO User Name:

Comments:

* Database User Name: ohimdtierproxy

* Database Password:

* Database Name: OHIDEV1

OK Cancel

Test SSO Access to OHI BO

1. Test SSO access to the OHI BO Forms application by entering the same URL as before with the correct Named Configuration at the end:

For proxySSO:

?config=OHIDEV1_SSO

For commonSSO:

?config=OHIDEV1_SSO2

2. This should again redirect you to the OAM server and present a Logon screen.



3. Log on with the LDAP user you determined before [LDAP user],
For proxySSO:
e.g. pboskabouter and the LDAP password, e.g. WawWa123# (not the database password Oehoeboeroe) .
For commonSSO:
e.g. paulus.boskabouter@donkerbos.nl and the LDAP password, e.g. Gregorius123#
4. This should now start downloading the jnlp file and start the application without the usual Forms logon dialog.
5. Within the application, select the menu option “Show” → “Info”. This will show the actual user is pboskabouter, not ohimdtierproxy or ohl_sso_common_user. The “Sessie info” button will show:
Single Sign-On : Yes
SSO type : Proxy or Common
6. Close the Forms application, but not the browser.
In the browser, enter the same URL as before with ?config=OHIDEV1_SSO or ?config=OHIDEV1_SS02 at the end.
7. This should start downloading the jnlp file and start the application without the OAM logon screen or the usual Forms logon dialog.
8. Close the Forms application, but not the browser.
9. In the browser, navigate to the OAM global logout page, at
http(s)://<oamserver>:<managedserverport>/oam/server/logout
10. In the browser, enter the same URL as before with ?config OHIDEV1_SSO or ?config OHIDEV1_SS02 at the end.
11. This should now redirect you to the OAM Logon screen again.

Optional actions

Block direct database account access

With the configuration actions and the changes to the OHI BO user account (pboskabouter), the user can still log on to the database and start the OHI BO Forms application with another Named Configuration in the URL. If you want to prevent this and only allow access using SSO execute the following:

For proxySSO:

As a DBA in the OHI BO database, execute:

```
ALTER USER pboskabouter PROXY ONLY CONNECT;
```

Test direct access:

```
connect pboskabouter/Oehoeboeroe@<OHIENV>  
This will result in:  
ERROR:ORA-28058: login is allowed only through a proxy
```

To revert:

```
ALTER USER pboskabouter CANCEL PROXY ONLY CONNECT;
```

For commonSSO:

As a DBA in the OHI BO database, execute:

```
DROP USER pboskabouter CASCADE;  
or (less drastic):  
ALTER USER pboskabouter account lock;
```

Limit validity of the jnlp file

In contrast with the generic jnlp file that is downloaded for the classic login without SSO, the jnlp file for SSO has a session identifier that can be used (only once) to get access to the Forms application. If a user downloads the file but does not continue to the Forms application, or if the file is intercepted, this poses a security risk. Therefore, you should limit the validity of the jnlp file c.q. the SSO-supplied session, by setting the following parameter in the named section (e.g. OHIDEV1_SSO or OHIDEV1_SSO2) in `formsweb.cfg`. The value is in seconds.
NOTE: this requires Patch 34316706 or Forms 12.2.1.19.0.

```
jnlpTimeout=30
```

Limit origin of the jnlp file

It is possible to restrict the use of the jnlp file further:

`jnlpMatchIP` specifies whether a JNLP file will be rejected if launched from an IP address that differs from where the JNLP file was generated. Note that this may not work reliably when behind a proxy server (e.g. load balancer, OHS). Set the following parameter in the named section (e.g. OHIDEV1_SSO or OHIDEV1_SSO2) in `formsweb.cfg`.

```
jnlpMatchIP=TRUE
```

Hide the connection details of the RAD after a failed logon attempt

When a user tries to logon to the Forms application using SSO and the logon fails, the forms application shows the non-SSO logon dialog screen by default. This logon screen will then display the connection details of the proxy user used in the RAD (but not the password). This may be confusing. It is possible to show a blank logon screen instead, by setting an environment variable.

To do so, add a line to the `<servlet>.env` file you specified for the named configuration:

```
FORMS_LOGIN_DIALOG_BLANK=TRUE
```

The Forms documentation also describes `FORMS_HIDE_LOGIN_RESET`, but that does not hide the login dialogue in our tests.

Protect OHI BO batch processing log files and output files

Without SSO, there is no practical way to protect the log files and output files generated by OHI BO (e.g. for batches). With SSO, we can now protect those files by registering the URLs in OAM.

1. Open the OAM console in a browser and log on:
<http://<hostname>:<AdminServer port>/oamconsole>
2. Click on the Tab “Application Security” and then on “Application Domains” in the Tile “Access Manager”.
3. Click on the button “Search”, select the Application domain with your Forms server name (where WebGate has been configured), and click on “Edit”.
4. In the new screen, click on the Tab “Resources” Click on the button “Search” and select a Resource that is protected, e.g. /reports/rwervlet/*. Then click on button “Duplicate”.
5. In the new screen, change the URL to the URL for the log files, e.g.
/OHI/OHIDEV1/ohiBase/log/*
and click on the button “Apply”.
6. Click on the button “Duplicate”, change the URL to e.g.
/OHI/OHIDEV1/ohiBase/out/*
and click on the button “Apply”.



The URLs are determined by the OHI system parameters for the virtual output and log directories. The /OHI/ part is an alias defined in forms.conf. The URLs point to \$OZG_LOG and \$OZG_OUT on the Forms server. To test the protection of these files, either double-click on a recent batch log record in the OHI batch screen SYS1001F or construct a URL from an existing file in \$OZG_LOG and \$OZG_OUT.



If you want to protect the log and out files of all OHI BO environments on the server, use wildcards in the URLs, e.g., /OHI/*/ohiBase/log/* and /OHI/*/ohiBase/out/*



When testing if a URL is protected, be aware of browser caching, especially for static files like the log and out files. Always close your browser and start a new browser session before and in between tests.

Additional information

- Each “Named Configuration” section in formsweb.cfg that you want to configure for SSO needs its own RAD, because the name is part of the key of the RAD. This means that when you do not use a Default RAD and a proxy account you will have to create user specific RADs for each section in the formsweb.cfg where SSO is used. Because these individual RADs contain the password for the database account, these need to be kept in sync with the actual password in the database. It is clear that RAD creation and maintenance must become a part of your account provisioning and management procedures.

