ORACLE
INSURANCE

Oracle Health Insurance Back Office

Security Guide

Version 1.5

Part number: F27899-01

May 1, 2020

ORACLE

Copyright © 2018, 2020, Oracle and/or its affiliates. All rights reserved.

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

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

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

U.S. GOVERNMENT RIGHTS

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

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

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

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

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

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

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

CHANGE HISTORY

Release	Version	Changes
10.18.1.0.0	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

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 (CTA13508)

Doc[2] Oracle Health Insurance Back Office HTTP Service Layer - Installation & Configuration Manual (CTA13681)

Doc[3] Oracle Health Insurance Back Office – SOAP Service Layer Installation & Configuration Manual (CTA13651)

Doc[4] Oracle Health Insurance Back Office – Reading, Writingd and Authorizing
(CTA13531)

Doc[5] Oracle Health Insurance Back Office - Custom Development (CTA13678)

Doc[6] Oracle Health Insurance Back Office - Object Authorization (CTA13533)

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

Contents

1 OHI Back Office Security Overview 7

1.1 Basic Security Considerations 7
1.2 Overview of OHI Back Office Security 7
..1.2.1.1 Product Terminology 8
1.3 Understanding the OHI Back Office Environment 9
1.4 Recommended Deployment Configurations 9
1.4.1 Forms Services Deployment 9
1.4.2 Web Services Deployment 11
1.4.3 OHI Back Office JET Deployment 12
1.5 SVL Services Security 13
1.6 HSL Services Security 14
1.7 Batch Processing Security 15
1.8 Operating System Security 15
1.9 Oracle Database Security 16
1.10 Oracle Fusion Middleware Security 17
1.11 Forms Services Security 17
1.12 OHI JET Security 18
1.13 Net Services Security 18
1.14 Oracle Security Documentation 18

2 Performing a Secure OHI Back Office Installation 19

2.1 Pre-Installation Tasks 19
2.2 Installing OHI Back Office Securely 19
2.3 Post Installation Tasks 19
2.3.1 Changing Default Passwords 20

3 Implementing OHI Back Office Security 21

3.1 Implementing SVL Services Security 21
3.1.1 Server-Side Authentication 21
3.1.2 Client-Side Authentication 21
3.1.3 Server-Side Authorization 22
3.1.4 Client-Side Authorization 22
3.1.5 Access Control 22
3.1.6 Data (transport) Integrity 23
3.1.7 Security Audit And Alarms 23
3.2 Implementing HSL Services Security 23
3.2.1 Server-Side Authentication 23
3.2.2 Client-Side Authentication 23
..3.2.2.1 Basic Authentication 24
..3.2.2.2 Token Validation For OAUTH2 24
3.2.3 Server-Side Authorization 25
3.2.4 Client-Side Authorization 25
..3.2.4.1 Basic Authentication 25
..3.2.4.2 Authorization through OAUTH2 25
3.2.5 Access Control 26
3.2.6 Data (Transport) Integrity 26
3.2.7 Security Audit and Alarms 26
3.3 Implementing Batch Processing Security 26
3.3.1 Identification / Authentication 26
..3.3.1.1 Authentication at OS level 27

..3.3.1.2 Identification / authentication in the Oracle database 27
3.3.2 Authorization 27
..3.3.2.1 Authorization at OS level 27
..3.3.2.2 Authorization at database level 28
..3.3.2.3 Authorization at application level 28
3.3.3 Access Control 28
3.3.4 Data (transport) integrity 28
3.3.5 Security audit and alarms 29
3.4 Implementing Operating System Security 29
3.5 Implementing Oracle Database Security 29
3.5.1 Identification / Authentication 29
..3.5.1.1 OHI object owner accounts 30
..3.5.1.2 Personal and other database accounts 30
..3.5.1.3 BATCH processing account 31
..3.5.1.4 DBA and other privileged accounts 31
3.5.2 Authorization 32
..3.5.2.1 Personal accounts for users of the OHI Back Office Forms GUI 32
3.5.3 Data (Transport) Integrity 33
3.5.4 Security Audit / Alarms 33
3.6 Implementing Oracle HTTP server Security 33
3.6.1 Identification / Authentication 33
3.6.2 Authorization 33
3.6.3 Access Control 34
3.6.4 Data (Transport) Integrity 34
3.6.5 Security Audit and Alarms 34
3.6.6 End-user identification / authentication 34
3.6.7 Client-side identification / authentication 34
3.6.8 Server-side identification / authentication 34
3.6.9 Authorization 34
3.6.10 Access Control 34
3.6.11 Data (transport) Integrity 34
3.7 Forms Services Security 34
3.7.1 Identification / authentication 34
3.7.2 Authorization 34
3.7.3 Access Control 35
3.7.4 Data (transport) Integrity 35
3.7.5 Security Audit and Alarms 35
3.8 Net Services Security 35
3.8.1 Identification / Authentication 35
3.8.2 Authorization 36
3.8.3 Access Control 36
3.8.4 Data (transport) integrity 36
3.8.5 Security Audit and Alarms 36

4 Security Considerations for Developers 37

4.1 Creating Custom Schemas 37
4.2 Creating Client Applications for HSL Services 37
4.2.1 Be careful with Basic Authentication 38
4.2.2 Do not use 'hsl.xxx.developermode=true' in a Production System 38
4.2.3 Restrict hsl.xxx.allowedorigins 38
4.3 Creating Client Applications for SVL Services 38
4.3.1 Be careful with Basic Authentication 38
4.4 Custom Development in PL/SQL and SQL 39
4.4.1 Preventing SQL Injection 39
4.4.2 Implement input validation 39
4.4.3 Use SEPS for Command Line Invocation 39

5 Appendix A – Additional Notes 40

5.1 Password Aging 40

1 OHI Back Office Security Overview

This chapter provides an overview of OHI Back Office security.

