ORACLE INSURANCE

Oracle Health Insurance Back Office

Security Guide

Version 1.20

Part number: G49637-01

January 15, 2026

CHANGE HISTORY

Release	Version	Changes
10.18.1.0.0	0.1	• Creation
10.18.1.0.0	0.2	• Processed some review input
10.18.1.0.0	1.0	• Publication
10.18.2.0.0	1.1	• Republished with different part nr.
10.19.1.0.0	1.2	• No changes. Republished with different part nr.
10.19.2.0.0	1.3	• Some changes in implementing database security paragraph.
10.20.1.0.0	1.4	• No changes, republished.
10.20.3.0.0	1.5	• Adapted for DB 19c and FMW 12.2.1.4 certification • Removed BICS • Added OHI JET • Incorporated changed SVC callouts • Textual changes/enhancements
10.20.8.0.0	1.6	• Added the implementation of Oracle Database Vault for OHI as specific security measure • Several minor textual adjustments
10.21.1.0.0	1.7	• No changes, republished with new part number.
10.21.2.0.0	1.8	• Added security paragraphs for the Service Consumer implementation through the use of queues and the optional use of the Oracle Service Bus
10.21.4.0.0	1.9	• Added paragraph 'Personal accounts for users of OHI Data Marts' • Added reference to 'Oracle Health Insurance Data Marts – Administrator Reference' • Paragraph 'BATCH processing account' modified • Added paragraphs for 'Scope of Security'and 'Security for non production environments'
10.21.7.0.0	1.10	• Complete review for actuality, many changes, minor and more major, in multiple paragraphs, emphasized at several locations security measures are not truly optional.
10.22.1.0.0	1.11	• No changes, republished with new part number.
10.22.3.0.0	1.12	• Added paragraph for Java Heap Inspection for SVL, HSL and PSL • Added reference to ADW for OHI Data Marts database
10.23.1.0.0	1.13	• No changes, republished with a new part number.
10.23.7.0.0	1.14	• Added sections for Oracle Setup Manager and VECOZO message service
10.24.1.0.0	1.15	• No changes, republished with new part number.
10.24.5.0.0	1.16	• Some textual tightening and some corrections in Appendix B, regarding Oracle Setup Manager.
10.25.1.0.0	1.17	• Republished with new part number. • Added a basic security consideration • Added strong recommendation to use webservice UUID / ID compatibility for a limited time only. • Some small textual changes.
10.25.2.0.0	1.18	• For HSL webservices the strongly preferred and default authentication and authorization has changed from Basic Authentication to OAuth2.0. Basic Authentication is still supported for backward compatibility but is deprecated as it is considered less secure.
10.25.4.0.0	1.19	• Added bullet regarding importance of individual authentication of regular and each separate check service in client-side authentication paragraph for HSL services.
10.26.1.0.0	1.20	• No changes, republished with new part number.

RELATED DOCUMENTS

A reference in the text (**doc[x]**) is a reference to another document about a subject that is related to this document.
Below is a list of related documents:

Doc[1]	Oracle Health Insurance Back Office - Installation, Configuration and DBA Manual (docs.oracle.com)
Doc[2]	Oracle Health Insurance Back Office HTTP Service Layer - Installation & Configuration Manual (docs.oracle.com)
Doc[3]	Oracle Health Insurance Back Office – SOAP Service Layer Installation & Configuration Manual (docs.oracle.com)
Doc[4]	Oracle Health Insurance Back Office – Reading, Writing and Authorizing (docs.oracle.com)
Doc[5]	Oracle Health Insurance Back Office - Custom Development (docs.oracle.com)
Doc[6]	Oracle Health Insurance Back Office - Object Authorization ((docs.oracle.com)
Doc[7]	Oracle Health Insurance Data Marts - Administrator Reference
Doc[8]	Oracle Health Insurance Back Office - Data Management Technical Reference (docs.oracle.com)
Doc[9]	Oracle Health Insurance Back Office HTTP Service Layer - User Manual (docs.oracle.com)

Unless otherwise indicated, these documents can be downloaded from docs.oracle.com.

Contents

1 OHI Back Office Security Overview

This chapter provides an overview of OHI Back Office security.

1.1 Scope of Security

In this document risk considerations are given and non open-ended recommendations are provided for improving the security of your OHI Back Office application landscape.

As mentioned, these recommendations are not open-ended: they should be applied whenever the signalled risk is acknowledged to be a realistic weakness for your organisation! Ignorance should be avoided when weighing these risks, make sure the knowledgeable people are involved and know what their responsibility is when deciding whether a recommendation is implemented or not. It should be weighed that although many of the recommendations are technically optional, they are in fact often not, given your responsibilities and applicable legislation. Each OHI customer should carefully determine whether a technical option is truly optional in the real life situation of today. Make sure you document your arguments when deciding not to implement specific recommendations.

Reason for the optionality of some of the recommendations is the additional product license that needs to be acquired (increased security has a price). OHI cannot enforce purchasing additional product options and as such offers optional implementation of these options. In certain Cloud deployment models these options are included in the price and there is probably no good remaining reason to not implement the options.

Many of the recommendations in this document apply to all of your OHI Back Office environments, so also to development, training and test environments. The scope should certainly not be limited to your OHI Back Office production environment. This is mentioned explicitly here, as non-production environments may be misused too.

Some misuse examples of a non-production environment are:

- A potential attacker might use a non-production environment to acquire knowledge about how the application is used and configured, within your organisation and landscape, to prepare attacks on the actual production environment.

- Some of your non-production environments may be a cloned copy of production and as such can be as vulnerable as your production environment for leaking the highly sensitive data stored. This may even apply when the data in the clone has been masked. Although the data is masked, the data may still be attributable to real-life persons by someone who has prior knowledge about the individual, for example based on rare medical treatments.

- A non-production environment may provide more details about application interaction and attack surface when the environment is less well protected and monitored. These details may help in creating an attack plan for the actual production environment.

Please make sure you work through this document and determine an action plan for any follow up that is needed or document why specific recommendations are not being implemented.

1.2 Basic Security Considerations

The following principles are fundamental to using any application securely:

- Keep software up to date at all levels (firmware, operating system, network, technology stack and application). This includes the latest supported product releases and any security related patches that apply to it. Typically installation of cumulative patch sets / release updates, containing fixes for detected weaknesses, and which are released periodically, should be part of your regular maintenance activities.

- Limit privileges as much as possible. Users should be given only the least access privilege necessary to perform their work. User privileges should be reviewed periodically, for example quarterly or monthly, to determine relevance to their current work position and requirements.

- Monitor system activity. Establish who should access which system components, and how often, and monitor those components to create a 'baseline' of regular usage patterns, with a verification of being representative for a baseline. Investigate unusual usage patterns.

- Install software securely. For example, use firewalls, secure protocols using TLS (SSL), and secure passwords. See "Performing a Secure OHI Back Office Installation" for more information.

- Learn about and use the OHI Back Office security features. See "Implementing OHI Back Office Security" for more information.

- Use secure development practices. For example, take advantage of existing database security functionality instead of creating your own application security. See "Security Considerations for Developers" for more information.

- Keep up to date on security information. Oracle regularly issues security-related patch updates and security alerts. You must install all security patches as soon as possible. See the "Critical Patch Updates and Security Alerts" web site: https://www.oracle.com/security-alerts/

- Do follow up as much recommendations as possible to implement a defence-in-depth strategy, so a bad actor who breaches one layer of defence may still be contained in other security layers. Defence in depth involves integrating multiple layers of defence.

1.3 Overview of OHI Back Office Security

The OHI Back Office application manages the health insurance administration of a health insurance provider.

The OHI Back Office database contains the policy records and claims data of the individual insurance members, including all financial details involved with processing these policies and claims. The larger OHI Back Office implementations have a population of 2 to more than 4 million members.

The OHI Data Marts database contains data derived from the OHI Back Office database, optimized for analysis and reporting by using a Star Scheme model with Fact and Dimension tables, and Data Marts for specific subjects.

OHI Back Office uses a multi-tier architecture based on Oracle technology. OHI Back Office components are installed in:

- Application tier: Oracle Forms application for Back Office users, batch scheduler and script definitions, web services, and OHI JET user interface components.
- Database tier: OHI Back Office database and (optional) OHI Data Marts data warehouse.

The architecture diagram below shows how the components are logically working together. It does not show the infrastructure components like firewalls, load balancers, etc.

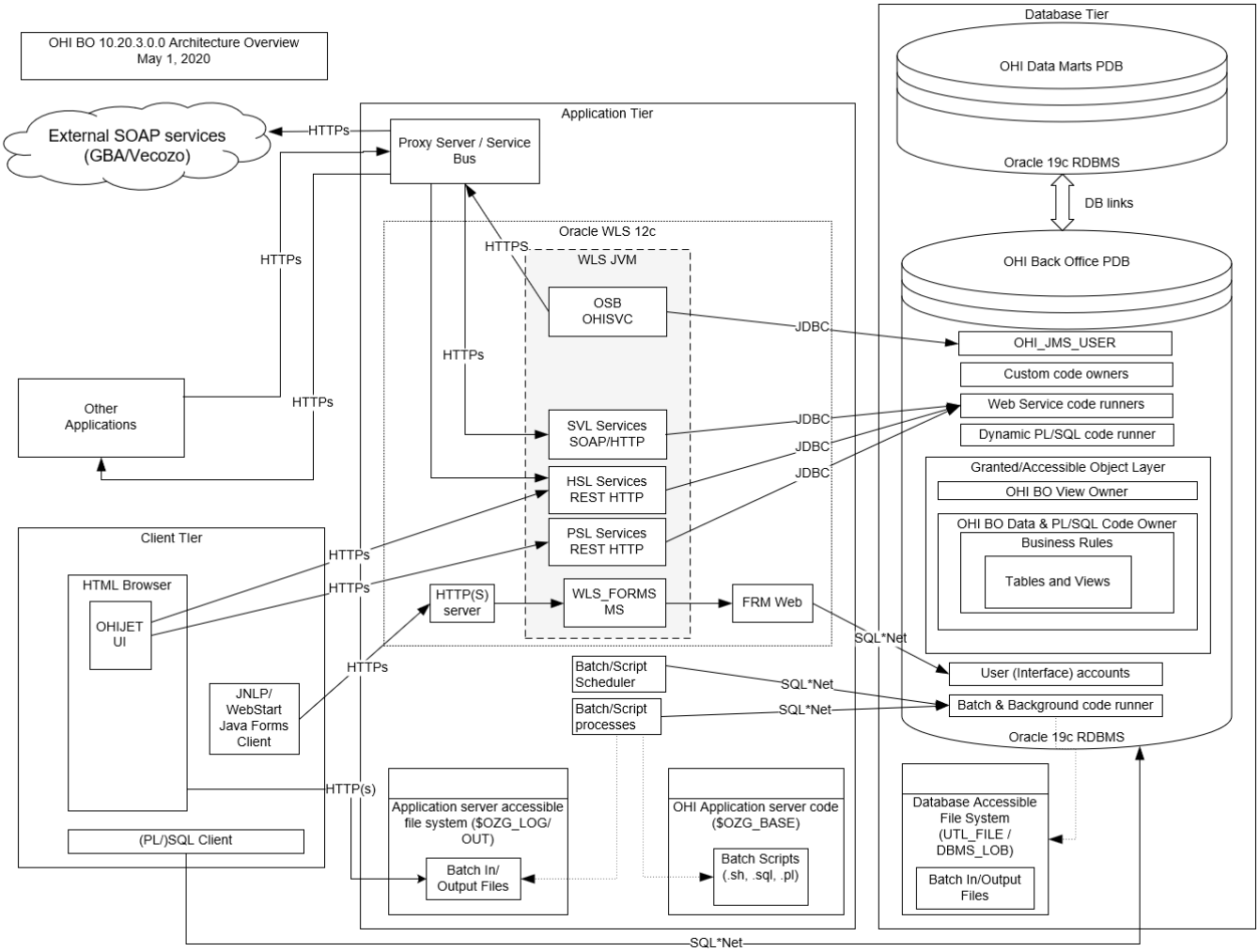


Figure 1 (MME 116285)

The Oracle Forms based user interface is accessed by Back Office users on the health insurance provider's intranet. Each back office user may have an Oracle database account (when SSO is not used), without other privileges than creating a session, to log in to the Forms application.

The OHI Back Office web services are used to integrate with existing applications or provide a back end to bespoke self-service portals for insurance members. The web services use at least Basic Authentication over SSL. OAuth2.0 is supported and recommended for REST services.

Most OHI Back Office customers use an additional custom database schema for custom developed software to complement the OHI Back Office functionality through custom PL/SQL packages. The OHI Back Office application is designed to always maintain data integrity and consistency, regardless of the way data accessed, even when directly through custom SQL.

On the application server, Oracle Fusion Middleware (FMW) is used to:

- deploy the forms application (through Oracle Forms Services);
- deploy OHI Back Office web services through Weblogic Server (WLS);
- deploy the OHI JET user interface application on WebLogic Server (WLS);
- provide static HTML through the HTTP server (OHS);
- optionally deploy an OHI standard Oracle Service Bus (OSB) project to implement calls to external web services

Wherever the term 'WLS' is used, it should refer to the Weblogic Server instance of Oracle Fusion Middleware.

1.4 Understanding the OHI Back Office Environment

A large amount of confidential information owned by healthcare business relations is stored in the OHI Back Office and OHI Data Marts databases. Examples of this are personal data and, on a higher level, medical treatment history. This data must not only be correctly entered and stored but because of its confidential nature it must also be properly safeguarded.