For proxySSO:

- Each user that connects to the OHI Back Office user interface through a proxy user will have an additional database session for that proxy user during the lifetime of the application session. This consumes a little more memory. This is a moderate amount, about 2MB per session.

- RAD creation, updates and deletion can be automated with WLST scripts that connect to the Admin Server of the WLS Domain. The commands to use are:

For proxySSO:

```
createCred(map="FormsComponent", key=";OHIDEV1_SSO",
user="ohimidtproxy", password="ohimidtproxy/midtproxyPW@
OHIDEV1", desc="SSO with named user accounts via proxy")

updateCred(map="FormsComponent", key=";OHIDEV1_SSO", user="
ohimidtproxy", password="ohimidtproxy/NewmidtproxyPW@
OHIDEV1", desc="updated SSO with named user accounts via proxy")

deleteCred(map="FormsComponent", key=";OHIDEV1_SSO")
```

For commonSSO:

```
createCred(map="FormsComponent", key=";OHIDEV1_SSO2",
user="ohi_sso_common_user", password="ohi_common_user/midtproxyPW@
OHIDEV1", desc="SSO without named user accounts")

updateCred(map="FormsComponent", key=";OHIDEV1_SSO2", user="
ohi_sso_common_user ", password="ohi_sso_common_user
/NewmidtproxyPW@OHIDEV1", desc="updated SSO without named user
accounts")

deleteCred(map="FormsComponent", key=";OHIDEV1_SSO2")
```



NOTE: For Group Resource RADs, you need to specify `<username>/<password>@<servicename>` in the `password=` parameter.



NOTE: `updateCred` appears to create the RAD if it is not present, so `createCred` is not really needed.

- The introduction of SSO will have consequences for your procedures to copy/clone databases, e.g., to refresh environments.
 - You need to make sure the password for the proxy account in the database matches that in the RAD, because the OPSS store is (usually) not stored in the same PDB.
 - Be aware that you still need to lock individual database accounts if you want to limit access to copied databases, even if you use proxy connections.
- These instructions only describe the enablement of Single Sign On for multiple Forms logon actions.
Aspects like synchronization with other directories than Active Directory, provisioning of users, self-service for users (for changing their SSO password, request additional access, etc.), etc. need to be addressed too.
This is typically very customer-specific and may require additional Identity Management products.

Removing WebGate

To remove the WebGate configuration from your Forms Server:

Remove the extra named configurations from `formsweb.cfg`

- Use the EM console to remove the RADs
- Stop OHS, the Forms Managed Server, the Admin Server and Node Manager in your Forms domain
- Execute the following commands on your Forms Server
- Become “oracle” and execute:

```
cd $DOMAIN_HOME
rm servers/<OHS_component_name>/cache/*
rm servers/ohs12214/logs/webgate.log

cd $DOMAIN_HOME/config/fmwconfig/components/OHS/<OHS_component_name>
rm webgate.conf
rm -rf webgate
vi httpd.conf
    Remove the last line:
        include "webgate.conf"

cd $DOMAIN_HOME/config/fmwconfig/components/OHS/instances/<OHS_component_name>
rm webgate.conf
rm -rf webgate
vi httpd.conf
    Remove the last line:
        include "webgate.conf"
```

- Start the Node Manager, the Admin Server, the Forms Managed Server and OHS in your Forms domain.

Access to OHIJET Application

Protect resources

The resources which are used by the OHIJET application must be protected. This can be done by registering these in OAM.



NOTE: This assumes all traffic for OHIJET will run via the Oracle HTTP Server (that is protected by WebGate) to the WebLogic Managed Server. You need to block direct access to the Managed Server where OHIJET is deployed, to prevent SSO can be circumvented.

1. Open the OAM console in a browser and log on:
<http://<hostname>:<AdminServer port>/oamconsole>
2. Click on the Tab “Application Security” and then on “Application Domains” in the Tile “Access Manager”.
3. Click on the button “Search”, select the Application domain (where WebGate has been configured), and click on “Edit”.
4. In the new screen, click on the Tab “Resources”. Click on the button “Action” and then “Create”.
5. In the new window “Create Resource” fill in the fields with following values:
 - Type = HTTP
 - Host Identifier = current host of application domain
 - Resource URL = /ohibo
 - Operations = select All (is default)
 - Protection Level = Protected
 - Authentication Policy = Protected Resource Policy
 - Authorization Policy = Protected Resource Policy
6. Click on button “Apply”
7. Click on the button “Action” and then “Create”.
8. In the new window “Create Resource” fill in the fields with following values:
 - Type = HTTP
 - Host Identifier = current host of application domain
 - Resource URL = /ohibo/../*
 - Operations = select All (default)

- Protection Level = Protected
 - Authentication Policy = Protected Resource Policy
 - Authorization Policy = Protected Resource Policy
9. Click on button “Apply”
 10. Click on tab “Authorization Policies” and then “Protected Resource Policy”
 11. Click on tab “Responses” and then on “Add”
 12. In the new window “Add Response” fill in the fields with following values:
 - Type = Cookie
 - Name = OHI_SESSION_ID
 - Value = \${session.id}

Oracle HTTP server

Configure Oracle HTTP server to include locations for the WebLogic managed server on which OHIJET is deployed, for example:

```
#Location for OHIJET
<Location /ohibo>
    SetHandler weblogic-handler
    WebLogicCluster localhost:7721
    DynamicServerList OFF
</Location>

#location for HSL services
<Location ~ "^/HSL">
    SetHandler weblogic-handler
    WebLogicCluster localhost:7721
    DynamicServerList OFF
</Location>

#location for PSL services
<Location ~ "^/PSL">
    SetHandler weblogic-handler
    WebLogicCluster localhost:7721
    DynamicServerList OFF
</Location>
```

OAM sends the session_id in a response cookie named OHI_SESSION_ID. Ensure this cookie has the secure and http_only flags set, for example:

```
<IfModule mod_headers.c>
    Header always edit Set-Cookie ^(.*)$ $1;HttpOnly;Secure
</IfModule>
```

Weblogic

In the WebLogic configuration of the managed server for OHIJET ensure to have the property “WebLogic Plug-In Enabled” set to “Yes”.

Configuration	Protocols	Logging	Debug	Monitoring	Control	Deployments	Services	Security	Notes				
General	Cluster	Services	Keystores	SSL	Federation Services	Deployment	Migration	Tuning	Overload	Concurrency	Health Monitoring	Server Start	Web Service

Click the **Lock & Edit** button in the Change Center to modify the settings on this page.