1.1 Basic Security Considerations

The following principles are fundamental to using any application securely:

- Keep software up to date. This includes the latest product release and any patches that apply to it.
- Limit privileges as much as possible. Users should be given only the access necessary to perform their work. User privileges should be reviewed periodically to determine relevance to current work requirements.
- Monitor system activity. Establish who should access which system components, and how often, and monitor those components. 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: http://www.oracle.com/technetwork/topics/security/alerts-086861.html

1.2 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 of the individual insurance members. The larger OHI Back Office implementations have a population of 2-4 million members.

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 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.

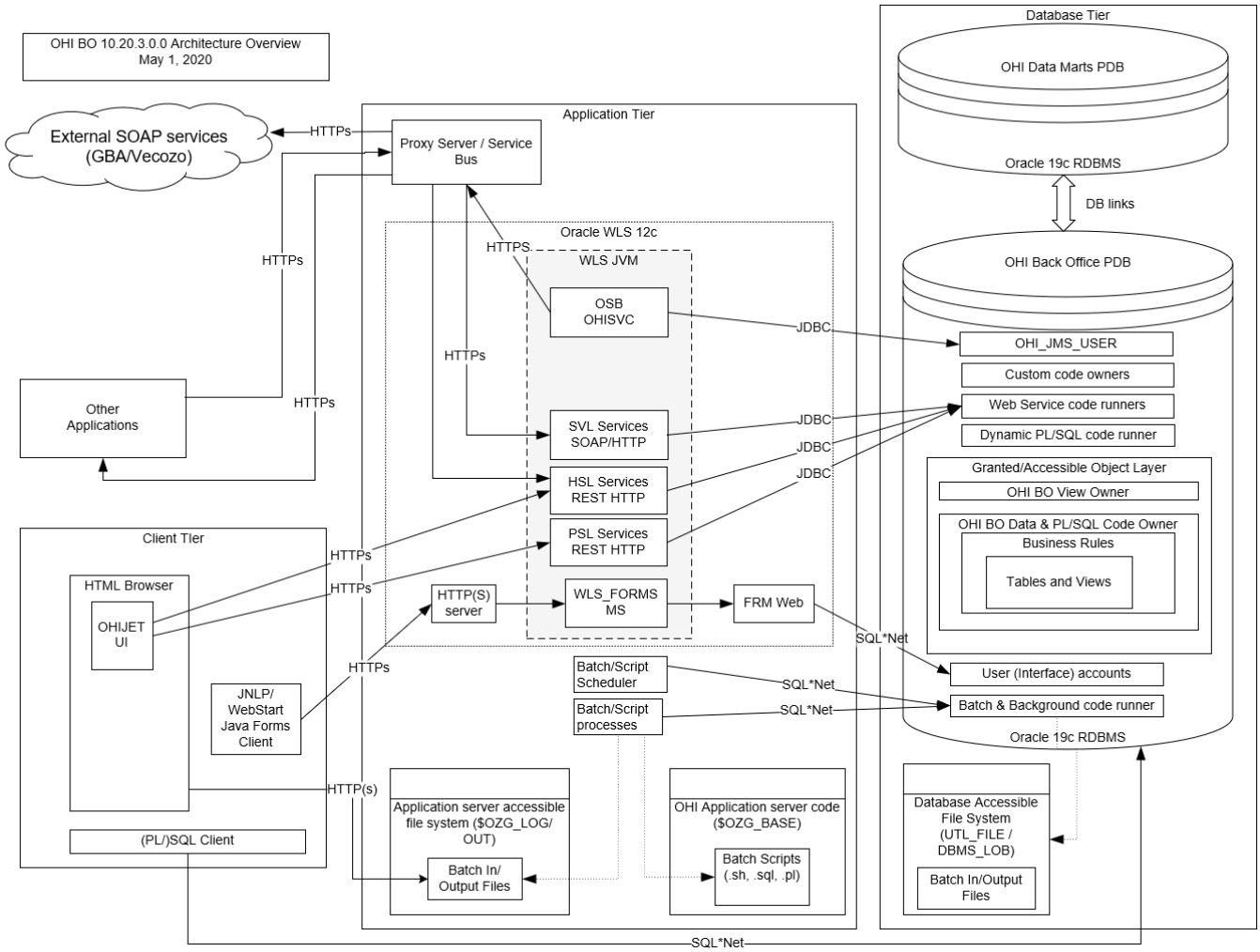


Figure 1 (MME 116285)

The Oracle Forms application to control the system is accessed by back office users on the health insurance provider's intranet. Each back office user has an Oracle database account to login 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 Basic Authentication over SSL. OAUTH2 is supported for REST services.