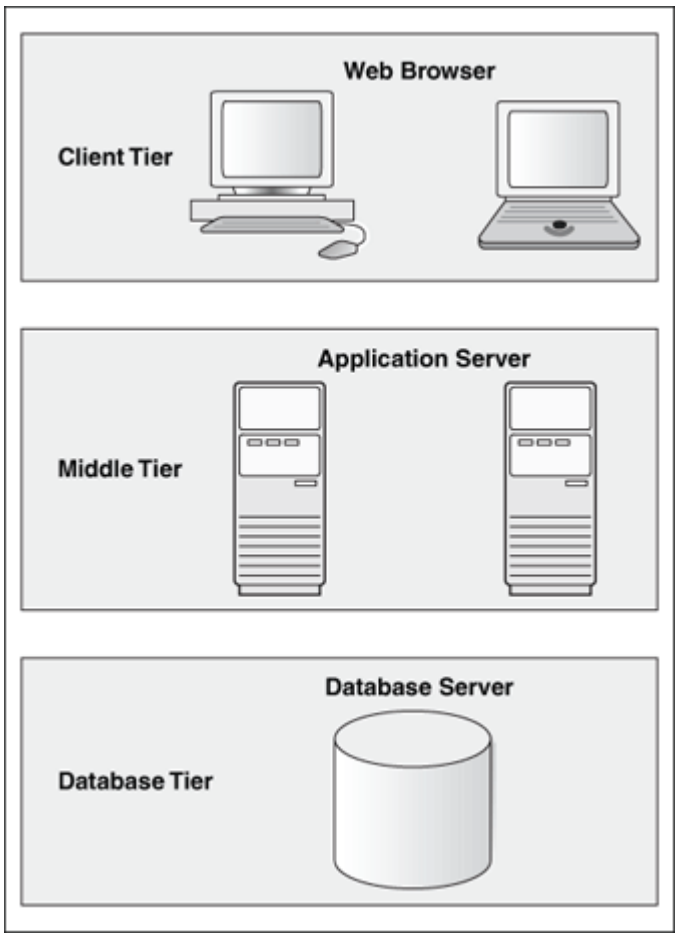
When planning your OHI Back Office implementation, consider the following:

- Which resources need to be protected?
    - You need to protect personally identifiable information (PI) such as name, social security number, date and place of birth, mother's maiden name.
    - You need to protect any confidential information that can be linked to an individual, such as financial information or claims history. This data may contain PHI (personal health related information)
    - You need to protect system components from being disabled by external attacks or intentional system overloads.
- Who are you protecting data from?
  For example, member data should be handled only by authorized OHI Back Office users. If you are deploying a self-service portal you should ensure that member data cannot be accessed by care providers, brokers or other members.
- What will happen if protections on a strategic resource fail?
  A worst case scenario is when PI or PHI data of insurance members is stolen or made public. Apart from the embarrassment this might trigger an investigation from the national GDPR (General Data Protection Regulation) watchdog potentially resulting in severe sanctions and high fines.
  Another risk is financial damage resulting from fraud committed by an external entity or by OHI Back Office users or administrators. These users and administrators may pose a higher risk because they already have some form of access to the application and the data.
  Last but not least failing protection may lead to some form of overloading the system, a kind of 'denial of service' attack, which may result in the application not being available and disrupting your processes.

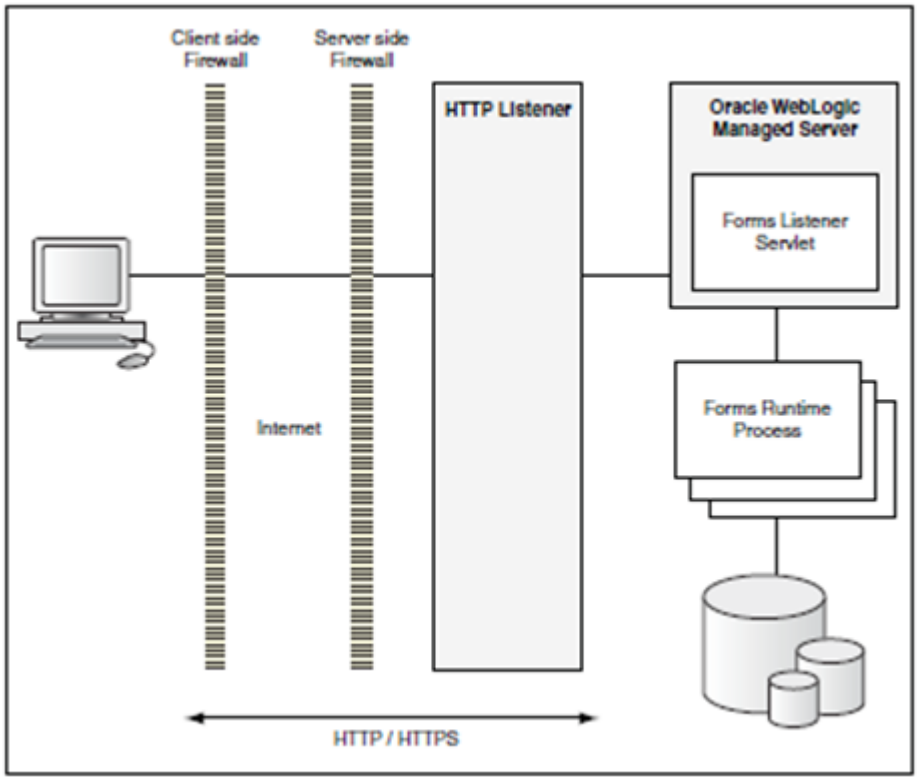
1.5 Recommended Deployment Configurations

1.5.1 Forms Services Deployment

If the OHI Back Office Forms GUI is only used by back office users on the intranet, the following set up would be adequate:



If the Forms GUI should be used over the internet (for example by a remote call center) HTTPS and appropriate firewall protection should be used to provide a minimum level of security:



However, direct Internet access for back office users should and may not be offered as it is too vulnerable for all kind of attacks. A connection over VPN provides much better protection against unauthorized access or overload of the application and should at least be used.

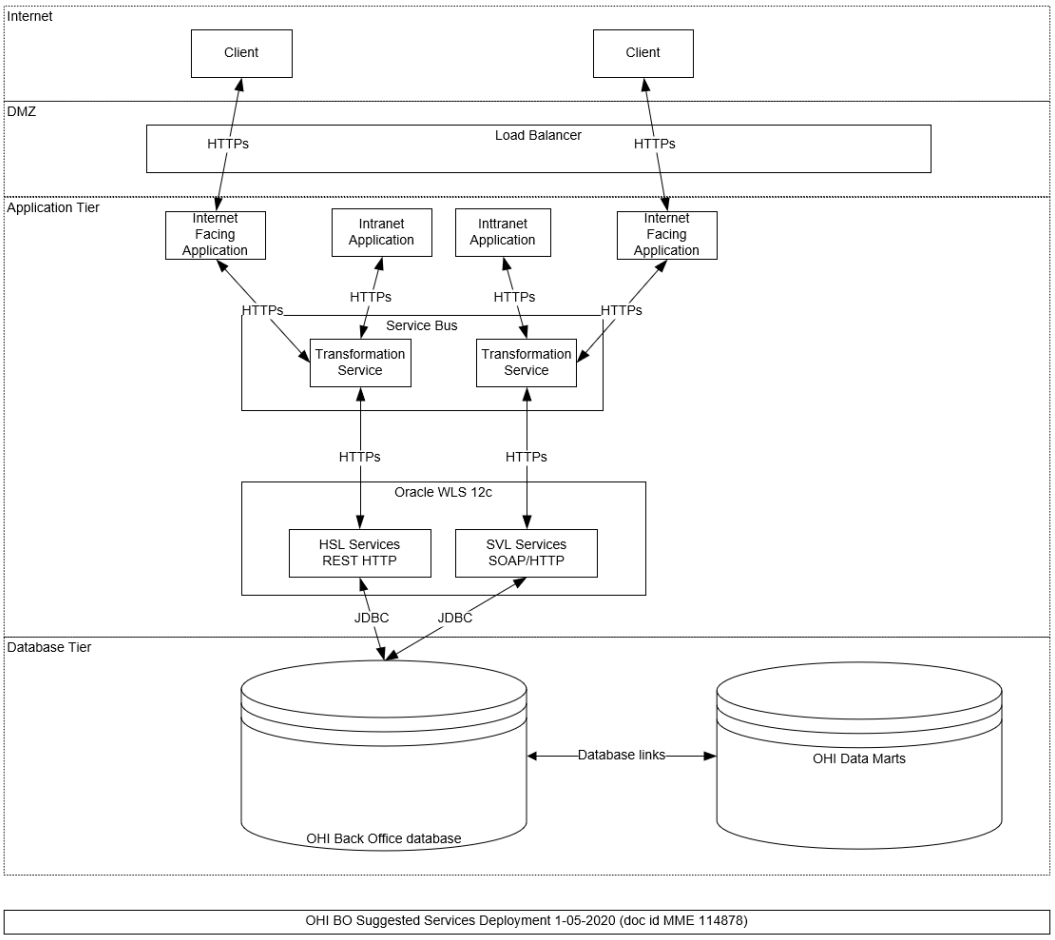
1.5.2 Web Services Deployment

When deploying OHI Back Office web services, take note of the following:

- Make a security risk assessment and access requirements list for each client application invoking OHI Back Office services and take appropriate action to mitigate these risks and provide only the required privileges.
- The OHI Back Office web services are meant as application interfaces and are not end user agnostic and as such do not offer any form of end user bound operation or data authorization. Make sure you only authorize these operations that are needed towards the calling application and implement an additional data and operation authorization structure within your calling application.
- The application server to which the web services are deployed should never be directly accessible from the internet, only access from trusted internal applications may be allowed.
- Creating a service hub with proxy services to call the OHI Back Office services provides a single point of definition for your client applications. The use of at least Basic Authentication of the HSL and SVL services would then also be restricted to the proxy services. Still these proxy services may not be accessible directly from the internet.
- The HSL services …. TODO
- If access to OHI Back Office web services is needed in some way indirectly over the internet, make sure you implement an application layer that

provides sufficient authentication and authorization towards the calling application functionality. This means that any form of operation or data authorization needs to be present in your calling application. Any other form of indirect access through the internet is strongly disapproved.

A suggested deployment is shown below:



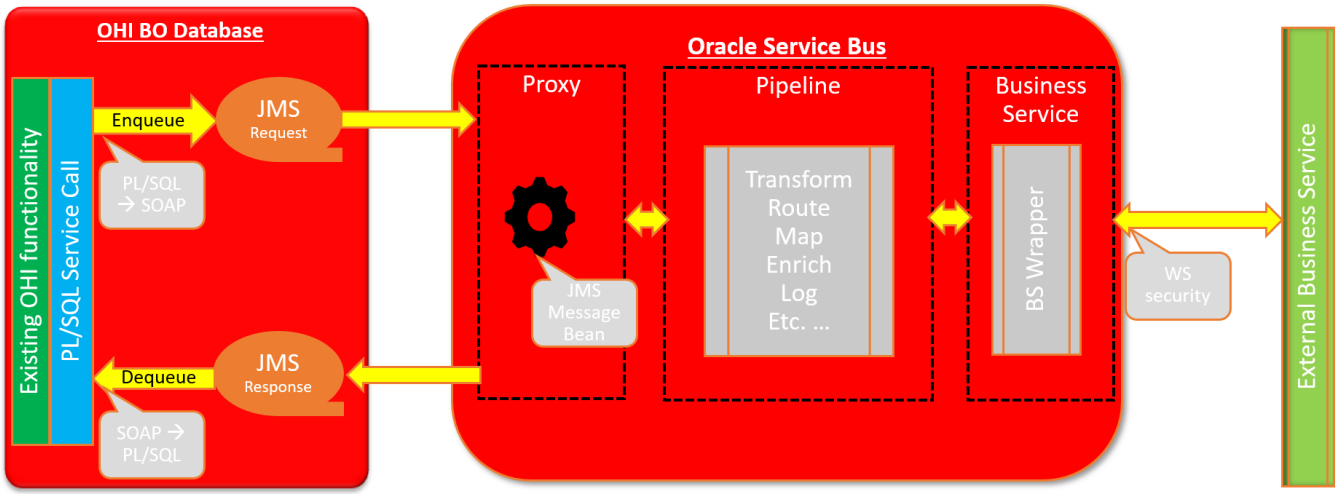
OHI BO Suggested Services Deployment 1-05-2020 (doc id MME 114878)

1.5.3 Service Consumer Deployment

OHI offers an architectural component that implements a synchronous callout to external SOAP based web services from within the database. This is referred to as the 'Service Consumer' implementation.

It offers a PL/SQL interface within the database to interact with the external web service and uses Advanced Queueing in the database (exposed as JMS Queues outside the database) as a pure technical implementation to provide this synchronous (!) interface.

The high level architecture is shown below.



Two AQ based JMS payload queues are used for communicating the request and response message with the application server. The application server passes the request message to the external web service by executing indirect calls and passing back the response message into the database by means of the response queue.

The functionality on the application server may be implemented by means of a custom solution or by deploying an OHI delivered Oracle Service Bus (OSB) project.