[Save](#)

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

[View JNDI Tree](#)

Name:	MS_SVL12214_BAJET21
Template:	(No value specified) Change
Machine:	localmachine
Cluster:	(Stand-Alone)
Listen Address:	
☒ Listen Port Enabled	
Listen Port:	6721
☒ SSL Listen Port Enabled	
SSL Listen Port:	7721
☐ Client Cert Proxy Enabled	
Java Compiler:	javac
Diagnostic Volume:	Low
Default Datasource:	

[Advanced](#)

Virtual Machine Name:	DOMSVL12214_MS_SVL:
WebLogic Plug-In Enabled:	yes

APPENDIX D - INSTALLING REQUIRED PERL MODULES

INTRODUCTION

The following additional Perl modules need to be installed for the OHI installation menu OHIPATCH to work.

Some modules are also used by OHI application software.

All modules can be retrieved from www.cpan.org, the links are provided in the table below. Minimum versions are shown that have been verified to work well for Linux7 and Linux8, respectively. Newer versions usually should work. Oracle does not provide support on Perl modules.

Set	Installation order	Name of CPAN Perl module	Version Linux7	Version Linux8	Available as yum rpm
		File::NCopy	0.34	0.36	No
		File::Remove	0.34	1.57	Yes
		Shell	0.73	0.73	No
		Shell::Source	0.01	0.01	No
		Term::ReadKey	2.30	2.37	Yes
		Carp	1.26	1.42	Yes
1	1	Test::Simple	0.98	1.302135	Yes
	2	Test::Warn	0.24	0.32	Yes
	3	File::Path	2.09	2.15	Yes
	4	Parallel::ForkManager	1.06	2.02	Yes
	5	DBI	1.607	1.641	Yes
	6	DBD::Oracle	1.74	1.74	No



Note 1: The `gcc` compiler is required for installation of the `Term::ReadKey` and `DBD::Oracle` modules.



Note 2: All modules *in a set* should be installed in a *specific order*, as indicated in the table above.



Note 3: Installing module `Term::ReadKey` on Oracle Enterprise Linux 7 can give errors during generating the module. See [Term::ReadKey issue](#).



Note 4: `DBD::Oracle` versions above 1.74 have an issue in combination with database PSU 19.15 (when installed on the Application Server). Symptom: Parallel compilation of Forms and Pro*C modules hangs in OHIPATCH, e.g. in step 800. Stay on version 1.74 until further notice.

MANUAL INSTALLATION PROCEDURE

The following steps have to be followed to install each module manually.

The steps must be executed by `root`. Note that easier installation methods are possible, using either `cpan` or `yum` utilities. See below.

- Download the zip file concerned to the application server which will run the installation menu and place it in a temporary directory, e.g. `$TMP`.

- Unzip the zip file into a temporary directory.
- Navigate to the directory created for the module.
- Generate instructions for the installation by running the make file:

```
perl Makefile.PL
```

- Run `make` to create the lib installation directory:

```
make
```

- Test the functionality of the module with

```
make test
```

- Install the module with

```
make install
```

- Delete the temporarily created directory.

Example

The following is an example installation of the `Term::ReadKey` module on Linux 8:

```
# cd /tmp
# tar -xvf TermReadKey-2.38.tar.gz      # *** step 2 ***
TermReadKey-2.38/
TermReadKey-2.38/example/
TermReadKey-2.38/example/test.pl
TermReadKey-2.38/META.yml
TermReadKey-2.38/genchars.pl
TermReadKey-2.38/Makefile.PL
TermReadKey-2.38/ReadKey.pm.PL
TermReadKey-2.38/MANIFEST.SKIP
TermReadKey-2.38/t/
TermReadKey-2.38/t/02_terminal_functions.t
TermReadKey-2.38/t/01_basic.t
TermReadKey-2.38/ReadKey.xs
TermReadKey-2.38/Configure.pm
TermReadKey-2.38/MANIFEST
TermReadKey-2.38/META.json
TermReadKey-2.38/SIGNATURE
TermReadKey-2.38/ppport.h
TermReadKey-2.38/Changes
TermReadKey-2.38/README

# cd TermReadKey-2.38 # *** step 3 ***

# perl Makefile.PL # *** step 4 ***

Checking if your kit is complete...
Looks good
Generating a Unix-style Makefile
Writing Makefile for Term::ReadKey
Writing MYMETA.yml and MYMETA.json
# make test # *** step 6 ***

Running Mkbootstrap for ReadKey ()
...
```

```
...
t/01_basic.t ..... ok
t/02_terminal_functions.t .. ok
All tests successful.
Files=2, Tests=8,  0 wallclock secs ( 0.01 usr  0.01 sys +  0.07 cusr  0.01
csys =  0.10 CPU)
Result: PASS

# make install # *** step 7 ***

"/usr/bin/perl" -MExtUtils::Command::MM -e 'cp_nonempty' -- ReadKey.bs
blib/arch/auto/Term/ReadKey/ReadKey.bs 644
"/usr/bin/perl" "-Iblib/arch" "-Iblib/lib" ReadKey.pm.PL ReadKey.pm
Creating ReadKey.pm
Bootstrapping the XS for blockoptions: 5
Done
Skip blib/arch/Term/ReadKey.pm (unchanged)
Manifying 1 pod document
Files found in blib/arch: installing files in blib/lib into architecture
dependent library tree
Installing /usr/local/lib64/perl5/auto/Term/ReadKey/ReadKey.so
Installing /usr/local/lib64/perl5/Term/ReadKey.pm
Installing /usr/local/share/man/man3/Term::ReadKey.3pm
Appending installation info to /usr/lib64/perl5/perllocal.pod
# cd ..
# rm -Rf TermReadKey-2.38 # *** step 8 ***
```

MORE AUTOMATED INSTALLATION PROCEDURE

You can use CPAN to install modules, as this is more convenient. However, this requires your application server to be connected to the internet. When internet access is enabled, you only need to specify a module. Other required modules are automatically selected.

An example (when logged on as root!):

```
# cpan
CPAN] install Parallel::ForkManager
```

(module File::Path will be automatically installed as dependency)

SPECIFIC INSTALLATION INSTRUCTIONS FOR DBD::ORACLE

To connect to the Oracle database, the Perl `DBI` module is used (independent database interface/access module), together with the Perl `DBD::Oracle` module (driver for Oracle databases).



Note: `DBD::Oracle` versions above 1.74 have an issue in combination with database PSU 19.15 (when installed on the Application Server). Symptom: Parallel compilation of Forms and Pro*C modules hangs in OHIPATCH, e.g in step 800. Stay on version 1.74 until further notice.