Most OHI Back Office customers use a custom database schema for custom developed software to access OHI Back Office data and PL/SQL packages. The OHI Back Office application is designed to maintain data integrity when data is manipulated directly through SQL.

..1.2.1.1 Product Terminology

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);

- 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.

1.3 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 high 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.

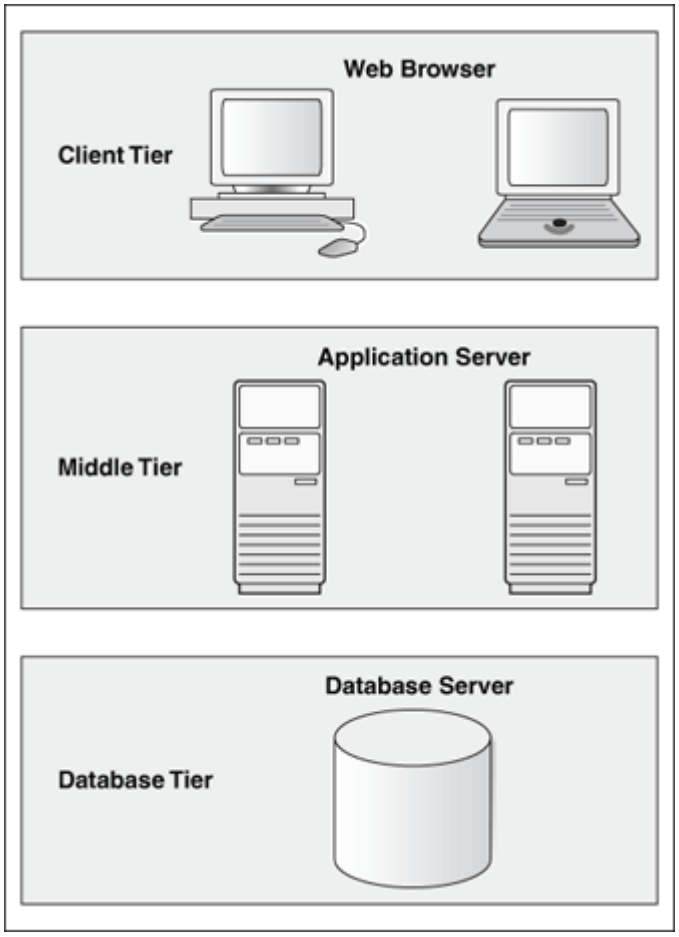
When planning your OHI Back Office implementation, consider the following:

- 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 (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 resources 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.
  Another risk is financial damage resulting from fraud committed by an external entity or, probably a higher risk because of already having a form of access, OHI Back Office users or administrators.

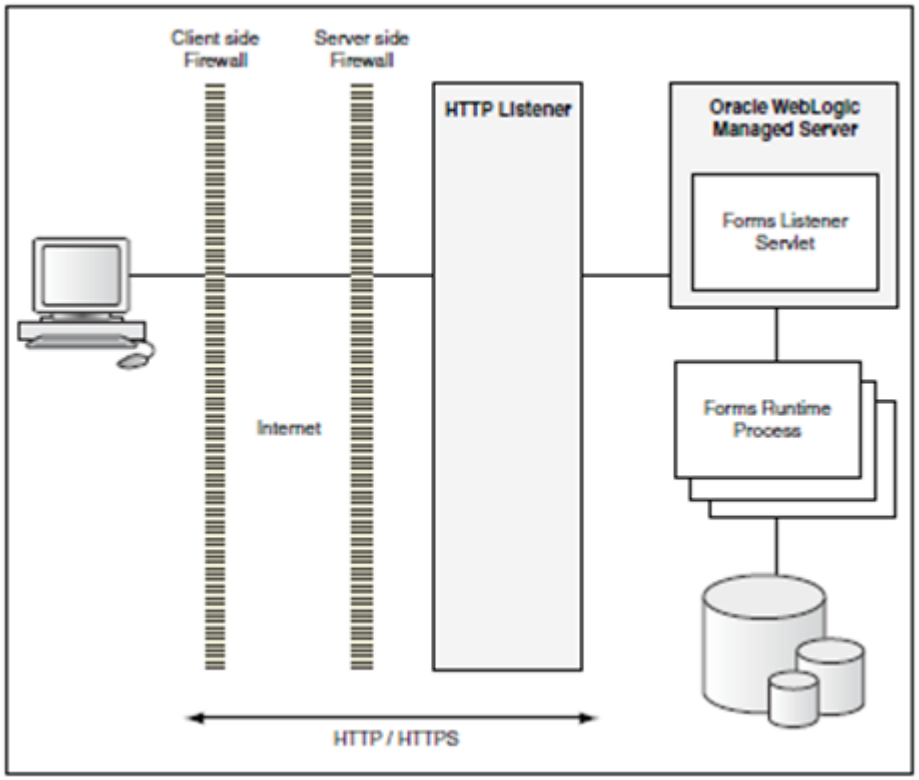
1.4 Recommended Deployment Configurations