When configuring and deploying this callout functionality, take note of the following:

- Make sure the AQ based JMS payload queues in the database are accessed by means of a connection pool in WebLogic Server for which the credentials are known on a need to know basis.
- Securely configure the WebLogic environment that provides access to the AQ queues, exposed as JMS queues.
- Make an assessment of the security measures required for the message processing application that implements the actual calls to the external web services.
    - When using the standard OHI provided OSB project, implement security measures for a secure OSB configuration with the help of the OSB security documentation.
    - When you prefer to use a custom solution make sure the risks are assessed including their mitigation.
- Implement a proxy layer that implements secure calls to the external web services and that requires authentication for the calling client processes.

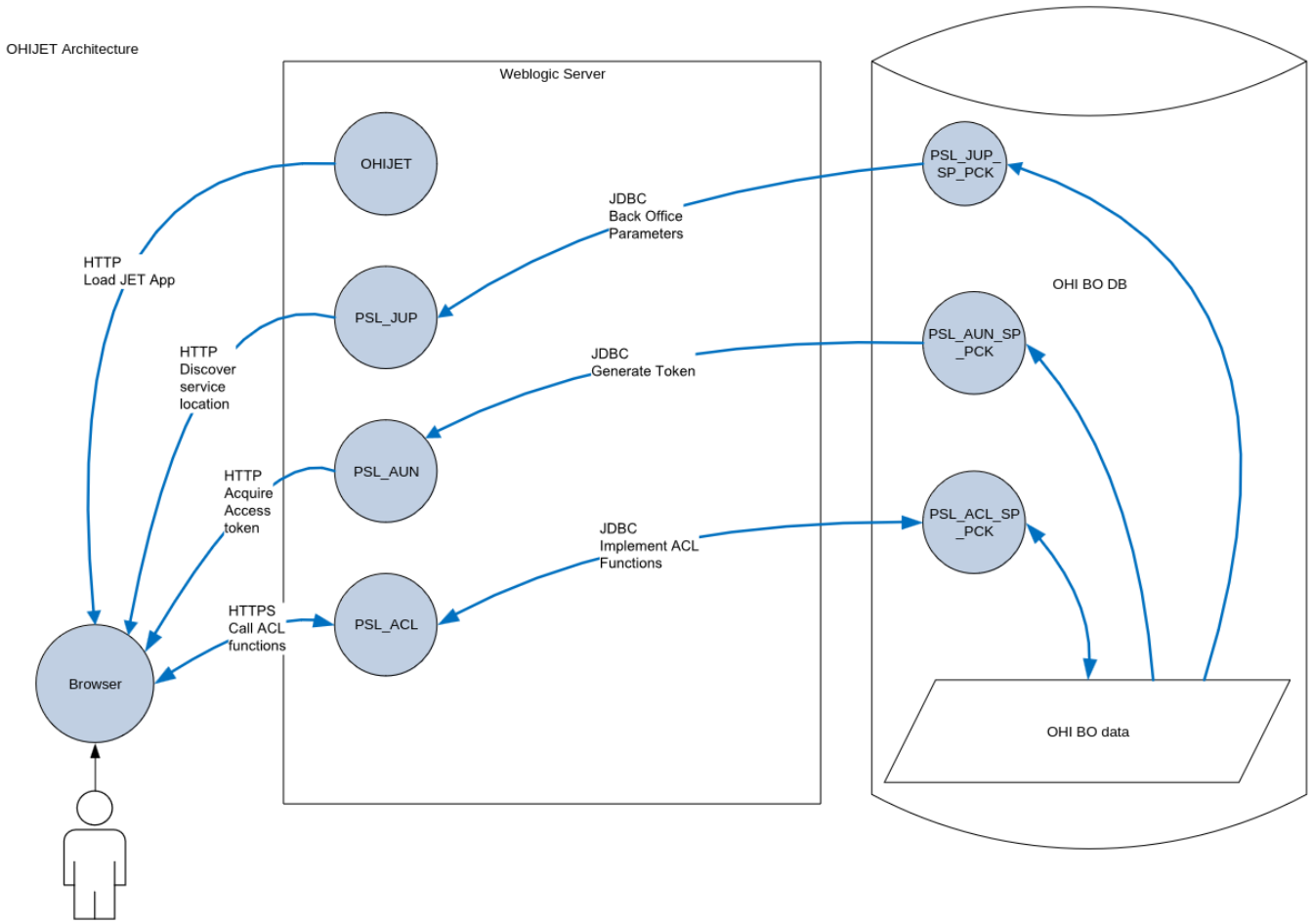
1.5.4 OHI Back Office JET Deployment

For the OHI JET based user interface a set of 'private' HTTP RESTful web services is used, referenced as the 'Private Service Layer', in short PSL.

It uses an OAuth2.0 based authentication and authorization architecture shown in the following diagram.

The technology used is identical as used in the HTTP RESTful Service Layer, referenced in short as HSL. The main difference is that for the PSL implementation, OAuth2.0 is used by definition. For the rest of this document this means that all security considerations that apply for HSL services also are applicable for PSL services.

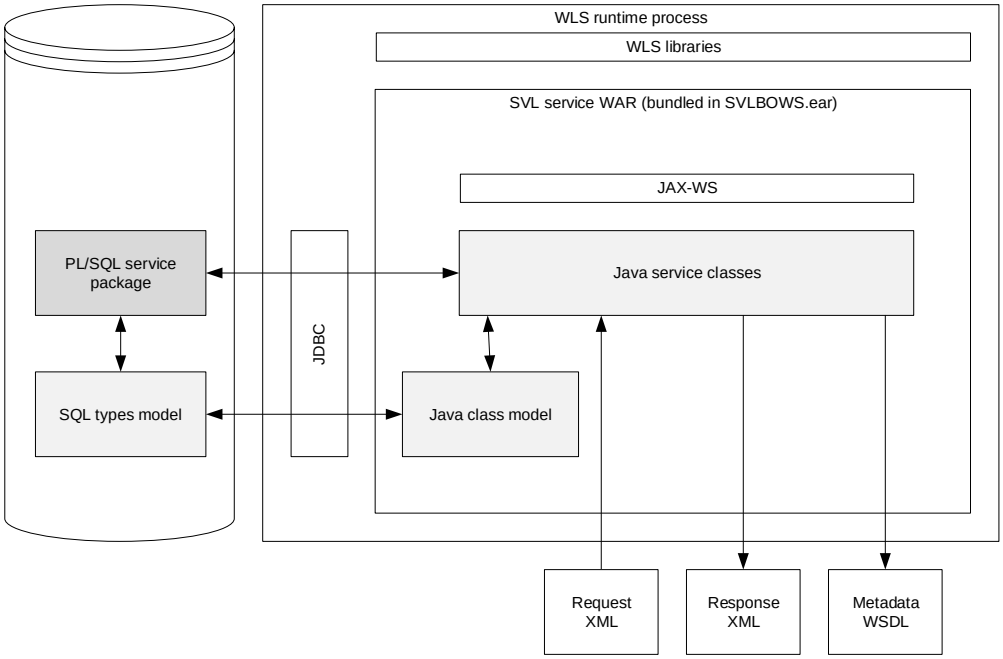
The PSL services are solely intended for service calls from the OHI JET application, no other usage is allowed or supported.



1.6 SVL Services Security

The (Business) Services Layer (SVL) contains synchronous SOAP/HTTP web services to retrieve and manipulate OHI Back Office reference data, such as creating and updating insurance policies, contracts with care providers etc.

The application architecture is outlined below:



The SVL services are deployed to Weblogic Application Server (WLS).

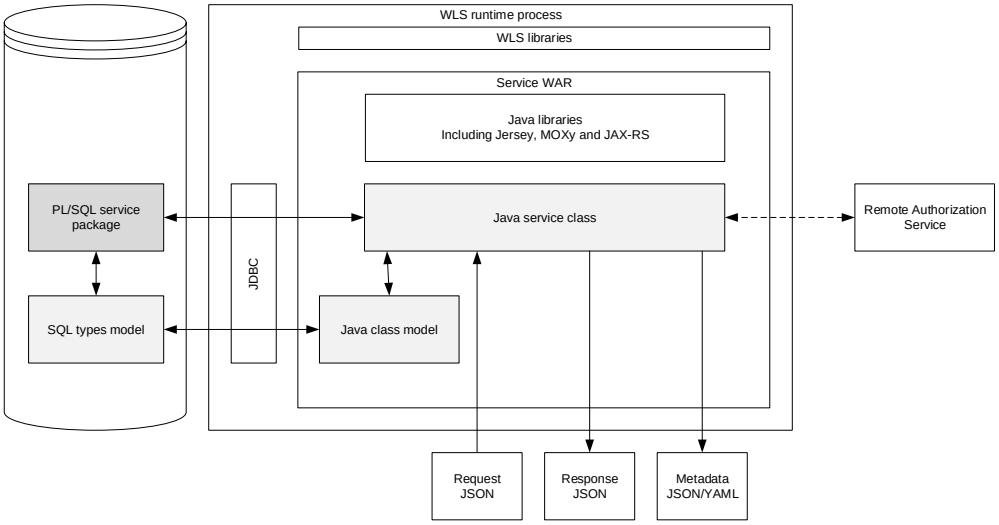
The default client-side security mechanism for SVL services is Basic Authentication over SSL. The default mechanism can be overruled with WLS policies and roles.

The database connection is set up through a data source managed by WLS.

1.7 HSL Services Security

The OHI Back Office HTTP Service Layer (HSL services) are synchronous RESTful services to support use cases, typical tasks performed by for example self-service users (insured members, care providers etc).

The application architecture is shown below:



The HSL services are deployed to WLS.

Default security is OAuth2.0 over SSL. Basic Authentication is still supported. To create the call to the remote Authorization Server it is necessary to add configuration data to the HSL properties file which is read at startup and resides on the application server.

The database connection is set up using a data source managed by WLS.

By default, technical error messages are suppressed in the HTTP response and appear only in the application log.

1.8 Service Consumer Security

For some processing work, interactively or by background processing, callouts to external (web) services are required from within the database, as all application logic is implemented in the database. These calls are always 'wrapped' in a web service specific pl/sql package.

Calling remote web services from within the database is implemented by means of AQ queues to exchange the request and corresponding response messages with an application that implements the actual web service calls, preferrably an OHI delivered OSB project which implements the calls instantaneously.

A WebLogic environment is needed to access the database based Advanced Queuing queues by means of a connection pool in order to make them accessible as JMS queues. The connection to the database uses an account that only provides access to the database queues and the JMS queues may only be accessed by authenticated WebLogic users.

The application that implements the actual calls should apply validation on the request and response messages, to protect against XML injection.

The web services should only be called over SSL, preferably using a VPN connection and should always require authentication with a secure password store.

The OHI delivered OSB project can be used to implement such functionality.

1.9 VECOZO message-based service (BRS) Security

OHI BO supports the VECOZO message-based service (BRS). Incoming Web Service messages are placed in a dedicated message table in the database to be processed by the Autonomous Processing Framework (AVF).

This opens up the possibility to overload the WebLogic server and especially the AVF capacity by sending large numbers of messages with large payloads.

Overloading of the WebLogic server can be prevented by using a Work Manager. This can be defined with standard WebLogic functionality and linked to the deployed application module OHIBOWebserviceVecozoBerichtuitwisseling. The Work Manager can use minimum and maximum values for execution threads or "fair use" percentages to limit the processing capacity for the deployed application module.

Overloading of the database can be prevented by lowering the priority of the AVF event definitions related to the VECOZO message processing. DBA's or functional managers should use the 'event definition settings' for AVF processing to set the priority to low, and keep the subpriority as low as is acceptable for a reasonable throughput time.

1.10 Background Processing Security

Most of the work in OHI Back Office is performed through background processing.

An example of the diversity of these tasks:

- Reading claims files into the database (SQL*Loader and SQL*Plus)

- Claims processing including generating payments (SQL*Plus)
- Producing XML output files for further processing in other applications (SQL*Plus)

These background processes are started and monitored by the OHI Back Office Batch Scheduler. This Pro*C program runs on the application server using a customer configurable 'batch' database account. This 'batch' database account has received a very limited set of object privileges necessary to monitor and start already earlier requested processes. No other application objects are accessible by this account.

The batch scheduler and batch processes use a Secure External Password Store (SEPS) to connect to the OHI Back Office database.

The batch scheduler starts and monitors a set of standard background maintenance processes, through the use of the database scheduler. Also the processes for the Autonomous Processing Framework (AVF) and the Business Event Processing Framework (BEF) are started and stopped when the batch scheduler is started and stopped. Next to that it starts and monitors individual batch script requests.

The invidual batch scripts write some overall progress output and technical logging information to OS files which can be viewed using the Forms application by the invoker of the batch script. Functional output is mostly written to database directories or separate output files on the application server and error messages are stored in the database.

The batch scheduler can be started and stopped from the command line by a technical application administrator on the application server it runs on and may be automatically started during the boot sequence of this server. A second application server can start the batch scheduler process to act as standby scheduler process for when the primary application server fails.

Access to the OS 'batch' account, the associated 'batch' database account and the SEPS file should be limited to technical application administrators only.

1.11 Operating System Security

A typical OHI Back Office production environment consists of

- One or more database servers with Oracle 19c database running the OHI Back Office OLTP database and the OHI Data Marts data warehouse database. The OHI Data Marts database can be located 'on premises' or in an Autonomous Data Warehouse (ADW) installation in the Oracle Cloud.
- An application server running OHI Back Office background processing (at OS level) and Oracle Fusion Middleware. FMW provides Oracle Forms Services for the OHI Back Office Forms GUI and Oracle WebLogic 12c for the OHI Back Office web services and the OHI specific OSB project for implementing calls to external web services.