`DBD::Oracle` performs a *connection test* on the database.

To be able to successfully perform this test, as part of installing `DBD::Oracle`, perform following steps:

Prerequisites

First install the Perl modules (if not already installed!):

- Test::Simple
- Test::Warn
- Test::NoWarnings (1.83 only)
- Devel::Peek (1.83 only)
- DBI

Specific Set up for Root Account before Installing DBD::Oracle

As `root` user, now perform following steps to correctly indicate the Oracle OCI (Oracle Call Interface) environment for the connection test.

This is done by setting `$TWO_TASK` to a relevant database, and `$ORACLE_HOME` to the database (client) software:

```
. <path to $OZG_ADMIN>/ozg_init.env
. ozg_init.env $OZG_ORATAB_DB19
. ozg_init.env <ENV>
export ORACLE_USERID=<OHI account>/<password OHI account>
export LD_RUN_PATH=$ORACLE_HOME/lib
```

Example

(For this example, it is presumed `ozg_owner` is the OHI application owner account in the `PROD` database; the Oracle environment will therefore have to be set to `$OZG_ORATAB_DB19` and the OHI environment to `PROD`)

```
. /u01/app/oracle/product/OHI/admin/ozg_init.env
. ozg_init.env $OZG_ORATAB_DB19
. ozg_init.env PROD
export ORACLE_USERID=ozg_owner/ozg_owner
export LD_RUN_PATH=$ORACLE_HOME/lib
echo $TWO_TASK → PROD
echo $ORACLE_HOME →> /ohi/oraBase/product/db19c
```

32-/64-bit

Be sure to use the right Perl version; you should be using the 64-bit Oracle Database (Client) software and the 64-bit version of Perl.

The Perl version can be checked with file `/usr/bin/perl` or `perl -v` (or `-V`).

Install DBD::Oracle

Now install `DBD::Oracle` by following the standard installation procedure for Perl modules; see paragraph [Installation procedure](#).

DBD::Oracle 1.74

On Linux 7, the following error during the `make test` phase may be ignored:

```
t/56embbeded.....ok 2/5DBD::Oracle::db do failed: ORA-00600: Interne
foutcode, argumenten: [kothc_uc_md5:lxerr], [], [], [], [], [], []. (DBD
ERROR: OCISmtExecute) [for Statement "CREATE or replace TYPE
table_embeda_type as varray(10) of varchar(30) "] at t/56embbeded.t line 48.
```

DBD::Oracle 1.83

On Linux 8, the following errors during the make test phase may be ignored:

```
t/25plsql.t ..... 1/86
#   Failed test 'p1 ok'
#   at t/25plsql.t line 357.
#       got: ''
#   expected: 'Hello'

#   Failed test 'p2 ok'
#   at t/25plsql.t line 358.
#       got: ''
#   expected: 'Y'
# Looks like you failed 2 tests of 86.
t/25plsql.t ..... Dubious, test returned 2 (wstat 512, 0x200)
Failed 2/86 subtests
```

Change OZG_DBDHOME variable in ozg_init.env

After (re-)installing `DBD::Oracle` the environment variable `OZG_DBDHOME` exported in `ozg_init.env` must be set to the oratab variable of the database it has been built with.

For example, when built with the 19c version of the database, `ozg_init.env` should contain:

```
export OZG_DBDHOME=$OZG_ORATAB_DB19
```

TERM::READKEY ISSUE

When installing the mandatory Perl modules on a Linux server a problem can occur with installing module `Term::ReadKey`. A workaround for this problem is to download and install the precompiled package.

The package can be downloaded from:

<http://rpmfind.net/linux/rpm2html/search.php?query=perl-TermReadKey>

Linux 7: `perl-TermReadKey-2.30-3.el7.rf.x86_64.rpm`

Linux 8: `perl-TermReadKey-2.37-7.el8.x86_64.rpm`

or use yum and let it find the right version and solve dependencies:

```
yum install perl-TermReadKey
```

APPENDIX E - INSTALLATION CHECKLIST

Following is a *quick reference checklist* for installing an OHI Back Office environment. The hyperlinks refer to the relevant paragraphs in this document.

1 – Prerequisites

- [Check OHI Back Office certification](#)
- [Check hardware & software requirements](#)
- [Check OS requirements](#) (for both Database & Application Server)
- [Download all required software](#) (OS, Database, Application Server, OHI, Perl, Java, etc.)
- [Install & configure OS](#)
- [Install additional required Perl modules](#)

Validation

Validate OS requirements by starting Oracle Universal Installer; all requirements should have been met.

2 – Install Database

- Install Oracle Database Server
- *Install All Product Languages*
- Install Oracle Database Examples (formerly Companion)
- Install Oracle Database Server patch set
- Install Oracle Database Server interim patches
- Apply Oracle Database Server workarounds for open bugs
- Configure Oracle Database Server
- [Create an OHI Back Office database](#)

Validation

Use OEM and/or OPatch to see if all interim patches are installed.

Are all workarounds mentioned in the Certification notes implemented?

Check the Database languages.

```
. ozg_init.env <Net Service Name>
. ozg_init.env DB19
sqlplus sys as sysdba
```

Enter a command that generates a message, for example:

```
SQL> exec dbms_output.put_line(1)
```

This command returns the message

```
PL/SQL procedure successfully completed.
```

Check if this message is in the correct language.

Check if the NLS_NUMERIC_CHARACTERS is set correctly.

3 – Install Application Server

- Install WebLogic Server
- Install Oracle Fusion Middleware Forms and Reports Services
- Install Oracle Fusion Middleware Forms and Reports Services patchset
- Install Oracle Fusion Middleware Forms and Reports Services interim patches
- [Initial Configuration Forms and Reports services](#)
- Detailed Configuration of Forms Services

Validation

Check the connection to the database and the version of the database client.

```
. ozg_init.env $OZG_ORATAB_DB19
. ozg_init.env <Net Service Name>
sqlplus sys as sysdba
```

Check the version of import and export utility and the Pro*C compiler.

```
. ozg_init.env $OZG_ORATAB_DB19
. ozg_init.env <Net Service Name>
imp help=y      => check version number
exp help=y      => check version number
proc -version => check version number
```

Check the version of the Developer software.

Use the following command to ensure the Oracle*Forms software is valid and check the version.

```
$OZG_BASE/utils/OHI_CMD.pl frmcmp_batch.sh -help
```

Check if the NLS_NUMERIC_CHARACTERS is set correctly.