1.4.1 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:



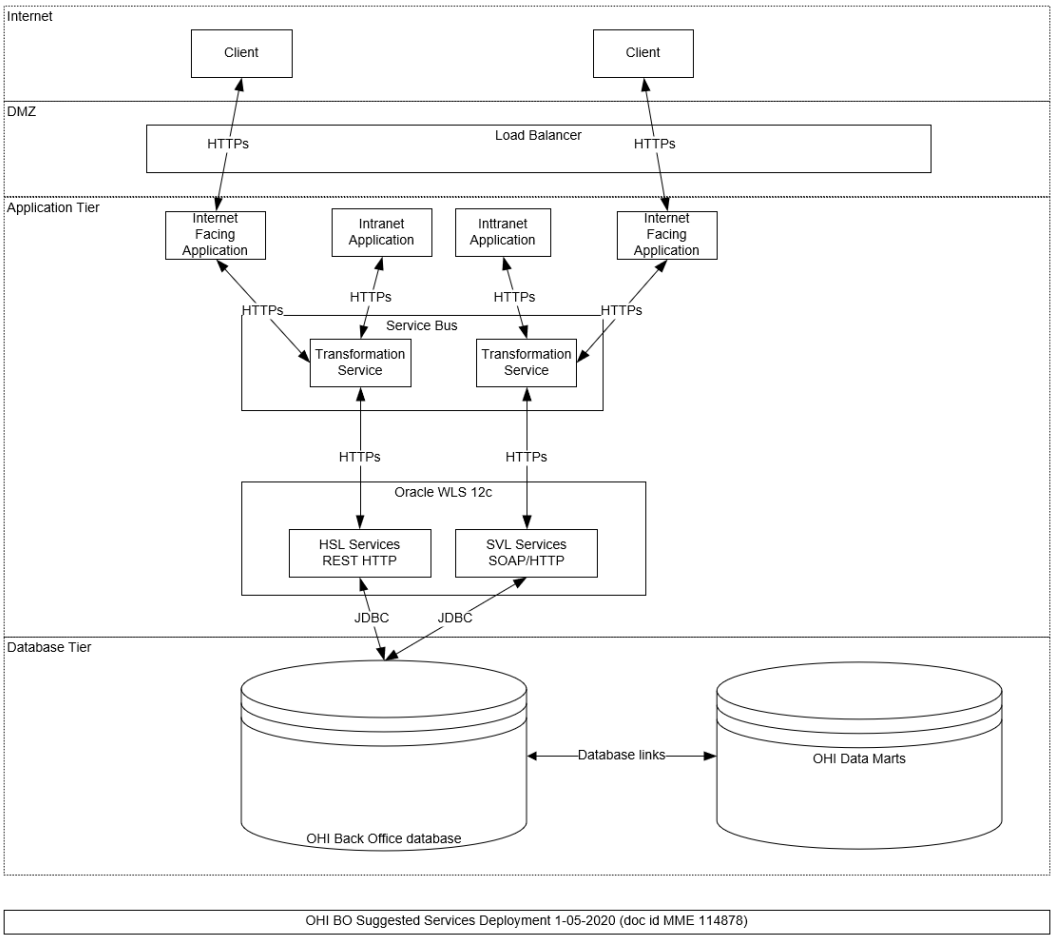
However, direct Internet access for back office users should not be offered. A VPN connection would provide much better protection against unauthorized access or overload of the application.

1.4.2 Web Services Deployment

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

- Make a security risk assessment for each client application invoking OHI Back Office services and take appropriate action to mitigate these risks.
- 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 operation or data authorization. Make sure you only authorize these operations that are needed and implement an authorization structure within your calling application.
- The application server to which the web services are deployed should never be directly accessible from the internet.
- 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 Basic Authentication of the HSL and SVL services would then also be restricted to the proxy services.
- If OHI Back Office web services may be needed in some way indirectly over the internet, make sure you implement an application layer that provides sufficient authorization towards the authenticated users. 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:

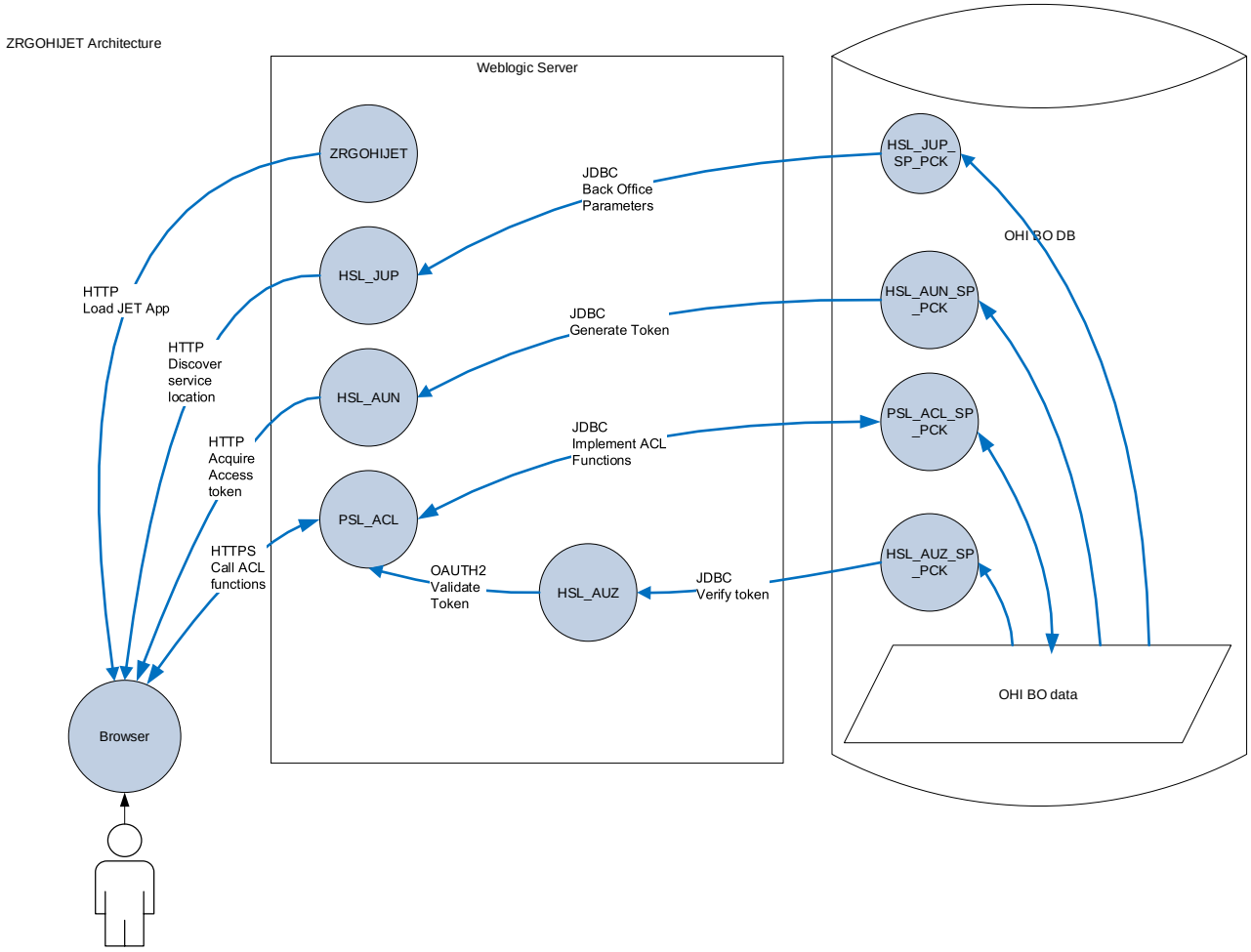


1.4.3 OHI Back Office JET Deployment

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 OAUTH based authentication and authorization architecture shown in the following diagram.

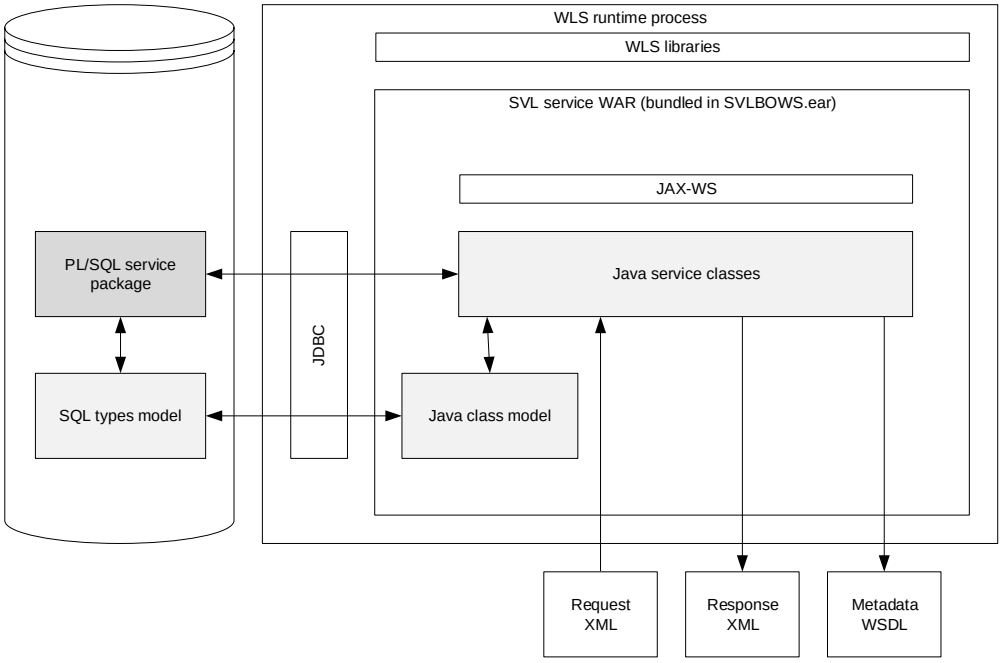
The technology used is identical as used in the HTTP RESTful Service Layer, referenced in short as HSL. The main difference is that for the PSL implementation by definition OAUTH2 is used. For the rest of this document this means that all security considerations that apply for HSL services also are applicable for PSL services.