Note that:

- The .out and .log OS files created by the OHI Back Office batch scripts are accessed over HTTP (triggered by calling the Oracle Forms 'show_document' builtin) and as such are publicly accessible unless configured differently. For most processes these files offer mainly some overall summary result and/or error information.

- OS level access to the database server should be limited to system administrators and database administrators only.
- OS level access to the application server should be limited to system administrators and (technical) application administrators only.
- Access to the SEPS file(s) on the application server should be limited to the file owner (read/write) and the OS user 'batch' (read-only).
- sFTP or a similar mechanism should be used to upload ASCII and XML files to the strictly authorized upload location, accessible by the application server, to process these by the OHI upload processes.
- The application servers and the database servers should never be accessible from the Internet.
- Access to the database servers from within the intranet should be limited where possible and audited. For regular users no direct database access is required, all user interface access is indirectly through user interface functionality running on the application server(s).

See the following documents:

- Guide to the Secure Configuration of Red Hat Enterprise Linux 7
- Hardening Tips for the Red Hat Enterprise Linux 7
- Oracle Linux 7: Security Guide

1.12 Oracle Database Security

All OHI Back Office data are stored in a single schema in an Oracle 19c pluggable database for optimal data consistency and reliability. Optionally this data is 'replicated' by a star-model in another schema in the OHI Data Marts data warehouse database.

A robust business rule layer, implemented directly in the database, as close as possible to the data, by means of PL/SQL, declarative constraints and database triggers, protects the integrity and consistency of OHI Back Office data.

Role-based security is implemented to grant the required set of privileges to authorized users. Users of the Forms GUI, who will use their personal dedicated database sessions, are dynamically granted database object privileges through a role only for the duration of the database session. For OHI JET users no direct database access is offered, all database access is executed via web service operations.

1.12.1 Separation of duties

Privileged database administrator accounts have very powerful privileges. In the best case personal accounts are created for each administrator but even than the generic accounts (like SYS and SYSTEM) remain present for persons who need to know the credentials to access these.

These powerful administrative accounts are able to:

- Create, drop or reset the password of any other account and grant that account similar powerful privileges (which enables identity theft or change).

- Access and manipulate the definition of any OHI database object as well as the contents of any OHI table. Meaning that all sensitive and financial data is fully exposed and vulnerable for any unauthorized change.

So anyone who wants to, or is forced to misuse these privileges is fully capable to do so and might even setup a cover-up to hide these actions or delete any traces.

The only reliable way to prevent this is to clearly implement a separation of duties way of working between these required administrative accounts and the accounts to manage users, their user privileges and theur access to OHI object data. Configuring, activating and implementing the database option 'Database Vault' is the only way to do this inside the Oracle database. Note that Database Vault is a licensed database product option.

To enable this necessary 'separation of duties' an OHI specific Database Vault 'realm' implementation is offered, for OHI Back Office as well as for OHI Data Marts, to create an additional protection layer for the OHI database objects, preventing the privileged database administrator accounts can access or manipulate the OHI database objects.

This OHI specific Database Vault realm, though technical optional, should be deployed to offer a better protection against data leaks caused by misuse of privileged accounts and should be considered as a 'must have' for hardening the OHI database.

1.12.2 Prevent unauthorized data access

All OHI data is stored in an Oracle database. Normally access is strictly authorized by means of database accounts and access privileges that are applied. However, the Oracle database stores the data in 'tablespaces' and the actual storage is implemented by using 'datafiles'. When someone gets on a low level access to (a copy of) these datafiles the data in it can be made accessible despite not having access to the database itself. Knowledgeable specialists may directly interpret the raw data in the datafiles or create a new database using these datafiles for which they can create their own account definitions. In this way an 'attacker' can get unauthorized data access.

To prevent against such a standard threat the database offers the option to encrypt all data in the datafiles through the use of Transparant Database Encryption (TDE), applied at the tablespace level. This prevents access to the actual data when some (backup) copy of the database gets into 'wrong hands'. Note that TDE is one of functionalities offered by the licensed Advanced Security database option.

Given the sensitivity of the data stored in OHI, implementing TDE for OHI Back Office and OHI Data Marts should be considered as non optional. Like implementing TDE is almost a standard action for Oracle Database customers that implement a secure database environment.

This 'hardening' of the database by implementing encryption still allows all authorized applications, whether it is the OHI Back Offce Forms or JET user interface, web service calls or an analytical application, to access and, when applicable, manipulate OHI Back Office and OHI Data Marts data safely and consistently.

1.12.3 Further database security considerations

For OHI Back Office a 'database centric approach' is used. This in order to limit resource usage, offer better scalability and eased impact analysis. Besides these strong advantages the main reason however is the strongly increased security and robustness that can be offered by this approach as there is no way to bypass the business rules logic that is implemented directly on top of the data in the database. This is a clear security advantage in comparison with a multi-tier architecture.

As a result extensive database functionality is used of which one should be aware. This to emphasize that database security is one of the most important security areas for your organisation.

Note:

- The UTL_FILE and DBMS_LOB packages are used to read and write OS files from and to database directories.
- The Oracle JVM in the database is used to implement retrieving directory contents for database directories.
- Oracle Advanced Queuing (AQ) queues are used as a technical transport mechanism to implement calls from within the database to web services in the outside world. A dedicated database external process is expected to use these queues to instantaneously implement the actual call to the web service to be called. For this OHI offers a standard Oracle Service Bus (OSB) project, which use is optional.
- It is recommended to use another Service Bus project or other middleware to act as a go-between between the OHI Back Office web service calls and the actual to be called remote service.
- Another set of AQ queues is used to publish functional messages meant to consume by other applications than OHI.
- The OHI Data Marts database uses a database link to read data from the OHI Back Office database. Only the relevant tables can be read using this database link.
- The OHI Back Office and OHI Data Marts databases contain PI and PHI data which may and must not be accessed by unauthorized users.

See for further information regarding database security the 'Oracle Database Security Guide'.

1.13 Oracle Fusion Middleware Security

The following components of Oracle Fusion Middleware (FMW) are used by the OHI Back Office application:

- Oracle HTTP Server serving for static content (HTML files, images, JAR files, online help, etc) and batch overview output and error logs.
- Forms Server listener servlet and Forms Server (see 'Forms Services' later in this chapter).
- WebLogic application server (WLS) for deploying and running web services (HSL services and SVL services).
- Oracle Service Bus (OSB), as optional component, running on WLS.

In most cases the same server is used for FMW and batch processing. It is therefore very likely that PI and/or PHI data will be processed on the application server(s).

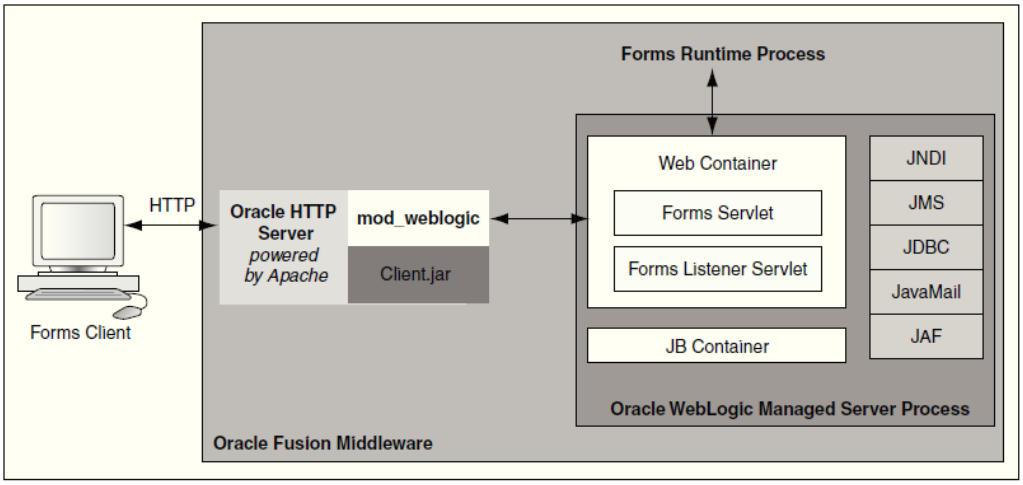
This means that the FMW instance(s) must be well secured.

See 'Oracle Fusion Middleware - Securing a Production Environment for Oracle WebLogic Server'.

1.14 Forms Services Security

Much of the OHI Back Office functionality is managed through an Oracle Forms application (aka 'Forms GUI'). It is accessed by back office users over the intranet.

The Forms GUI runs on Forms Services which is part of FMW.



The Forms server listener creates a Forms runtime process for each new user interface session. The Forms session connects to the OHI Back Office database through Oracle Net Services.

The presentation of the forms session is handled by the Forms Client which runs using a Java Plugin (JPI) in the browser (deprecated) or in a separate Java Web Start process if the JNLP plugin is used (the preferred and advised option). The Forms Client communicates with the forms session through HTTP(S).

FMW most often also includes the WLS 12c domain to which the OHI Back Office web services are deployed.

Recommendations:

- Use HTTPS for the communication between the Forms Client and the forms session. This should help prevent PI/PHI data from being captured through a network sniffer.
- Encrypt the Net Services traffic between the forms session and the OHI Back Office database (see 'Net Services' below).
- Set `FORMS_RESTRICT_ENTER_QUERY=true` to prevent SQL injection within the user interface windows.

See the security-related topics in 'Oracle Fusion Middleware Forms Services Deployment Guide'.

Each functional module, consisting of one or more 'windows' is subject to the functional role-based module authorization as enforced by the OHI Back Office forms user interface.

1.15 OHI JET Security

For OHI Back Office a growing number of Oracle JET based user interface windows is being available. This is a fully browser based user interface.

These OHI JET based windows are using a private set of HSL services to access the OHI Back Office database, named PSL (Private Service Layer for OHI JET).

Authentication for this separate UI is fully based on the database accounts with accompanying OHI officer accounts, as also used for the Forms based user interface for OHI Back Office.

Each OHI JET based window is, like a Forms based window, a so called 'module'. Authorization to these OHI JET windows is also dependent on the functional module authorization as arranged within OHI Back Office.

1.16 Net Services Security

The OHI Back Office and OHI Data Marts databases are accessed through Oracle Net Services (aka SQL*Net).

Recommendations:

- Implement Net Services encryption to help prevent that malicious users may capture sensitive (PI/PHI) data in transit to or from the OHI Back Office and OHI Data Marts database.
- Change listener configuration to restrict listener access to trusted clients and systems.

1.17 Oracle Security Documentation

To implement security, Health Insurance Back Office uses other Oracle products, such as Oracle Database, Oracle WebLogic Server and Oracle Service Bus. See the following documents:

- Oracle Database Security Guide
- Oracle Fusion Middleware Securing a Production Environment for Oracle WebLogic Server
- Oracle Fusion Middleware Forms Services Deployment Guide
- Oracle Service Bus Administration and Secure Oracle Service Bus
- Oracle Linux 7: Security Guide

These documents and ther Oracle documentation is available from the Oracle Help Center at:

http://docs.oracle.com

2 Performing a Secure OHI Back Office Installation

This chapter aims to help you prepare for a secure OHI Back Office installation.

For information about installing OHI Back Office, see **Doc[1]**.

For information about installing OHI Data Marts, see **Doc[7]**.

2.1 Pre-Installation Tasks

The following pre-installation tasks should be performed before OHI Back Office can be installed:

- Install and harden operating system of application server.
- Install and harden operating system of database server.
- Install Oracle database on database server.
- Install Oracle Fusion Middleware on application server.
- Procure SSL certificate for FMW installation
  used by Forms Services, SVL and HSL services.
- Define a back up strategy which does not compromise the security of OHI Back Office production data.
- Implement Transparent Data Encryption (TDE) for the database.
- Install and Configure Oracle Database Vault.

2.2 Installing OHI Back Office Securely

When installing OHI Back Office components:

- Do not install and authorize any components you do not need. For example, only configure and authorize web services that you will use.
- Read the paragraphs in this manual about the security aspects of the component before installing.
- Read the installation manual for the component for additional security aspects.
- Use your common sense to avoid security risks.
- Define and implement a strategy for logging changes to data in the OHI Back Office tables.
- Keep a log of your implementation decisions for discussing with your Data Protection Officer.

2.3 Post Installation Tasks

After OHI Back Office is installed ensure that the following measures have been taken:

- Use secure files for storing weblogic usernames and passwords.

- Turn on encryption for Oracle Net Services.
- Limit access to Oracle Net Services to trusted clients and systems.
- Turn on encryption for Forms Client.
- Design and implement a sufficient secure database password policy ('Access Control and Security Privileges' in **Doc[1]**). Beware of password expiration moments for accounts that are accessed automatically, like for the 'batch' database account or for OPSS accounts used by the Forms FMW infrastructure processes. To enable as well as protect against the side-effects of expiration make use of the password rolling over window functionality offered by database 19.12 (July 2021) and later.
- Ensure that sensitive data is masked or anonymized in non-production data.
- Remove any unneeded database links.
- Enable Database Vault and the OHI specific realm for Database Vault.
- Notify your Data Protection Officer.

2.3.1 Changing Default Passwords

If you have implemented a password policy, you should have got rid of default passwords by now.

Nevertheless it makes sense to review all passwords of installed components so that they cannot be guessed by reading the documentation.