4 – Prepare OHI Back Office Installation

- Set up configuration templates (ozg_init.env)
- Set up directory structure

Validation

Check if Operating System specific settings are correct.

For Linux

```
. ozg_init.env $OZG_ORATAB_FRS122119
```

LD_LIBRARY_PATH must be set.

Check if the Perl installation software compiles without errors and check the Perl version.

```
. ozg_init.env $OZG_ORATAB_DB19
. ozg_init.env <Net Service Name>
perl -version
perl -c $OZG_BASE/utils/OHIPLIB.pm
perl -c $OZG_BASE/utils/OHIPATCH.pl
```

Check if the installation menu can be started.


```
$OZG_BASE/utils/OHIPATCH.pl <Net Service Name>
```

Fill the parameters and a release (release files must exist in \$OZG_PATCH)

If the menu options appear, the following is correct:

- Perl libraries
- Developer version
- database connection

5 – Install OHI Back Office

- Perform installation
- Configure the application
- Set up logo's
- Set up Forms Terminal Resource file

Validation

Test following URLs to check the valid configuration of the HTTP server.

```
http://<server>:<port>/forms/frmservlet?config=<env>  
http://<server>:<port>/OHI/<env>/bin/OZGIMG01.gif  
http://<server>:<port>/OHI/<env>/bin/OZGIMG02.gif  
http://<server>:<port>/OHI/admin/logo.gif
```

Start the application; use the URL:

```
http://<server>:<port>/forms/frmservlet?config=<env>
```

Now check:

- The images on the startform and Info screen;
- Icons on iconic toolbar;
- Database settings, FMW version, etc. in the Info Screen (Help -> Info + button "Session")
- Availability of release documentation => Double-click on release in the *Releases* screen;
- Availability of Online Help (F1 key);
- Correct language of Forms messages (for example in login screen);
- Running Batch scheduler;
- Run script => *Valideren Business Rules*;
- Check logfiles of scripts;
- Run *Object Validation* => Parameter = B (Both database and filesystem)
Ensure no errors are reported.

APPENDIX F – SETTING UP OEM METRIC FOR SQL PLAN MANAGEMENT

In this appendix we will explain how to create a (new) metric extension to check SYSAUX tablespace overrun for the SQL Plan Base Management usage.

- Log in Cloud Control
- Go to the main menu “Enterprise”, next “Monitoring”, next “Monitoring Extensions”.
- Click Create button.

The screenshot shows the Oracle Enterprise Manager Cloud Control 12c interface. The top navigation bar includes 'Enterprise', 'Targets', 'Favorites', and 'History'. The main heading is 'Metric Extensions'. Below this, there is a search bar and a 'Create...' button. A table at the bottom lists metric extensions with columns: Name, Target Type, Display Name, Version, Description, Deployed Targets, Monitoring Templates, Status, and Owner.

Name	Target Type	Display Name	Version	Description	Deployed Targets	Monitoring Templates	Status	Owner

- Enter the Display Name. In this example we used “SMB_SPACE_USAGE2”.
- For Adapter use “OS Command – Multiple columns”.
- Enter a description, optional.
- For “Collection Schedule” you can use “Data Collection” “Disabled” during tests phase and enable it later on.

ORACLE Enterprise Manager Cloud Control 12c Help Log Out

Metric Extensions

General Properties Adapter Columns Credentials Test Review

Edit SMB_SPACE_USAGE2 (ME\$SMB_SPACE_USAGE2) v1 : General Properties Back Step 1 of 6 Next Finish Cancel

Specify the basic properties for the metric extension.
The default collection can be overridden on a target instance basis in the target's Metric and Collection Settings page.

General Properties

Target Type Database Instance

Name ME\$ ME\$SMB_SPACE_USAGE2
A Metric Extension Name can only contain alpha-numeric characters and the following non leading special characters : ("_", ".", "-", "+")

* Display Name SMB_SPACE_USAGE2

* Adapter OS Command - Multiple Columns
Tokens OS command output using user-specified delimiter

Description SID Check alert log file for Sql Plan Management Base storage.

Collection Schedule

Data Collection Disabled Enabled

Use of Metric Data Alerting Only Alerting and Historical Trending

Upload Interval 1 Collections

Frequency Collection Frequency Weekly
Weekly On Sunday

Start Immediately after deployment Time

Advanced

- Click “Adapter” on the left top or “Next” on the right top.

ORACLE Enterprise Manager Cloud Control 12c Help Log Out

Metric Extensions

General Properties Adapter Columns Credentials Test Review

Edit SMB_SPACE_USAGE2 (ME\$SMB_SPACE_USAGE2) v1 : Adapter Back Step 2 of 6 Next Finish Cancel

Specify the adapter definition properties.
Note that connection/credential properties cannot be edited, although credentials may be overridden in Step 4 of this wizard.

Basic Properties

* Command /usr/bin/perl
Specify the command to execute. The command line is: Command + Script + Arguments

Script Check_SMB_space_usage2
Optionally specify a file to pass as an argument to the command.

Arguments %SID%
Additional Arguments are appended to the command line.

Delimiter |
Specify the delimiter used to tokenize the command output.

Starts With em_result=
Specify the starting string of metric result lines.

Advanced Properties

Upload Custom Files
You can upload custom files here, which will be packaged into the Metric Extension. Usually, these are custom scripts or binaries that you want to execute.

Add Upload

Filename	Size	Edit	Delete
Check_SMB_space_usage2	406		

OS Command - Multiple Columns

This adapter executes the specified command, and returns the command output, delimited by a specified string, as multiple columns.

For example, if the command output is: em_result=1|2|3 and the Delimiter is set as |, then three columns are populated with values 1,2,3 respectively.