1.5 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).

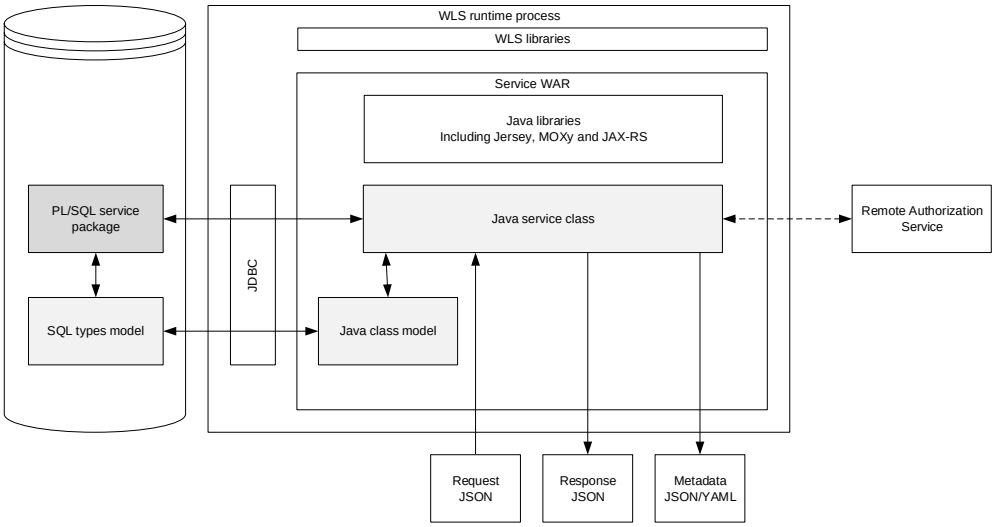
The default client-side security mechanism for SVL services is Basic Authentication over SSL. The default mechanism can be overruled with WLS policies and roles.

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

1.6 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 Basic Authentication over SSL.
OAUTH2 is supported for fine-grained authorization. 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, error messages are suppressed in the HTTP response and appear only in the application log.

1.7 Batch Processing Security

Much of the work in OHI Back Office is performed through background processing. An example of the 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)
- Calling remote web services from within the database (by means of AQ queues to communicate with an OSB project which implements the calls instantaneously).

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.

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

The invidual batch scripts write 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.

The batch scheduler is started from the command line by a technical application administrator.

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

1.8 Operating System Security

A typical OHI Back Office production environment consists of

- A database server with Oracle 19c database running the OHI Back Office OLTP database and the OHI Data Marts warehouse database.
- 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').
- 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) should be limited to the file owner (read/write) and the OS user 'batch' (read-only).

- sFTP or a similar mechanism will be used to upload ASCII and XML files to the application server to be processed by the OHI Back Office batch scheduler.

- The application server should not be accessible from the Internet.

See the following documents:

- Guide to the Secure Configuration of Red Hat Enterprise Linux 7

- Hardening Tips for the Red Hat Enterprise Linux 7

1.9 Oracle Database Security

All OHI Back Office data are stored in a single schema in an Oracle 19c pluggable database for optimal data consistency.

A robust business rule layer implemented in PL/SQL protects the integrity of OHI Back Office data.

Role-based security is implemented to grant the minimum set of privileges to authorized users. Users of the Forms GUI are dynamically granted privileges just for the duration of a session.
This 'hardening' of the database allows other programs than the OHI Back Offce forms application to access and manipulate OHI Back Office data safely and consistently.

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 zip compression of files created in database directories.

- Oracle AQ queues are used as a technical transport mechanism to implement calls from within the database to web services in the outside world. An 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 OSB project, which use is optional.

- It is recommended to use another Service Bus 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.

- The OHI Data Marts database uses a database link to read data from the OHI Back Office database where a database link to OHI Data Marts is used to call the actual ETL processes.

- The OHI Back Office and OHI Data Marts databases contain PI and PHI data which must not be accessed by unauthorized users.

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

1.10 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 etc) .
- Forms Server listener servlet and Forms Server (see 'Forms Services' later in this chapter).
- WebLogic application server (WLS) for deploying web services (HSL services and SVL services).
- Oracle Service Bus (OSB), as optional component, running on WLS.