Passwords to look out for:

- OS accounts
- Standard Oracle database accounts
- Weblogic admin account
- OHI Back Office schema accounts, BATCH account, runtime account for SVL and HSL services
- Custom schema accounts

Finally, ensure that passwords for a production system are different from non-production systems.

3 Implementing OHI Back Office Security

This chapter provides an overview of the relevant security aspects of each component of OHI Back Office.

3.1 Implementing SVL Services Security

3.1.1 Server-Side Authentication

The web service requires a database connection to the database provided by a connection pool.

The database connection should use an account specifically created for SVL services which has a minimal set of object privileges. This means only SVL web service related database objects can be accessed by the account.

The database connections are managed by the standard connection pooling mechanism of the WebLogic application server.

This implies:

- The connection details (connect string with username/password) are linked to a logical database source (for example DSohiSVL).
- Username/password details are managed by WebLogic application server and are not known to the calling application.

A properties file is used to configure the data source and OHI BO officer (e.g. 'WS_INTERFACE_USER1') for each web service operation. This properties file is stored at the OS level and is read when the WLS managed server process is started.

Risks / measures:

- Risk: The passwords for the datasources are stored in encrypted form in the Weblogic configuration and are known to WLS administrators.
  Action: ask the functional application administrator to choose and enter the passwords for data sources (see notes about 'Password Aging' in Appendix A).
- Risk: The passwords for the datasources are known to the functional application administrators.
  Action: implement a logon trigger that limits access for SVL specific accounts by allowing only connections from the IP addresses or network subdomain of the application servers. The implementation of a logon trigger is described in doc[1].

3.1.2 Client-Side Authentication

All SVL services are preconfigured to use a default security policy (policy:Wssp1.2-2007-Https-BasicAuth.xml).

This security policy requires:

- Authentication by username/password (user created in WLS security realm).
- SSL encryption
- Inclusion of timestamp data

The purpose of the default security policy is to reduce the risk that members of your organization can invoke SVL web services to retrieve or manipulate OHI Back Office data.

It is strongly recommended that you further strengthen the security measures to reduce the risks of unauthorized use of the SVL services.

With the WLS console you can replace the default policy with a stronger policy.

Common risks / measures when using the default policy:

- Risk: The web service may be called from anywhere in your network.
  Action: use network access control to restrict the servers that access the web service.

- Risk: The username/password to access the web service is known at the client application level.
  Actions:

  - Use password aging to force periodical changing of the passwords.

  - Ensure that unauthorized personnel do not have access to the server running the web service client application.

  - Consider using certificates at server and client level.

Note that the default policy is limited to transport layer security. Consider the implementation of message level security if requests for SVL services are routed through other servers.

3.1.3 Server-Side Authorization

The web service should run under a separate Oracle database account, specifically created for running SVL services.

The procedure to create this account and its privileges is described in **Doc[3]**.

Risks / measures:

- The service layer account may be abused by others to make unauthorized changes to OHI Back Office data.
  Actions: use strong passwords and password aging (with a procedure to periodically change this in time) to reduce the likelihood that unauthorized users can guess the username / password. The only personnel likely to know the username / password of the service layer account will be the WLS or functional application administrators.

3.1.4 Client-Side Authorization

Each SVL web service is configured to use a security policy which requires the calling application to supply a username/password and timestamp data for each call.

The username/password must belong to a valid WLS user account.

Risks / measures:

- Risk: Any valid Weblogic user in the same security realm can call the web service, for example the WLS administrator.
  Action: revise the security configuration of the web service to require a

specific role, which limits the number of WLS users to those linked to this role.

- Risk: When no special actions are taken all deployed web services may be used when these are configured while only a limited set may be needed.
  Action: revise the security configuration of the web service to require a specific role and give this role only access to the services that are required.

3.1.5 Access Control

The WebLogic application server provides the option to make services available to a select group of systems.
Applying access control to the calling servers likewise (which themselves function as a client of the web service) makes the risk of unauthorized use easily manageable.

For more information, please read the instructions about using network connection filters in the WebLogic Server documentation.

3.1.6 Data (transport) Integrity

The default security of SVL web services requires SSL encryption.
If a modern implementation such as TLS 1.2 (used by WebLogic 12c) is used, it is impossible to capture the XML messages with a network sniffing tool.

Risks / measures:

- Risk: SSL (transport layer encryption) is insecure if there are multiple hops between the web service client and the application server.
  Action: choose a more secure policy to implement message level security.

3.1.7 Security Audit And Alarms

The WebLogic server offers extensive functionality for logging and auditing purposes. The standard documentation contains more information.

3.1.8 Java Heap Inspection

Doc[2] describes some ways to add JVM options to the Server Start arguments of the WebLogic Managed Server. For debugging purposes it is sometimes required to add JVM options to be able to attach to the JVM using tools like Java Mission Control and Java Flight Recorder, even from machines other than the Weblogic server.

Setting JVM options to allow processes to attach to a running JVM introduces a security vulnerability for Java Heap Inspection and must always be avoided. Therefore you should enable thse options in production only temporarily, while debugging a specific production issue.

3.2 Implementing HSL and PSL Services Security

This chapter describes security measures for all RESTful services, both the HSL services, the 'OHI' RESTful API, as well as the 'private' RESTful service layer names PSL. The PSL services are only a special dedicated version for private use by the OHI JET application. Where HSL is referenced you may also read PSL.

3.2.1 Server-Side Authentication

The web service requires a database connection to the database provided by a connection pool.

The database connection should use an account specifically created for HSL services which has a minimal set of object privileges. This means only HSL web service related database objects can be accessed by the account.

The database connections are managed by the standard connection pooling mechanism of the WebLogic application server.

This implies:

- The connection details (connect string with username/password) are linked to a logical database source (for example DSohiHLS).
- Username/password details are managed by WebLogic application server and not known to the calling application.

A properties file is used to configure the data source and OHI BO officer (eg. 'WS_INTERFACE_USER1') for each web service operation. This properties file is stored at the OS level and is read when the WLS managed server process is started.

Risks / measures:

- Risk: The passwords for the datasources are stored in encrypted form in the Weblogic configuration and are known to WLS administrators.
  Action: ask the functional application administrator to choose and enter the passwords for data sources (see notes about 'Password Aging' in Appendix A).
- Risk: The passwords for the datasources are known to the functional application administrators.
  Action: implement a logon trigger that limits access for SVL specific accounts by allowing only connections from the IP addresses or network subdomain of the application servers. The implementation of a logon trigger is described in doc[1].

3.2.2 Client-Side Authentication

Starting with OHI BO release 10.18.1.0.0, all HSL operations support Basic Authentication and OAuth2.0. Since release 10.24.8.0.0, OAuth2.0 is the default.

The properties file hsl.properties controls several security settings. It is processed when the HSL service is started.

Important:

- '`hsl.app.authorization`' defines which authentication methods are allowed for a HSL application: 'BASIC' or 'TOKEN'.
  - If TOKEN is used then the service must be deployed to WLS with 'Custom Roles and Policies'.
- HTTP OPTIONS (pre-flight check) may be called with HTTP and does not need to be authenticated.
- Use `hsl.`*`app`*`.allowedorigins=`*`serverlist`* to indicate 'trusted' servers for which CORS (Cross-origin resource sharing) response headers may be issued.
  If this property is not set, the calling application is supposed to be hosted on the same server as the WebLogic Application Server.

- Regardless of the setting of hsl.app.allowedorigins it is recommended to use WLS network access control to limit the number of servers which may act as HSL service clients.
  This effectively means that HSL services may never be called directly by end users!

- OHI provides so called 'helper' or 'check' webservice operations to obtain additional data elements. These can be used when a calling environment only has a single guessable (and as such 'insecure') identifier to start a webservice conversation. The additional data element can typically be used in combination with the original identifier to call a webservice operation that returns a non-guessable UUID for an object which can be used during the rest of the webservice conversation. For more information please consult the HSL user manual **Doc[9].**
  It is very important that these check operations are authenticated separately from the regular webservice operations, to implement an additional line of defense. Each individual check service should have an individual authentication, so a breach of authentication for one such check service does not compromise the others.

- ***Basic Authentication***

  This is the deprecated security mode for HSL services.

  It implies:

  - Authentication by username/password of an account in the WLS security realm.

  - SSL encryption

  The purpose of using Basic Authentication is to reduce the risk that members of your organization can invoke HSL services to retrieve or manipulate OHI Back Office data.

  Specific risks / measures:

  - Risk: The Weblogic username/password to the HSL service is known at the client application level.
    Actions:
    1) force periodical changing of the passwords (but beware to change them in time, before expiring)
    2) ensure that unauthorized personnel do not have access to the server running the web service client application.
    3) Consider using certificates at server and client level.

- ***Token Validation For OAuth2.0***

  Starting with OHI release 10.18.1.0.0 it is possible to use OAauth2.0 token validation. It has become the default and preferred way of authorization and authentication in combination with OpenID Connect (OIDC) since release 10.24.8.0.0.

  If token validation is configured, the HSL application calls the remote Authorization Service to validate the token, before the operation itself is attempted.

  ☞ For the OHI JET based user interface, token validation for OAauth2.0 is the standard method used for authentication. The authentication itself is done by the database as only valid OHI officer accounts with an associated database account are allowed to use OHI JET functionality, and only when authorized.

  Note:

- The HSL application must be deployed with Custom Roles and Policies to use token validation.

- The access token must be a JSON Web Token (JWT).

3.2.3 Server-Side Authorization

The web services should run with a separate Oracle database account, specifically created for running HSL services.

The procedure to create this account and its privileges is described in **Doc[2]**.

Risks / measures:

- Risk: The service layer account may be abused by others to make unauthorized changes to OHI Back Office data.
  Actions: ensure that the service password is strong, changed frequently and only known to a very small number of people within your organization. See 'Password Aging' in Appendix A.

3.2.4 Client-Side Authorization

As of OHI Back Office 10.18.1.0.0, OAuth2.0 can be used instead of Basic Authentication.

- ***Basic Authentication***

  If Basic Authentication is used the HSL application requires the calling application to supply a username/password for a valid WLS user account.

  Risks / measures:

  - Any valid Weblogic user in the same security realm can call the web service, for example the WLS administrator.
    Action: revise the security configuration of the web service to require a specific role, which limits the number of WLS users to those linked to this role.

  - A Weblogic user with access to a HSL application can use all operations.
    If fine-grained authorization is needed, consider using authorization using OAuth2.0.

- ***Authorization through OAuth2.0***

  When using OAuth2.0 authorization, an Authorization Service is used to verify that the caller is a legitimate user of the system and that the caller is authorized for the requested HSL service operation (expressed through path+method).

  Using OAuth2.0 offers the following advantages over basic authentication:

  - No need to pass WLS credentials when calling the HSL operation.

  - Fine-grained authorization

  Risks / measures:

  - Risk: The Authorization Service is a potential point of failure.
    Action: Verify that access token verification is working correctly. Both authentication and authorization should be rigorously tested.

See **Doc[2]** for a complete description of setting up HSL for OAuth2.0.

3.2.5 Access Control

It is strongly recommended to:

- Never allow end-users to invoke the HSL services directly.
  Always use an application or at least a middleware component such as Oracle Service Bus, which implements authorization in some way, as a go-between the HSL service and its client applications.

- Restrict access to the HSL services to a limited number of 'trusted' servers.
  WebLogic Application Server allows you to configure a limited access list. As an alternative, use network configuration and firewalls.

For more information, please read the instructions about using network connection filters in the WebLogic Server documentation.

3.2.6 Data (Transport) Integrity

HSL web services require SSL encryption for all calls with the exception of HTTP OPTIONS.

Risks / measures:

- Risk: SSL (transport layer encryption) is insecure if there are multiple hops between the web service client and the application server.
  Action: choose a more secure policy to implement message level security.

HSL web services use UUID's (Universally Unique Identifiers) to identify OHI objects. These UUID's are not guessable as they consist of a long non-guessable hexadecimal string, in order to protect against the Insecure Direct Object Reference (IDOR) risk.

For compatibility reasons two OHI Back Office parameters are present to keep on using the deprecated guessable numeric object identifiers, such as policy numbers relation numbers, etc. during a transition period in which existing usage needs to be adapted.

These parameters apply only to the HSL webservices, the PSL webservices always use UUID's as object identifiers. These parameters are intended to use during a transition period in which the calling applications are adapted for supporting UUID's.

As long as these compatibility settings are used a warning is issued during the object check as part of an OHI release installation.

It is strongly advised to plan for a limited time to use these compatibility parameters as this disables an extra line of defense against potential misuse of the HSL webservices.

Risks / measures:

- Risk: A prolonged use of the UUID compatibility settings for the HSL webservices increases the period during which a bad actor, who has breached one or more security layers, may use easy guessable (mostly more or less successive) values for identifiers.

3.2.7 Security Audit and Alarms

The WebLogic server offers extensive functionality for logging and auditing purposes. The standard documentation contains more information.

3.2.8 Java Heap Inspection

Doc[2] describes some ways to add JVM options to the Server Start arguments of the WebLogic Managed Server. For debugging purposes it is sometimes required to add JVM options to be able to attach to the JVM using tools like Java Mission Control and Java Flight Recorder, even from machines other than the Weblogic server.

Setting JVM options to allow processes to attach to a running JVM introduces a security vulnerability for Java Heap Inspection and must always be avoided. Therefore you should use these options only in production while debugging a specific production issue.