Properties

- Command - The command to execute. For example, %perlBin%/perl. The complete command line will be constructed as: Command + Script + Arguments.
- Script - A script to pass to the command. For example, %scriptsDir%/myscript.pl. You can upload custom files to the agent, which will be accessible under the %scriptsDir% directory.
- Arguments - Additional arguments
- Delimiter - The string used to delimit the command output.
- Starts With - The starting string of metric result lines. For

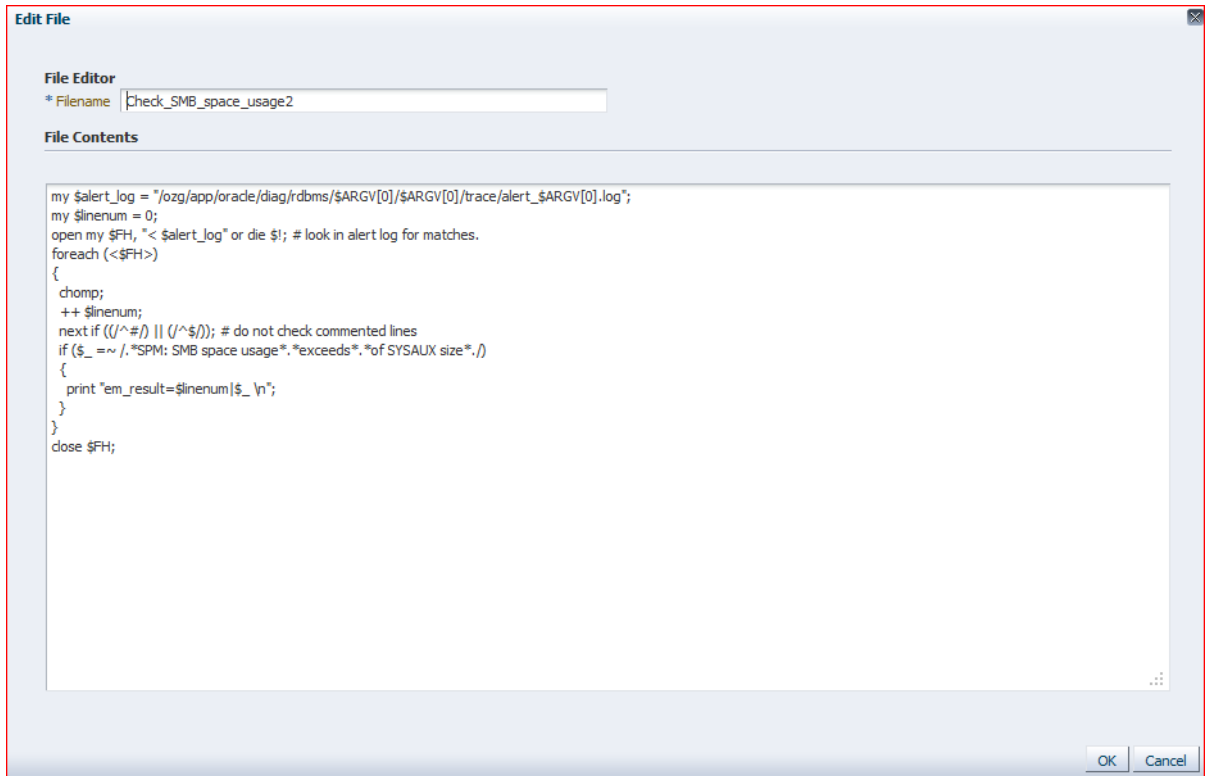
Available Variables

Variables can be used in adapter properties. Variable names are case-sensitive. To escape '%', use '%%'.

Name	Description
%perlBin%	location of perl binary
%scriptsDir%	directory where scripts are stored
%INSTANCE%	name of target instance
%TYPE%	target type
%DISPLAY_NAME%	display name of target instance
%TYPE_DISPLAY_NAME%	display name of target type
%OracleHome%	Oracle Home Path
%MachineName%	Listener Machine Name
%Port%	Port
%SID%	Database SID

- For this script Perl is used.
- Use a local Perl executable. The available Perl executable in the variable might not work as expected.
- Enter the script name, in this example “Check_SMB_space_usage2”.

- Enter arguments used in the script: “%SID%”
- Enter the script output delimiter: “|” (vertical bar).
- Specify the output starting string: “em_result=”
- Click “Add” button to add the script lines.



```
my $alert_log =
"/u01/app/oracle/diag/rdbms/$ARGV[0]/$ARGV[0]/trace/alert_$ARGV[0].log";
my $linenum = 0;
open my $FH, "< $alert_log" or die $!; # look in alert log for matches.
foreach (<$FH>)
{
    chomp;
    ++ $linenum;
    next if ((/^\#/) || (/^$/)); # do not check commented lines
    if ($_ =~ /. *SPM: SMB space usage*. *exceeds*. *of SYSAUX size*./)
    {
        print "em_result=$linenum|$_ \n";
    }
}
close $FH;
```

- Click “OK” to save the script.
- Click “Columns” on the top left or “Next” on the right.

ORACLE Enterprise Manager Cloud Control 12c Help Log Out

Metric Extensions

General Properties Adapter **Columns** Credentials Test Review

Edit SMB_SPACE_USAGE2 (ME\$SMB_SPACE_USAGE2) v1 : Columns Back Step 3 of 6 Next Finish Cancel

Metric columns are specified here.
The order of the metric columns matter, and it should match the order that they are returned from the adapter.

Columns

View Add Edit Delete Move Up Move Down

Name	Display Name	Column Type	Value Type	Alert Threshold		
				Comparison Operator	Warning	Critical
Linenum	Line_number	Key Column	Number			
line	Line	Data Column	String			

- Create columns as shown in the picture above.
- Click “Add” & “New Column” and enter the details
- Name: Linenum, Display Name: Line_number, Column Type: Key column, Value type: Number
- Name: Line, Display Name: Line, Column Type: Data column, Value type: String
- Click “Credentials” on the left top or “Next” on the top right.

ORACLE Enterprise Manager Cloud Control 12c Help Log Out

Metric Extensions

General Properties Adapter Columns **Credentials** Test Review

Edit SMB_SPACE_USAGE2 (ME\$SMB_SPACE_USAGE2) v1 : Credentials Back Step 4 of 6 Next Finish Cancel

You can override the default credentials with custom monitoring credential sets.
Custom monitoring sets can be created through the command line tool: `emcli create_credential_set`. Refer to the manual for additional details.

Credentials

* Host Credentials ☒ Use Default Monitoring Credentials ☐ Specify Credential Set

☒ No credential sets are available for Host Credentials. You can create custom monitoring credential through the command line tool `emcli create_credential_set`.
Specify the credential used to launch the OS command.

Input Credentials **Add**

Credential Reference	Credential Set	Edit	Delete

Pass additional credentials to the OS command via the standard input stream

- Use the default credentials or set other known credentials.
- Click “Test” on the top left or “Next” on the top right.

ORACLE Enterprise Manager Cloud Control 12c Help Log Out

Metric Extensions

General Properties Adapter Columns Credentials **Test** Review

Edit SMB_SPACE_USAGE2 (ME\$SMB_SPACE_USAGE2) v1 : Test Back Step 5 of 6 Next Finish Cancel

You can perform real-time metric evaluations here on specified test targets.
It is recommended that you test your metric extension here first before deploying to targets. Targets that you select need to be up in order for your test to succeed.