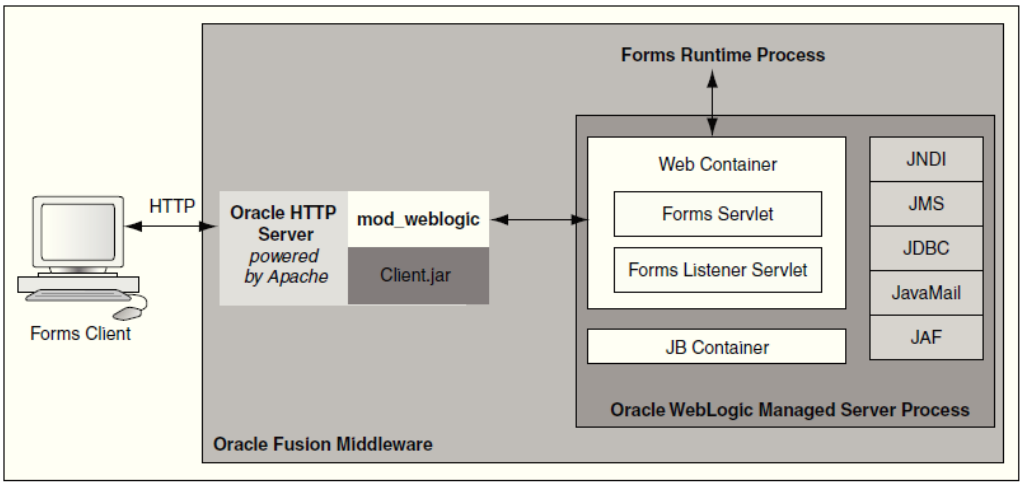
In most cases the same server is used for FMW and batch processing. It is therefore very likely that PI and/or PHI data will be processed on the application server(s). This means that the FMW instance(s) must be well secured.

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

1.11 Forms Services Security

Much of the OHI Back Office functionality is managed through a 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.
FMW also includes the WLS 12c instance to which the OHI Back Office web services are deployed.



The Forms server listener creates a Forms runtime process for each new 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 or in a separate Java process if the JNLP plugin is used. The Forms Client communicates with the forms session through HTTP(S).

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 data (see 'Net Services' below).

- Set `FORMS_RESTRICT_ENTER_QUERY=true` to prevent SQL injection.

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

1.12 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.

Authentication for this separate UI is fully based on the database accounts with accompanying OHI officer accounts as 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.

1.13 Net Services Security

The OHI Back Office database is 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 database.

- Change listener configuration to restrict listener access to trusted clients and systems.

1.14 Oracle Security Documentation

To implement security, Health Insurance Back Office uses other Oracle products, such as Oracle Database and WebLogic Server. 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 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]**.

2.1 Pre-Installation Tasks

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

- Install and harden application server.
- Install and harden 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 strategy for logging changes to data in the OHI Back Office tables.
- Define a back up strategy which does not compromise the security of OHI Back Office production data.
- Implement Transparent Data Encryption (TDE).

2.2 Installing OHI Back Office Securely

When installing OHI Back Office components:

- Do not install any components you do not need. For example, only deploy 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.
- 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.
- Ensure that sensitive data is masked or anonymized in non-production data.
- Remove any unneeded database links.
- Notify your Data Protection Officer.

2.3.1 Changing Default Passwords

If you have implemented a password policy, you should have got rid from 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 creates a database connection to the database.

The database connection should use an account specifically created for SVL services which has a minimal set of object privileges.

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 ohiDS).
- 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:

- The passwords for the datasources are stored in encrypted form in the Weblogic configuration and are known to WLS administrators.
  Action: ask the application / database administrator to enter the passwords for data sources (see notes about 'Password Aging' in Appendix A).

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 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 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 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:

- 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.

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:

- 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.

3.2 Implementing HSL Services Security

3.2.1 Server-Side Authentication

The web service creates a database connection to the database.

The database connection should use an account specifically created for HSL services which has a minimal set of object privileges.

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 ohiDS).

- 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:

- The passwords for the database connections are stored in encrypted form in the Weblogic configuration and are known to WLS administrators.
  Action: ask the application / database administrator to enter the passwords for data sources (see notes about 'Password Aging' in Appendix A)

3.2.2 Client-Side Authentication

Starting with OHI BO 10.18.1.0.0, all HSL operations support Basic Authentication (by default) and OAUTH2.

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’, ‘TOKEN’ or ‘NONE’.
    - ‘`hsl.app.authorization=NONE`’ may only be used for the HSL_AUN ‘postToken’ operation. This operation logs into the database with the credentials of the OHI Back Office user and returns a JWT token to be used in OAUTH validation.
    - If TOKEN or NONE is allowed 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 HTML 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 directly called by end users!

..3.2.2.1 *Basic Authentication*

This is the default 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.
  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.

..3.2.2.2 *Token Validation For OAUTH2*

Starting with OHI release 10.18.1.0.0 it is possible to use OAUTH token validation.

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

Beware, for the OHI JET based user interface token validation for OAUTH2 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 authorized 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 JWT
- The `hsl.app.tokenvalidation` properties in the hsl.properties file must be set to configure the token validation call.

3.2.3 Server-Side Authorization

The web service should run under 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:

- 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 can be used instead of Basic Authentication.

..3.2.4.1 *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.

..3.2.4.2 *Authorization through OAUTH2*

When using OAUTH2 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 offers the following advantages over basic authentication:

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

- Fine-grained authorization

Risks / measures:

- 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.

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.

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.

3.2.7 Security Audit and Alarms

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

3.3 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).