3.3 Implementing Service Consumer security

Callouts from PL/SQL within the database to external web services are implemented through passing a dedicated combination of a correlated request and response AQ queue and a message handler application on an application server to implement the actual web service call.

This message handling application can be an OHI provided Oracle Service Bus (OSB) component or a custom solution.

3.3.1 Identification / Authentication

Three levels of authentication are relevant.

- ***Authentication at database level***

  The database AQ queues need to be accessed through a connection pool that uses a data source to connect to a database account that has privileges to access the queues.

  Risks / measures:

  - Make sure the password needed for authentication to this account is known on a need to know basis.
  - Prevent automatic expiration to avoid service interruptions, but for security reasons do implement a regular interval to reset the password and reset it when there is a change in the team of people that knows the password.

- ***Authentication at WebLogic Server/OSB level***

  The JMS queues that are created in WebLogic to expose the AQ database queues are accessible through WebLogic. Therefore, the authentication to the WebLogic environment where the JMS queue definitions reside should get attention.

  Risks / measures:

  - Make sure credentials for getting access to WebLogic are only known on a need to know basis and manage these well.

- ***Authentication for accessing the external web services***

  The "message handler application" on the application server needs to authenticate when calling the (proxy for) the external web services.

Risks / measures:

- Make sure the authentication details for the web services are only known on a need to know basis.
- Implement a secure store for the way the credentials are stored.

3.3.2 Authorization

The three levels of authentication mean also three levels of authorization apply.

- ***Authorization at database level***

  The database user that is used for accessing the database AQ queues has authorization to enqueue and dequeue multiple OHI messaging queues.

  Risks / measures:

  - As more queue authorizations are present than strictly necessary, misuse of the account, meaning not allowed enqueue or dequeue actions that disturb the correct functional working, should be prevented. A trigger could be implemented to check only the allowed application servers can connect to this database account.

- ***Authorization at WebLogic/OSB level***

  Access to the WebLogic environment provides access to the connection pool, JMS queue definitions and the OSB project definition as well as all the other WebLogic reources. There is standard no additional authorization implemented.

  Risks / measures:

  - As usually no addtional authorization is arranged for the WebLogic environment, privileges can be widely misused to execute actions that can harm the OHI process handling. For this reason bring down the Admin Server during runtime use to prevent any unwanted actions or changes.

- ***Authorization at the external web service level***

  The external web services must require authentication. It is up to the called web service environment whether any additional authorization is setup. Usually a more generic account will be used providing access to the different external web services.

  Risks / measures:

  - The generic account might provide authorization for other unwanted resources. Make sure the (proxy) service environment does only provide authorization for the necessary external web services.
  - When possible check at the called side that only calls are being made from authorized application servers preventing potential misuse for the web services from other sources that have network access to the server hosting the (proxy for) the external services.

3.3.3 Access Control

As the external services are typically application integration services no specific access control is implemented.

Risks / measures:

- Implement the relevant authentication and authorization measures to prevent the external services are accessible from other sources than the intended application interface functionality.

3.3.4 Data (Transport) Integrity

The exchanged web services may contain confidential information.

Risks / measures:

- The calls to the external web services require appropriate encryption, a sufficient modern implementation of SSL should be applied.
- As the message queues contain a temporarily copy of the message a process that can access the message queues has access to the confidential data. This means no other access should be given, outside the regular privileges, for using or querying other OHI data. Also caution is needed for access given for investigating potential interface problems, make sure authorized personnel will obey appropriate confidentiality policies.

3.3.5 Security audit and alarms

As several components are involved auditing and alarms can be applied for these different components.

- ***Database level***

Risks / measures:

- Consider auditing any logon attempt to the database account that provides access to the relevant queues. This may of course be part of the more generic measures taken at the database level.
- Do list regularly which accounts have dequeue/enqueue privileges for the queues to prevent unwanted privileges may exist for a longer period of time.
- Prevent retention times for the relevant queue tables are longer than necessary for monitoring and supporting the correct working.

- ***WebLogic/OSB level***

The WebLogic server offers extensive functionality for logging and auditing purposes. The standard documentation contains more information.

Also for the Oracle Service Bus instructions for monitoring and securing access are provided in the Administration Guide.

Additional security measures for the OSB are described in the manual for Securing your OSB environment.

See the documentation references in the first chapter for the relevant documents.

3.4 Implementing Batch Processing Security

Background processes are started and monitored by the OHI Back Office Batch Scheduler. This Pro*C program runs on the application server using a customer configurable 'batch' database account.

In most cases, the batch scheduler runs under the 'batch' OS account although this is not a requirement.

The OHI Back Office batch scheduler retrieves the password for the database batch account from a secure external password store (SEPS).

3.4.1 Identification / Authentication

- *Authentication at OS level*

  The OS user logs in as linux user, typically 'batch', using a sufficient complex password to be set by the system administrator.

- *Identification / authentication in the Oracle database*

  The batch scheduler presumes the configuration of a Secure External Password Store (SEPS). This means that a SQL connection can be created using an account alias, identifying a set of credentials in the SEPS, instead of passing the account credentials on the command line.

  Doc[1] describes how to set up SEPS for use with the batch scheduler.

  Example: assume a SEPS with account alias 'batch_prod' configured to connect to the production database as database user BATCH. If the batch scheduler is started with the account alias 'batch_prod' the SEPS is read and a connection is made to the database account BATCH, using the password that was hidden in the SEPS.

  Risks / measures:

  - OS accounts/individuals with access to (a copy of) the secure password store can impersonate the BATCH database account to logon to the OHI Back Office database. The risk is limited as the account has limited privileges but may be misused to interfere and potentially disrupt batch processing.
    Actions:
    - Regularly change the password for the BATCH account (this requires an update to the SEPS file as well).
    - Ensure that the SEPS file can only be read by 'batch' and the owner of the OracleNet listener. Ensure that no one but the owner of the SEPS file has write-permissions.
    - Implement a logon trigger to reject sessions from unauthorized systems such as desktop computers.
      The implementation of a logon trigger is described in **doc[1]**.

3.4.2 Authorization

- *Authorization at OS level*

  The OS user 'batch' is only used to start the OHI Batch Scheduler and does not own OHI Back Office application files.

The batch scheduler and its associated batch scripts need to create the following files:

- A batch scheduler log with process information in the $OZG_LOG directory.
- A progress output file with progress data in $OZG_OUT/<configurable folder path> for each request.
  Viewable by the requestor using the OHI Back Office Forms GUI.
- A log file with process data in $OZG_LOG/<configurable folder path> for each request.
  Viewable by the requestor using the OHI Back Office Forms GUI.
- Other output files (CSV, XML, HTML, etc) written to database directories or application server folders. Which folders these are is determined during configuration of OHI Back Office.
  These folders are and should standard not be viewable by back office users through the HTTP server, access to these folders should be authorized separately.

The directories $OZG_OUT, $OZG_LOG and their subdirectories must be writable by the 'batch' user and readable (!) by the OS user running Fusion Middleware / WLS / OHS to show created output.

The database directories likely to be written by batch scripts must be writable by the OS account that runs the database processes, typically 'oracle'. In order to allow the back office users to view these output of batches they started, these database directories must be synchronized or shared with the application server or a different shared storage solution. As these files may contain sensitive data some way of secure authorized file access needs to be implemented.

- ***Authorization at database level***

  The Oracle database batch account has database role OHI_ROLE_BATCH to access a very limited set of database objects to be able to start the OHI Back Office batch process functionality. These processes can only be started through an interface package, available to the batch account, when an actual request for such a process has been entered and reached the status 'starting', which status can only be set by the internal logic of the batch scheduler, so it is not possible to run the logic of such a process when using this batch account to connect to the database. This in order to further prevent any misuse of having access to this batch account.

  This is implemented as additional security measure since the database batch account is authenticated through SEPS at the OS level by anyone (any OS user on the application server) with file access to the SEPS file. A typical example is a system administrator and in this way persons with having this, already very privileged, access are prevented from having access to or impact on the OHI Back Office data.

- ***Authorization at application level***

  The actual authorization to request the execution of an OHI batch processing script is based on the module authorization as provided by the OHI Back Office application.

3.4.3 Access Control

You can only logon as a linux user 'batch' if you have the correct password or if you have super user (root) rights.

Risks / measures:

- Administrators and 'root' can log in as user 'batch'.
  Actions:
  - Limit the users who can login as 'batch' to application administrators.
  - Limit the users who can login as 'root' to 1-2 people.
  - Change the passwords for 'root' and 'batch' regularly.
  - Remove .rhosts, .host.equiv etc. in the home directory of 'batch' which could be used to log on from other systems.

3.4.4 Data (transport) integrity

The 'data' produced by the batch scheduler and its associated scripts that can be viewed from the OHI Back Office Forms GUI is mostly progress and process related.

The actual output (XML, CSV, HTML etc) cannot be viewed by default by the requestor but may be accessed by other processes.

Risks / measures:

- Users can use the script requests to view output of any confidential process (namely, via the forms built-in *show_document* or directly via URLs).
  Actions:
  Set up authorizations on batch output (described in **doc[4]**) and purge output directories regularly (for example by a daily cron job that deletes all output files older than 2 days).

3.4.5 Security audit and alarms

The UNIX 'last batch' command can be used to find out who has been logged-in as 'batch' in the last x days. An example of the output is shown below:

```
batch    pts/1        vlan441dhcp473.n Tue Sep  2 07:24 - 07:31  (00:06)
batch    pts/16                        Mon Sep  1 16:47   still logged in
batch    pts/18                        Mon Sep  1 12:08 - 18:23  (06:15)
batch    pts/16                        Mon Sep  1 10:01 - 16:47  (06:45)
batch    pts/1        vlan441dhcp484.n Fri Aug 29 15:18 - 16:00  (00:42)
batch    pts/14                        Fri Aug 29 12:03   still logged in
batch    pts/8                         Fri Aug 29 11:59   still logged in
batch    pts/1        vlan441dhcp223.n Thu Aug 28 18:46 - 18:52  (00:05)
batch    pts/8                         Thu Aug 28 16:51 - 17:14  (00:23)
batch    pts/18                        Thu Aug 28 15:47 - down   (04:27)
```

Please note that most UNIX systems can be configured such that the log involved can be purged periodically.

In addition, database auditing can, if required, be used to monitor the behaviour of the 'batch' database user.

3.5 Implementing Operating System Security

See the following documents:

- Guide to the Secure Configuration of Red Hat Enterprise Linux 7
- Hardening Tips for the Red Hat Enterprise Linux 7
- Oracle Linux 7: Security Guide

3.6 Implementing Oracle Database Security

All OHI Back Office data is stored in a single schema in an Oracle 19c pluggable database. Thus, optimized functionality is implemented to guard consistency of the data. On-going developments in the field of authorization and integrity monitoring contribute to the consistency and security of the OHI Back Office database.

Quite some years ago, a very strategic decision was taken to protect the consistency and integrity of OHI Back Office data with a robust business rule implementation layer in PL/SQL. This approach is now also known as the 'smart database paradigm'.

The OHI Back Office application contains PL/SQL packages for creating and maintaining standard security roles for different profiles which can be assigned to custom schema's and the BATCH account.

At the same time, an authorization mechanism was developed to grant users of the OHI Back Office Forms GUI the required set of database privileges only for the duration of the GUI session.

This 'hardening' of the database means that OHI Back Office data can be accessed and updated safely and consistently with any tool that can connect to the database without compromising the business rules that guard the data.

Additionally to this an implementation of Oracle Database Vault has been developed that resulted in an OHI specific realm definition. When this, still optional but highly advised realm, by means of using Database Vault, is implemented, the privileged DBA accounts can no longer access the OHI database objects and the user management is transferred to other privileged accounts introduced solely for this purpose.

3.6.1 Identification / Authentication

To access OHI Back Office data, a database connection is required. To get a connection, the caller must login with a valid username and password combination (stored in the database).

The OHI Back Office data can be accessed by:

- OHI object owner accounts ('table owner', 'view owner', 'dynamic pl/sql user'), only possible when the OHI specific Database Vault realm is not implemented.
- Personal accounts, standard with minimal create session privileges, for individual OHI Back Office Forms and OHI JET GUI users. These accounts are dynamically given additional privileges to OHI Back Office objects for the duration of a session by the Forms GUI. As the OHI JET GUI uses web services to access the data the personal account is used only for authentication and authorization.
- A 'batch' account for batch processing.
- DBA and other privileged accounts (like database accounts with special OS privileges like SYSDBA or accounts with SELECT ANY TABLE privilege, etc.), only in the situation that the Database Vault option and the OHI specific realm are not implemented.
- Custom schemas used for custom development.

- A JMS queueing account, for access to OHI provided AQ queues with a JMS payload.

- Web service servicing connection pool accounts used by HSL, PSL and SVL services (these accounts have only access to web service supporting code objects).

- ***OHI object owner accounts***

  The table owner account is a very privileged account which contains all data objects and many code objects. Through this account data and code can freely be manipulated, meaning it is a very high risk account and should not be accessible during regular use of the application.

  The view owner and the dynamic pl/sql owner account can also be used to freely access almost all data.

  Risks / measures

  - Risk: password is known to too many persons and may fall in hands of someone with less appropriate intentions.
    Action:
    a) Make sure the password is sufficient complex so it cannot easily be remembered; and make sure it is changed regularly.
    b) Implement the OHI specific realm i.c.w. Oracle Database Vault which prevents logging on as this requires additional authorization for logging on with these accounts.