Test Targets

+ Add - Remove Run Test

Target Name	Target Type	Hostname	Current Status	Agent
tech.world	Database Instance	tech.world	Up	...

Test Results

Clear Results

Target Name	Line_number	Line	Error Message
tech.world			
	1	SPM: SMB space usage (15663104) exceeds 1.000000% of SYSALUX size (954204160).	
	2	SPM: SMB space usage (26804224) exceeds 1.000000% of SYSALUX size (954204160).	
	77	SPM: SMB space usage (26804224) exceeds 1.000000% of SYSALUX size (954204160).	

- Click “Add” to add a test environment.
- When a test environment is present you may run a test using the “Run Test” button.
- Click “Finish” when done.
- Next step is to deploy the metric and add more environments to it to be checked for.
- Also the Data Collection disabled in the beginning can be enabled and set.

ORACLE Enterprise Manager Cloud Control 12c Help Log Out

Metric Extensions

General Properties Adapter Columns Credentials Test **Review**

Edit SMB_SPACE_USAGE2 (ME\$SMB_SPACE_USAGE2) v1 : Review Back Step 6 of 6 Next Finish Cancel

Editing version 1 of metric extension ME\$SMB_SPACE_USAGE2 for target type Database Instance.

General Properties

Target Type: Database Instance
Name: ME\$SMB_SPACE_USAGE2
Display Name: SMB_SPACE_USAGE2
Adapter: OS Command - Multiple Columns
Description: SID Check alert log file for Sql Plan Management Base storage.
Version Comment:

Collection Schedule

Data Collection: Enabled
Collection Frequency: Weekly
Repeat Every: 15 Weekly
Use of Metric Data: Alerting and Historical Trending
Upload Interval: 1

Adapter Properties

Command: /usr/bin/perl
Script: Check_SMB_space_usage2
Arguments: %SID%
Delimiter: |
Starts With: em_result=

Advanced Properties

EM_TARGET_ADDRESS: (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=%MachineName%)(Port=%Port%))(CONNECT_DATA=(SID=%SID%)))
EM_TARGET_NAME: NAME
EM_TARGET_ORACLE_HOME: OracleHome
EM_TARGET_ROLE: Role

Custom Files

Check_SMB_space_usage2 (406 bytes)

Columns

Name	Display Name	Column Type	Value Type	Comparison Operator	Alert Threshold
					Warning Critical
Lineum	Line_number	Key Column	Number		
line	Line	Data Column	String		

Credentials

Host Credentials: Uses Monitoring Credentials of Target.

APPENDIX G – PROVIDING ACCESS TO JMS QUEUES

This Appendix describes how to provide access to standard OHI provided JMS payload queues from within the database and from within WebLogic. The queues themselves are created and maintained by the OHI installation software.

DATABASE ACCESS

A standard OHI database account needs to be created for accessing database queues with a JMS payload of Message Type 'TextMessage'. The account is optional and as such needs to be created manually if you want to use the JMS queues.

When this account is created it will receive enqueue and dequeue privileges on the JMS queues (the standard queue as well as the exception queue) when step 120 of OHIPATCH is run. So when you create this account please run OHIPATCH step 120 afterwards in order to make sure the relevant enqueue and dequeue privileges are granted.

The database account is needed to provide secure and limited access to the JMS queues. Both the normal and the exception queues can be accessed.

The account needs to be created to support two types of JMS queue usage:

1. The more secure solution for implementing web service calls to web services in the environment outside OHI Back Office, so for 'consuming' web services. These web services implement mostly services as offered by Vecozo, for example 'fraud checking', publishing the 'insured member' list, checking addresses, etc.
2. The functionality to publish events and messages within OHI Back Office on a JMS queue to process these events outside OHI.

The first usage is needed in almost each OHI Back Office production instance when this type of web service call technique is implemented. This technique replaces the callouts by means of using Java in the database and the use of the Database WebService callout utility, as used until OHI Release 10.20.1.0.0.

The second usage is offered since the 10.19.1.2 release and the usage is optional.

The account can be used for accessing the JMS queues from within WebLogic as foreign queues.

If you have decided you need this account in your OHI environment, use the commands below.

Beware, the username is fixed and needs to be OHI_JMS_QUEUE_USER.

```
Create user ohi_jms_queue_user
identified by &password_for_user
/

grant create session to ohi_jms_queue_user
/

grant alter session to ohi_jms_queue_user
/

grant execute on dbms_aqin to ohi_jms_queue_user
/

grant execute on dbms_aqjms to ohi_jms_queue_user
/
```

The minimum privileges are shown above. You may specify additional properties for default and temporary tablespace. To prevent service interruptions, you should prevent password aging

(it should never expire) or make sure a password change is implemented throughout the infrastructure chain when a new password needs to be set. You might also consider to create a logon trigger to limit the access to this account from specific machines or IP-addresses.

Please do not forget to run OHIPATCH step 120 afterwards.

When this account is present the queues can be accessed from external applications for dequeuing or enqueueing. This can be done by directly logging on to the database and using several API's as documented in the [Programmatic Interfaces](#) chapter of the Oracle Database Advanced Queuing User's Guide. The `dbms_aq pl/sql` package can be used. In Java, use either the standard JMS interface `javax.jms` package or the Oracle JMS interfaces as provided in the `oracle.jms` package.

Another option is to access the JMS queues from within a WebLogic domain, which will probably be the most common way to access the queues. This is described in the next paragraph.

WEBLOGIC ACCESS

Typically, a JMS queue is accessed from within a Java application using JNDI names. For the OHI provided JMS queues this is possible by defining them as Foreign Queues in WebLogic Server.

There are two options to do this:

- One of them is manually, typically used when some specific configuration or tuning is required.
- The other option is to use some template files to do this in an automated way, for most non-production environments this may be sufficient.

For both options it is assumed a WebLogic Server domain is already configured, with an Admin Server and at least a Managed Server, possibly in a cluster.

The instructions below describe how to access the queue `ALG_JMS_QUEUE`.

Manual Configuration

When you have specific requirements other than what is implemented in the automated way, we expected you to be familiar with the configuration aspects. We only describe high level steps here.

For more information please consult the Oracle WebLogic Server documentation manual "Oracle Fusion Middleware [Administering JMS Resources for Oracle WebLogic Server](#)".