3.3.1 Identification / Authentication

..3.3.1.1 *Authentication at OS level*

The OS user logs in as UNIX user 'batch' using a password to be set by the system administrator.

..3.3.1.2 *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 manipulate OHI Back Office data.
  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.
  - Consider a logon trigger to reject sessions from unauthorized systems such as desktop computers.
    The implementation of a logon trigger is described in **doc[1]**.

3.3.2 Authorization

..3.3.2.1 *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.
  Not viewable by back office users.

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. As these files may contain sensitive data some way of secure file access needs to be implement.

..3.3.2.2 *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 packge when an actual request for such a process has been entered and reached the phase 'starting' so it is not possible to run 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 needed since the database batch account is authenticated through SEPS at the OS level by anyone (any 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 access are prevented from having access to the OHI Back Office data.

..3.3.2.3 *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.

3.3.3 Access Control

You can only logon as UNIX 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.

3.3.4 Data (transport) integrity

The 'data' produced by the batch scheduler and its associated scripts which 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).

3.3.5 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.

3.4 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

3.5 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 of OHI Back Office data with a robust business rule implementation layer in PL/SQL.

The OHI Back Office application contains PL/SQL packages for creating 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.

3.5.1 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')
- Personal accounts, standard with minimal create session privileges, for individual OHI Back Office Forms GUI users. These accounts are dynamically given additional privileges to OHI Back Office objects for the duration of a session by the Forms GUI.
- A 'batch' account for batch processing.
- DBA and other privileged accounts (like database accounts with special OS privileges like SYSDBA).
- Custom schemas used for custom development.
- A JMS queueing account, for access to OHI provided AQ queues with a JMS payload.
- Service accounts used by HSL and SVL services (these accounts have only access to service supporting code objects).

..3.5.1.1 *OHI object owner accounts*

The table owner account is a very privileged account which contains all data objects and many code objectes. Through this account data and code can freely be manipulated.
The view owner and the dynamic pl/sql owner account can be used to freely access almost all data.

Risks / measures

- Risk: password is known to too many persons.
  Action: make sure the password is sufficient complex so it cannot easily be remembered and changed regularly.
- Risk: account is used for other purposes than OHI release maintenance.
  Action: By default lock the standard 3 schema accounts as they should only be needed when OHI patch/release installations are executed.

..3.5.1.2 *Personal and other database accounts*

This includes users of the OHI Back Office Forms GUI, custom development schemas and individual users accessing the OHI Back Office database with a client tool (for example SQLPLUS). The database administrator will create Oracle database accounts to login to the OHI Back Office database.

Risks / measures

- Risk: default passwords are not changed.
  Action: expire passwords when created, implement password expiration, setup a policy for enforcing a certain complexity for passwords and lock accounts during longer periods in which they are not needed.
- 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 immediately and/or delete these accounts.

- 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 criteria (length, different to old 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 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 enable the OZG_ROL role.

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

  - Lock account when not needed; only use account in production for deployment.

- 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; also make sure the configuration files are stored with minimum access rights and use encryption in tooling whenever possible.

..3.5.1.3 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.

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

..3.5.1.4 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 gice access to DBA accounts. 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 privileges assigned to these accounts. Action: 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.

3.5.2 Authorization

New accounts only may login 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.

In the case of the personal accounts for Forms GUI users this is done dynamically upon login 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 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!

..3.5.2.1 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. Therefore OZG_ROL should never be granted to a database user.

3.5.3 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 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.

3.5.4 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, deleter or changed.

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

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

3.6 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.

3.6.1 Identification / Authentication

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

3.6.2 Authorization

The Apache 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.

3.6.3 Access Control

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

By default no restrictions are applied to access to the Apache server. It is worth considering restrictions in larger organizations.

3.6.4 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.

3.6.5 Security Audit and Alarms

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

3.6.6 End-user identification / authentication

No user authentication is used for OHS.

3.6.7 Client-side identification / authentication

Not used.

3.6.8 Server-side identification / authentication

Not used.

3.6.9 Authorization

Not used.

3.6.10 Access Control

Not used.

3.6.11 Data (transport) Integrity

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

3.7 Forms Services Security

3.7.1 Identification / authentication

The user identifies 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.

3.7.2 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.

In specific cases the information that is shown to the user is determined dynamically when the information is displayed based on the authorization.

3.7.3 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'.

3.7.4 Data (transport) Integrity

Sending a HTTP(S) request to the Apache 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.

3.7.5 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.

3.8 Net Services Security

3.8.1 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.

3.8.2 Authorization

Not applicable.

3.8.3 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.

3.8.4 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.

3.8.5 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.

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 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.

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.

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.

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 unsafe for use over the Internet.

As you already need to ensure that the HSL application is never called directly this should not be any issue and even for user over the intranet the use of OAUTH2 (see **Doc[2]**) may be considered.

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 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 (!) Weblogic user may invoke a SVL service operation.

When deploying a SVL service you should consider 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 SQL cursors. This means that variables and parameters with user input are used as 'bind variables' and may not alter the query itself.

If this is for some reason not possible make sure you 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 Envrypted 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 like:

- 'batch' database account
- Web service accounts for HSL, PSL and SVL services

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