  - Risk: account is used for other purposes than OHI release maintenance which may lead to accidental or intentional disruptive actions.
    Action:
    a) Determine what usage is needed and create a separate account (or accounts) with less disruptive goal oriented privileges for executing the 'other purposes'.
    b) By default lock the standard 3 schema accounts outside maintenance periods as they should only be needed when OHI patch/release installations are executed.
    c) Implement the OHI specific realm i.c.w. Oracle Database Vault which prevents logging on as this requires additional authorization for logging on with these accounts.

- ***Personal and other database accounts***

  This includes users of the OHI Back Office Forms and OHI JET GUI, custom development schemas and individual users accessing the OHI Back Office or OHI Data Marts database with a client tool (for example SQL*Plus).

  The personal accounts for these users will be created by an account with account management privileges. When Database Vault is enabled a separate account manager role will be present which can be assigned to one or more accounts with this privilege, which enables these accounts to create the personal accounts and/or reset passwords for users who require this. As long as Database Vault is not enabled the database administrator will also have the privileges for creating and managing these accounts, which is a combination of responsibilities that should be avoided as this can be misused, especially when such an administrative account is used by a group of administrators.

  Risks / measures

- Risk: initially assigned/chosen passwords are not changed.
  Action: expire passwords when created, being a component of generic password expiration (see later) and lock these accounts when they are not being used within a few days after the initial password assignment.

- Risk: accounts created by the system (DBSNMP, OUTLN etc.) can be accessed.
  Action: lock these accounts unless they are actually used.

- Risk: accounts for employees who have left remain extant.
  Action: implement a 'leaver' process that locks/ends immediately and/or delete these accounts.

- Risk: the application deployment in the the OHI Back Office database comes with a default 'manager' account and role 'MANAGER_ROL' for 'bootstrapping' the authorization structure. This account is not tied to a person and should only be used to setup a role based application authorization structure.
  Action: As soon as the authorization structure is done the database account should be locked and only be unlocked in case of 'emergency' when the authorization structure that has been setup is by accident insufficient to grant authorizations.

- Risk: non personal OHI 'group accounts' are created which are being used by multiple persons, resulting in not being able to track down certain actions by such an account to a single person.
  Action: Never use any kind of group accounts, always use personal accounts. Make sure that when a set of privileges is needed for a group to assign these to a role and grant the role to the personal accounts. Each action in the application should be executed in such a way that a single person can be held accountable for. This means that also batch and background processes that are planned directly or indirectly should be administered through accounts that identify a single person. Although this might add complexity for managing the privileges it is important to implement this.

- Risk: passwords for OHI Back Office users remain unchanged, facilitating misuse by third parties.
  Action: define and implement a password policy. This includes expiring passwords after a period of time and implementing complexity criteria (length, different to previous passwords, use of a mix of numbers/characters, etc.) that must be met by new passwords to reduce the risk of guessing each other's passwords.

- Risk: accounts are used to connect to the database using a different environment than the standard user interface in an attempt to try to exploit the additional privileges that are assigned to the OZG_ROL role.
  Action: use the Back Office Parameter that identifies the IP address(es) of the application server(s), which may enable the secure application role. This prevents users in trying to misuse their account by enabling the secure application role while not using the user interface.

- Risk: custom code schema accounts which are also registered as an OHI officer enable the misuse of the secure application role.
  Actions:

    - Ensure that custom schema accounts are not known as an OHI officer so they cannot be used to manipulate OHI data.

    - Limit the custom code account to OZG_ROL_DIRECT or similar privileges directly assigned to the custom code account.

    - Lock custom code accounts when not needed; only unlock and use account in production during deployment of custom code changes.

- Risk: custom code and other interface accounts, that are configured once in an interface definition and normally will have no expiration policy, because this can cause interface disruption. Typically passwords of these accounts are in some configuration file. Examples: special interface scripts, connection pools.
  Action: administer these accounts on a special list and change their password on a regular basis to prevent misuse; use password rolling (available from database 19.12) to ease these changes; also make sure the configuration files are stored with minimum access rights and use encryption in tooling whenever possible.

- ***BATCH processing account***

  Used exclusively by the OHI Back Office batch scheduler and its batch processes. The 'batch' database account should be used exclusively by the 'batch' OS user and no OHI officer account should be related with it.

  NOTE: This database account also exists in the OHI Data Marts database although there is no batch scheduler active in this environment. The BATCH account is granted the role OBD_ETL_ROLE which has been granted execute privilege on the necessary database packages to perform an ETL run (Extract-Transform-Load) to load data from OHI Back Office to OHI Data Marts.

  For risks and actions, see 'Implementing Batch Processing Security' above.

- ***DBA and other privileged accounts***

  SYSOPER and SYSDBA privileges for OS users, typically used by the 'Oracle' operating system user(s) on the database server for DBA activities, do impose a risk as these give access to a powerful DBA account like SYS. See the Oracle Database Security Guide for setting up your Oracle software environment on the server. Apply security measures for limiting access as much as possible, without compromising the management facilities.

  Risks / measures:

  - Risk: the operating system password for the user 'oracle' remains unchanged and use by third parties cannot be excluded.
    Action: set password expiry and complexity enforcement at operating system level.

  - Risk: misuse by an administrator who abuses the access rights by viewing or changing data because of the extreme powerful privileges assigned to these accounts.
    Action:
    a) implement audit logging on the OS and the database level, consider to use the option to redirect logs to a separate environment and combine these measures with a strong separation of duties. This will make sure illegal misuse of privileges will be logged while that logging cannot be influenced without noticing this.
    b) implement Database Vault and the OHI specific realms for OHI Back Office and OHI Data Marts so database administrative accounts have no privileges for accessing the OHI application objects and also cannot grant

them selves the privileges to do so as such privilege management is assigned to separate account manager accounts.

3.6.2 Authorization in the OHI Back Office database

New database accounts meant for OHI use may only logon to the database (the CREATE SESSION system privilege is required in order to do this).
Privileges to select or manipulate tables or to execute database procedures and packages must be explicitly assigned to the account and normally are not present.

In the case of the personal accounts for Forms GUI users this is done dynamically upon logon to the Forms application:

- The application administrator is responsible for creating an application account (OHI officer account) with the same name as the database account.
- If the user logs in with a personal account without a matching OHI officer account, access to the application is denied.
- If the user logs in with a personal account and a matching time valid OHI officer account, the Forms application will grant privileges for the duration of the session.

For all other situations, database roles or direct granted object privileges are used to grant privileges for the type of account.

In special cases, like for custom code, specialized users can still receive SELECT privileges (OZG_ROL_SELECT role) or the capability to execute DML operations on the 'rule protected' tables using the OZG_ROL_DIRECT role where the integrity is fully safeguarded in the database.

Note that the role OZG_ROL may never be granted directly to any account in the database!

- ***Personal accounts for users of the OHI Back Office Forms GUI***

  The identification/authentication of OHI Back Office users has been implemented as a two-step procedure:

  - All OHI Back Office users are created as Oracle database users with only CREATE SESSION privilege so that they can merely login to the Forms GUI.
  - In order to use the application, the logged-in user needs many more privileges. If the application administrator has created a matching time valid 'OHI user' (OHI officer) for the database account, the secure database role OZG_ROL is dynamically granted for the duration of the Forms session. The database role OZG_ROL has additional privileges to support specific application logic, including privileges for running the Forms based windows. Therefore OZG_ROL should never be granted to a database user.

- ***Personal accounts for users of OHI Data Marts***

  It is recommended to create a separate personal account for each user of Oracle Health Insurance Data Marts. The identification/authentication of OHI Data Marts users has been implemented as a two-step procedure:

  - All OHI Data Marts users are created as Oracle database users with only CREATE SESSION privilege so that they can merely login to the database using SQL*Plus or another enduser query tool.

- In order to access the fact and dimension tables of the data warehouse the logged-in user needs more privileges: The database role OBD_ROL_SELECT has select privileges on all relevant DWH tables and views. For administrators an additional role has been created: OBD_ROL_ADMIN. This role has select privileges on all technical tables and views. If a user is granted both roles he can access all tables/views of the OBD_OWN schema.

- ***Implement Oracle Database Vault for OHI***

  As mentioned DBA accounts and other privileged accounts have very powerful system privileges. 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 is likely that security policies and legal obligations cannot be fulfilled when such administrative accounts can fully access this sensitive data.

  For these reasons 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.

  - The passwords of the technical schema accounts should only be known to the technical administrators. These schema accounts are blocked for use until a functional administrator actively enables a session that permits access, typically done when technical application maintenance is needed. In this nor the technical nor the functional administrators can get exclusive access to these technical schema accounts, only when the cooperate these accounts can be used to access the OHI application objects.

  For more information please see **Doc[1].**

3.6.3 Data (Transport) Integrity

The database files can only be accessed by the 'root' and 'oracle' operating system accounts. Administration of these accounts and access to the database files is the administration organization's responsibility.
Obviously, only 'administrators' should have access to these accounts.

Risks / measures:

- Risk: When access to (a copy of) the database files is acquired in some way these files can be used to dump database blocks in order to interpret stored data. Or the files may be used to create a new database with known accounts in which SQL can be used to simply access the data.
  Action: Implement Tranparent Data Encryption (TDE) at tablespace level, a feature of the Advanced Security database option, to protect against this risk.

3.6.4 Security Audit / Alarms

The Oracle database has many features for configuring database auditing which is explained in the 'Oracle Database Concepts' manual in the first section regarding Security. The 'Monitoring' paragraph describes database auditing possibilities.

The OHI Back Office application provides a change-logging mechanism for the 'business rule enabled' tables.

- By default the user and time of creation and last change are stored within each functional record, including what values have been inserted, deleted or changed.

- Detailed logging can be activated on a per-table basis to trace individual changes chronologically ordered by transaction number.

This OHI logging provides a basis for further additional monitoring and raising security alarms.

Next to this functionality is present to comply with the GDPR to track which user did view data related to a specific person in the database. When this tracking is enabled it is logged which user did query data of which 'relation' through which 'window', where that window determines which data elements might have been viewed.

3.7 Implementing Oracle HTTP server Security

The Oracle HTTP server, in short OHS, is based on the Apache HTTP server. For OHI Back Office it is used for directing the Forms UI generated traffic to WebLogic Forms servers or to retrieve OHI related files, like online help, output and log files or OHI release documentation.

3.7.1 Identification / Authentication

There are a number of ways to configure the Oracle HTTP Server to perform identification / authentication. These are not used currently.

3.7.2 Authorization

The Oracle HTTP Server can be extensively configured to grant access to resources.

Examples here are:

- The option to display directory listings.

- The option to view files in a directory.

- The option to call modules (e.g. PERL, CGI, etc.).

Doc[1] describes how the HTTP server should be configured for use with OHI Back Office and how to set up the authorization structure to restrict access.

3.7.3 Access Control

The Oracle HTTP Server can be configured to grant access to specifically named servers / IP addresses.

By default no restrictions are applied to access the Oracle HTTP Server. It is worth considering restrictions in larger organizations.

3.7.4 Data (Transport) Integrity

The integrity and confidentiality of data can be improved by using encryption. To do this requests are routed through the SSL port. The drawback is the increased server load as encryption is paired with a great deal of processing work.
This mechanism is not used by default. Please note that implementing SSL does not entail changes to the OHI Back Office programs.

3.7.5 Security Audit and Alarms

The Oracle HTTP Server has extensive and easily configurable features for logging administration activity and HTTP requests from users.

3.7.6 End-user identification / authentication

No user authentication is used for OHS.

3.7.7 Client-side identification / authentication

Not used.

3.7.8 Server-side identification / authentication

Not used.

3.7.9 Authorization

Not used.

3.7.10 Access Control

Not used.

3.7.11 Data (transport) Integrity

SSL should be used to encrypt the data between the client and OHS.

3.8 Forms Services Security

3.8.1 Identification / authentication

The user needs to authenticate and in this way provide his/her identifcation as soon as the forms runtime session is started. The opening screen (with the menu) is only displayed after this identification has been executed successfully.

3.8.2 Authorization

Authorization is arranged as follows:

- The user identifies itself using a username/password combination.

- After logging on the user is assigned a database role dynamically.
- After logging on the application menu is created dynamically for the user based on the assigned application roles.

In specific cases the amount of information that is shown to the user is determined dynamically, when the information is displayed, as data may be filtered based on additional authorization within OHI.

3.8.3 Access Control

To prevent SQL injection, which can be used by highly skilled users to get access to unauthorized data, it is advised to turn `FORMS_RESTRICT_ENTER_QUERY` on:

```
FORMS_RESTRICT_ENTER_QUERY=TRUE
```

For more information regarding the environment settings file please look for the 'envFile' string in the 'OHI Installation, Configuration and DBA manual'.

3.8.4 Data (transport) Integrity

Sending a HTTP(S) request to the Oracle HTTP Server starts a session.

By default the message encryption is not activated, but Oracle has added a proprietary encryption scheme that can be enabled by setting environment variable `FORMS_MESSAGE_ENCRYPTION` to 'true'. However, this is not as strong as the SSL standard.

When running in HTTPS mode the traffic between the Forms client and the Forms Server process is encrypted with industry standard SSL encryption. The database login information from the client to the application server is encrypted when running in HTTPS mode.

The setup of HTTPS is described in the Oracle® Fusion Middleware Administering Oracle Fusion Middleware 12c (12.2.1.4) manual. Chapter 6 contains the relevant paragraphs for enabling inbound SSL for Oracle HTTP Server, inbound SSL for Oracle WebLogic Server and SSL between Oracle HTTP Server and Oracle WebLogic Server. In paragraph 6.4 the configuration of SSL for the Web Tier is discussed while in paragraph 6.5 the configuration of SSL for the Middle Tier is dicussed.