You typically connect to the Admin Server of the WebLogic domain to implement the configuration by executing the actions below.

In the examples below an example naming is used that matches the naming as implemented by the automation script described in the next paragraph, when the example values over there are used. The naming convention used in the example values relies on a standard prefix 'OHI_JMS' and as environment name 'BDVM03'.

- Create a JDBC data source
 - Create a new generic data source (example name: `DS_OHI_JMS_BDVM03`), specify the JNDI name (example: `jdbc/DS_OHI_JMS_BDVM03`), the database type Oracle and a database driver (JDBC thin XA driver, probably for Service connections)

- Specify the connection info (database name, host name, port and OHI_JMS_QUEUE_USER for database user with its password and test the connection)
 - Target the JDBC data source
- Create JMS Server
 - Specify a name (example: OHI_JMS_BDVM03_Server) and a group (Global unless you want to restrict it)
 - None for Persistent Store
 - Target the JMS Server
- Create JMS Module
 - Create a System Module (example name: OHI_JMS_BDVM03_Module) with Global scope
 - Target the JMS Module
 - Do not yet add resources
- Create Foreign Server
 - Select the new System Module and create a new resource of type Foreign Server (example name: OHI_JMS_BDVM03_ForeignServer)
 - Target the Foreign Server
 - Edit the JNDI Properties and specify the JNDI name of the data source to use (example: datasource=jdbc/DS_OHI_JMS_BDVM03)
- Create Connection factory
 - Within the created Foreign Server create a new Connection Factory (example name: OHI_JMS_BDVM03_ConnectionFactory)
 - Specify a Local JNDI Name (example OHI_JMS_BDVM03_LOCAL_CF) and as Remote JNDI Name the exact string 'XAQueueConnectionFactory'
- Create JMS queue as Destination
 - Within the created Foreign Server select Destinations, choose to create a new one and specify a name (example: OHI_JMS_BDVM03_QUEUE)
 - Specify a local JNDI Name (OHI_JMS_BDVM03_QUEUE) and as Remote JNDI Name the exact string 'Queues/OZG_OWNER.ALG_JMS_QUEUE' (if your OHI BO table owner name is different from OZG_OWNER specify that name instead of OZG_OWNER)

Automated Configuration

In order to speed this up some helper scripts and a template for a property file are provided within the \$OZG_BASE/conf/Back-Office folder structure. These files can be copied and adapted to implement a customer specific configuration script.

In the Back-Office specific folder three files are provided:

1. ohi_jms_queue.properties.template
2. ohi_jms_queue.py
3. ohi_jms_queue.sh

The first file is a template file to specify property values that are used by the second file, a Python script used by the WebLogic Scripting Tool (WLST).

The third script is a shell script that can be started to call the WebLogic Scripting Tool with the Python script and the property file as arguments.

For this to work please implement the following steps:

1. Copy the files to a separate location.
2. Rename the `ohi_jms_queue.properties.template` file to the same file without the `.template` postfix (or make a copy of the template file and name it `ohi_jms_queue.properties`).
3. Adapt the properties of this `ohi_jms_queue.properties` file.
 - a. Admin url – describes the complete URL, example:
`admin.url=t3://server1.domain1.nl:9999`
 - b. Environment name, example:
`env.name=BDVM03`
 - c. WebLogic Domain for environment, example:
`env.domain=DOM1`
 - d. Type of Target: *Server* for Managed Server or *Cluster*, example:
`env.target.type=Server`
 - e. Name of Target, example:
`env.target.name=MS_BDVM03`
 - f. Accountname of data source user, this is a fixed value like below:
`datasource.user=OHI_JMS_QUEUE_USER`
 - g. Environment variable containing password of this user, named `OHI_ENV_DS_PWD` if you use the accompanying `.sh` script. In this way it is prevented a password is stored in the properties file, as this imposes an extra security risk. Example:
`datasource.pwd_osvar=OHI_ENV_DS_PWD`
 - h. The name of the AQ queue in the database, example:
`db.queue=ALG_JMS_QUEUE`
 - i. Schema name of OHI table owner (most often `OZG_OWNER`) in uppercase, example:
`datasource.ohi_table_owner=OZG_OWNER`
 - j. Database hostname for JDBC connection, example:
`datasource.dbhost=dbserver1.domain1.nl`
 - k. Oracle Listener port (1521 most common), example:
`datasource.dbport=1521`
 - l. Service name of the database service, example:
`datasource.dbservice=bdvm03`
4. Make sure you set the WebLogic environment variables in your terminal session so they reference the WebLogic installation you have planned to make the JMS queue available in. When this is the standard WebLogic environment you are using for the Forms installation for OHI Back Office you typically will run:
`. ozg_init.env FRS122119`
At this moment your WebLogic domain is the Forms domain and you need to change it to the domain you planned to expose the queue, the domain as specified by the `env.name` property in the `ohi_jms_queue.properties` file. Make sure you set the `DOMAIN_NAME` and `DOMAIN_HOME` correct (it is best to adapt your `ozg_init.env` file so you can specify this domain).
5. Make sure your current directory is where you copied the files to and edit `ohi_jms_queue.sh` to specify a data source password for environment variable

OHI_ENV_DS_PWD, only when the environment does not contain sensitive data (a test or training environment with no real production data). See the file itself.

The name of the variable should match with the property value for `datasource.pwd_osvar` in the passed properties file. When you do not set a password value the Python script will ask for one when the data source is created.

The shell script starts WLST with the `ohi_jms_queue.py` script. That script has an 'overwrite' option (-o) and a 'property file' option (-p followed by the name of the properties file). When the overwrite option is specified a possible existing data source and existing JMSmodule with a same name as specified by the properties will be deleted so they can be recreated based on the provided settings. Your customizations, when present in the configured WebLogic domain, will be lost.

6. Next run:

```
./ohi_jms_queue.sh
```

The script will ask for a WebLogic administrator account and password interactively. You can automate this and specify a credentials file like is described in the [paragraph](#) for automating the startup and shutdown of WebLogic and providing credentials.

When the Python script fails for some reason and you have resolved the cause it is advised to release the edit lock and undo the implemented changes by using the Admin console. As long as there are outstanding changes a next run of the script may fail.

When the behavior after some trial runs of the script are not as expected make sure you restart at least the Managed Server to clean up potential old cached settings.

Messages returned directly from the OHI provided template script `ohi_jms_queue.py` are prefixed with 'OHI: ' to distinguish them from feedback or error messages from the WLST commands itself.