Note: As using HTTPS may cause a heavier CPU load, it is recommended to run the webserver Tier on separate encryption-hardware.

3.8.5 Security Audit and Alarms

By default the forms server listener logs all requests for new sessions in a directory like
$MW_BASE/user_projects/domains/frs_d1/servers/WLS_FORMS/logs/access.log.

Here you can view the moments in time a forms session is started and the IP address used.

The HTTP requests from the browser for starting the forms executable, retrieving the Java classes, images, etc. are logged in a directory like
$ORACLE_INSTANCE/diagnostics/logs/OHS/ohs1/access_log.

3.9 Net Services Security

3.9.1 Identification / Authentication

Not applicable to users.

The listener can be configured in such a way that a password is requested on a call to the UNIX server. This will prevent the Net Services being stopped / started without authorization.

By default only localhost accounts can access the listener for status querying and other actions. Make sure you do not set the LOCAL_OS_AUTHENTICATION_<listenername> to OFF.

3.9.2 Authorization

Not applicable.

3.9.3 Access Control

It is recommended to restrict access to the listener to trusted clients and systems.

A firewall or load balancer can be configured for this.
Otherwise you may want to set `tcp.valid_node_checking` and `tcp.invited_nodes` in the sqlnet.ora file to restrict access to trusted clients and systems.

See the Net Services documentation also for the features that are provided by the Oracle programs in this area.

3.9.4 Data (transport) integrity

The traffic between the client and the TNS listener can be encrypted. The TNS listener configuration has to be changed to achieve this. This is strongly advised if your database server is also directly accessible through SQL enabled tools as data may pass unencrypted over your intranet.

3.9.5 Security Audit and Alarms

There are various options for logging the use of Net Services at operating system level. In doing so, logging can be configured on both the client side and on the server side.

3.10 Application Administration

3.10.1 Identification / Authentication

Make sure that only personal database accounts are created for regular application use.

The owner accounts and other powerful non personal accounts like the MANAGER account are locked and only unlocked when strictly necessary.

3.10.2 Authorization

The functional authorization should be carefully setup and no more authorizations should be granted than needed for the job.

Prevent that authorizations are granted that allow both a change and the approval of a change, these are 'toxic combinations'. For that reason no role should be created that receives all authoriations, despite how efficient this may seem.

3.10.3 Access Control

It is recommended to enable 'change logging' for all data that may be deemed sensitive so it always can be determined who implemented when what change on a sensitive data field.

Of course activating change logging should be implemented with care as for very large tables the amount of logged changes may become far too large to manage and check.

Enabling change logging for premium amounts, bank account numbers, addresses and names of relations might be a good starting point.

By default, change logging for the authorization structure is activated and cannot be deactivated.

3.11 Security for non-production environments

Cloning a production database for testing purposes should only be done when it cannot be avoided. A cloned production database contains both personal and sensitive medical data. Unauthorized access to these data, which might seem to be a lesser risk for a clone, may also lead to a data leak.

3.11.1 Subsetting

A non-production environment based on a subsetted clone of a production database should be treated with the same security measures as the production environment itself.

OHI provides a subset implementation based on the Oracle Enterprise Manager (OEM) plugin (an OEM option) for this purpose, to create a fully functional subset database of your production database. This database will contain all OHI setup data but will only contain a subset of the persons, policies and claims related data in your production environment, for example an approximately 1% or 10% subset of the original data.

This reduces the risk of access to sensitive data for test and development purposes but also reduces elapsed time for regresession testing of certain functional processes, what might result in increased stability of your application as testing runs might be repeated more often and at lower resource costs.

3.11.2 Masking

OHI provides optional functionality to implement a masking strategy on copies/clones of the OHI production database, which in itself is strongly advised for a non production-environment.

In fact, each non production-environment based on a full or subsetted clone of a production database should be masked to hide the personal data.

The process of masking hides the personal data but does not change the related details like policy data and medical treatment data.

This means that it remains possible to single out individual members when querying the database with sufficient background information, like family and medical history.

Therefore an environment based on a masked database, although in itself providing much more protection against data leaks, should be treated with the same security measures as the production environment itself.

4 Security Considerations for Developers

OHI customers each have their own implementation of OHI Back Office and their adjacent application setup to cater for their own individual requirements. OHI Back Office provides many 'hooks' for custom code to finetune the core processes of the system.

Also customers create custom developed software to directly access and manipulate OHI Back Office data or integrate with other applications.

Doc[5] provides a reference for custom development.

This chapter provides some initial information for developers about how to create secure applications for OHI Back Office, and how to extend OHI Back Office without directly compromising security.

You are well advised to research multiple sources to become more aware of technology related risks. A visit to www.owasp.org is a good starting point.

4.1 Creating Custom Schemas

You are free to create your own custom applications on top of OHI Back Office, provided you do not make any changes to the OHI Back Office schema object definitions and associated system and object privileges. This means you will use a custom schema to interact with the OHI Back Office schema. Such a schema can receive a set of allowed object privileges on OHI Back Office schema objects through a standard documented grant routine.

It is not allowed to change these privileges and these are strictly checked and enforced during each OHI release installation activity.

This means with a custom schema, you do not have all the privileges of the OHI BO owner schema, but you will have sufficient privileges to use most of the standard OHI BO functionality:

- select privileges on the majority of functional and technical OHI tables;
- fine-grained query access on sensitive columns of selectable, sensitive data containing, functional OHI tables;
- fine-grained insert and update access on columns of user DML allowing functional OHI tables (this access is not granted on OHI tables which are fully application maintained)
- table API's to access (retrieve and manipulate) functional OHI tables;
- a limited list of additional PL/SQL packages, procedures and functions.

You may have multiple custom applications interfacing with OHI Back Office, for example a self-service portal, custom batch processing scripts, and an interface which processes CRM data into OHI Back Office.

A custom schema per interfacing application makes it easier to manage these different applications and split the post-installation work and testing efforts whenever a new OHI Back Office release is installed.

The custom schema's should only be used to store highly trusted code, given the wide OHI Back Office access they have. Never should a connection to these schemas be used in a production environment for directly querying or manipulating OHI Back

Office, these accounts should only store the custom application logic to implement custom code batch processes or implement a custom OHI Back Office application interface component.

These custom code schema accounts should be locked and only be unlocked for maintenance purposes, when they are used for deploying new custom code component versions.

4.1.1 Provide OHI realm authorization for custom code schema's

When Oracle Database Vault is implemented with the OHI specific realm you need to determine what to do regarding custom code objects.

When you use definer rights privileges in stored pl/sql code you need to provide realm authorization for your custom code account(s) to make sure the code can access the OHI BO objects.

You should also consider adding the custom code to the realm definition to protect the custom code objects from being changed by any account with sufficient system privileges to do so (typically DBA accounts).

4.2 Creating Client Applications for HSL Services

HSL services are designed to support typical self-service tasks such as creating a new member policy, registering a change of address, viewing a members' claims history etc.

The contents of a request is deserialized into a Java object and then mapped to a SQL type before handed to the PL/SQL implementation of the service operation.

When calling a HSL service operation, SQL injection should not be a problem:

- All parameters are type-checked. String values are not only checked for length but must also match with a predefined regular expression.
- All queries are done with bind variables. SQL injection is only a risk if the query is created on the fly.

4.2.1 Be careful with Basic Authentication

When using Basic Authentication the client sends a HTTP header with WLS credentials for every request. This makes Basic Authentication less safe, even for using in an intranet environment.

Consider using OAuth2.0 (see **Doc[2]**) for improved security.

4.2.2 Do not use 'hsl.xxx.developermode=true' in a Production System

When developing client applications you can set the HSL property `hsl.xxx.developermode=true` to include error messages in the HTTP response.

When taking the application to production, you should ensure that `hsl.xxx.developermode=false`.

The error message will still be logged, but the HTTP response will not provide the entire error message anymore.

4.2.3 Restrict hsl.xxx.allowedorigins

Cross-origin resource sharing (CORS) happens if a client application on server 'A' requires to access server 'B' to invoke an HSL service operation.

Before invoking the desired service operation, the client should first send an OPTIONS request to the service URL to receive an `Access-Control-Allow-Origin` header with a list of allowed origins.
If the server hosting the client application is not listed as an origin, an error should occur.

It is tempting to use a wildcard in `hsl.xxx.allowedorigins` or create an extensive list of potential servers. However, you should try to keep the allowed origins list as brief as possible to reduce the risk of another application hijacking the HSL service.

4.3 Creating Client Applications for SVL Services

SVL services provide an API for listing, retrieving and writing OHI Back Office data.

SVL services are typically used for application integration and much less to support front-end applications.

They are SOAP/HTTP web services with a document-style interface (XML format). The XML contents of a request is deserialized into a Java object and then mapped to a SQL type before handed to the PL/SQL implementation of the service operation. The attribute values of SQL types may be used as bind variables for SQL queries but are never inserted literally into a query. Therefore, the risk for SQL injection is extremely small.

4.3.1 Be careful with Basic Authentication

By default, SVL services use Basic Authentication. This means that any (!) authenticated Weblogic user may invoke a SVL service operation unless configured differently.

When deploying a SVL service you should consider strongly to create specific WLS user(s) to access the SVL services and to set up the WLS security realm to allow only those users to access the SVL services, or even a limited set of specific web service operations.

4.4 Custom Development in PL/SQL and SQL

Custom developed PL/SQL and SQL should be created in a custom schema.

Many OHI Back Office customers create their own SQL scripts to be run by the OHI Back Office Batch Scheduler.

4.4.1 Preventing SQL Injection

SQL injection may occur if user input is used to dynamically create and execute SQL queries. You may prevent the risk of SQL injection by using explicit parameterized SQL cursors in PL/SQL. This means that variables and parameters with user input are used as 'bind variables' and may not alter the query text itself.

If this is for some reason not possible make sure you always validate the input. Consider using the validations as offered by the DBMS_ASSERT package.

4.4.2 Implement input validation

If your application has a user interface, implement input validation and strong data typing to minimize the risk that inappropriate values are entered.

4.4.3 Use SEPS for Command Line Invocation

Avoid passing username and password on the command line when executing custom code. Instead consider using an account-alias which can be looked up in a Secure Encrypted Password Store (SEPS). Of course all precautions regarding the access to a SEPS file, as described earlier in this document, should be taken into account.

5 Appendix A – Additional Notes

5.1 Password Aging

Password aging helps to make the application more secure. It should not be used for the service accounts mentioned below unless a strict procedure is implemented that changes the passwords in time:

- 'batch' database account
- Web service accounts for HSL, PSL and SVL services
- Data source connection pool accounts for accessing database AQ queues

The passwords for these accounts should always be periodically changed in the database in combination with a change in the associated connection pools and a restart of the service.

Consider using the rolling password change functionality introduced in database 19.12 to mimize any service interruption as result of an executed password change not yet implemented in the different password stores.

6 Appendix B – Securing Oracle Setup Manager

6.1 Oracle Setup Manager

Oracle Setup Manager (OSM) can be used to copy the setup data of OHI Back Office from a source environment to one or more target environments. This works by generating files with an export of relevant setup data from the source environment and importing these in the target environment.

The aim of OSM is to setup OHI Back Office environments (ultimately the production environment) in a repeatable, controlled and auditable manner.

Therefore, it is important to secure this process by strictly limiting access to the functions and artefacts involved in updating the setup data and subsequently exporting and importing it.

The following security measures should be implemented, on top of the measures to secure the source and target environment described elsewhere in this document.

Securing the use of OSM starts with the source environment:

- Limit the number of users that have access to the source environment (both direct database access and using the forms application) to only those users that are responsible for (and are allowed to) changing the setup data. Do not solely rely on function authorization, but lock or remove the accounts for all others users (which might be present when the environment is cloned from another environment).

- Use separate roles in OHI that allow access to the setup and configuration screens. Remove (write access in) these screens from other roles.

- Create a separate role for exporting OSM data and grant this role to even fewer users. Only users that are allowed to transfer the setup data to a different environment should have this role.

- Activate mutation logging for the configuration tables and the setup manager tables.

- Only spin up the source environment when it is needed and bring it down when the changes have been completed.

- Do not allow access to custom code schemas in the source environment to developers. Only technical managers should have privileges.

The files that are generated should be protected:

- Generate/export these to a separate Database Directory that is not granted to public in the database and has very limited access on the database server.

- Transfer the generated file to the target database server in a secure way, so use secure copy or SFTP and whenever possible prevent storage of files in between source and target. Especially prevent downloading to for example local storage on a PC or, even worse, to a portable non-encrypted USB device.

- Import in the target environment from a separate Database Directory that is not granted to public in the database and has very limited access on the database server.

- Always compare the file hash with the value in the source environment before importing the file.

- Limit the OS file access for the file system locations that are used as export and import database directories. Only the account "oracle" and the account used for transferring the files should have access.

In the target environment(s), and especially in Production:

- Only grant privileges on the import functionality and import reporting functionality of OSM, using a separate role. Do not grant privileges on the other OSM functionality to any role.

- Do not grant write privileges on the regular setup screens. OSM import should be the only way to change the setup.

Some general measures:

- Because custom code could manipulate setup data, use peer review to check for malicious manipulation of setup data and only allow installation using an official procedure.