ORACLE
INSURANCE

Oracle Health Insurance Back Office

Reading, Writing and Authorizing Oracle Health Insurance Application Files and Messages

Version 1.17

Part number: F50672-01

January 14, 2022

CHANGE HISTORY

Release	Version	Changes
10.12.2.0.0	1.2	• Changed examples in Reading output
10.14.2.0.0	1.3	• Some minor adjustments
10.15.1.0.0	1.4	• Removed references to Oracle Reports
10.15.3.0.0	1.5	• No relevant updates, only republished
10.16.1.0.0	1.6	• No changes
10.16.2.0.0	1.7	• Some minor textual improvements
10.17.1.0.0	1.8	• No changes
10.17.2.0.0	1.9	• No changes
10.18.1.0.0	1.10	• Added information regarding the isolation of database directories of pluggable databases in database 12.2 through PATH_PREFIX
10.18.2.0.0	1.11	• Republished with different part nr.
10.19.1.0.0	1.12	• No changes. Republished with different part nr.
10.19.2.0.0	1.13	• No changes, republished with different part nr.
10.20.1.0.0	1.14	• No changes, republished.
10.20.8.0.0	1.15	• Title adjusted as next to files also messages are involved • Content extended for queued output messages
10.21.1.0.0	1.16	• Added information regarding queued JSON and XML output messages • New part number.
10.22.1.0.0	1.17	• 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)

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

CONTENTS

Introduction

File output is mostly generated within the Oracle Health Insurance application using the so-called *batch or script functionality.* This at least applies to most files being generated. Files often contain many messages for different message topics (policies, claims, etc.) in one and the same file.

Message output may also consist of messages but typically with one message per topic (claim, policy, relation, etc.), mainly in XML or JSON format, typically published on a JMS queue.

The files involved are either stored on the database server or the application server, in a directory that is to be specified per request or pre-configured.

The bulk of file output is the XML file output, which is to be generated by certain batches since the first release in 2005 onwards. You can specify your own file name and 'database directory' (from a pre-defined list) for the XML output (and XSD output if applicable) per script request. The technical log messages from the batch are, however, saved to the regular pre-configured output location on the application server. The system management department of an OHI customer determines the actual locations of these directories and how you can access these.

In addition, online help information stored in files in HTML format is used within the application, started from within the user interface and accessed through the web browser and the OHI web server which provides access.

Next to that, the documentation files for the delivered (patch) releases can be accessed and viewed in the same way from within the application (the 'releases' window).

This document deals with the functional aspects of these application files (layout, configuration and use) as well as of the application messages, mainly published on a JMS queue.

Requesting output from a script

By default a script request creates a standard output file. The location and naming of this file is described below.

The output for XML output-producing batches, which have been realized in increasing numbers since the first OHI release in 2005, is written to separate files (not the standard output file for a batch). Please refer to the chapter that deals with this subject in detail for more information.

Location of application software

The Oracle Health Insurance batch scheduler processes the script requests submitted by the application.

Various types of modules are started to this end, including shell scripts, SQL*Plus modules, perl scripts, etc.

The “Directory environment" field in the "System/Management/General/System parameters" screen (SYS1010F) should specify the directory under which the modules to be started exist in the specified subdirectory structure. Often environment variable $OZG_BASE is used but you may of course choose another approach and corresponding folder, specified hard coded or through an environment variable.

See the following document for the mandatory directory structure:

Oracle Health Insurance Back Office Installation, Configuration and DBA Manual

Naming

The following conventions apply for naming files produced by Oracle Health Insurance script requests:

Filename structure

The file name is built up from a script request number and an extension:

```
<script number in a maximum of 14 positions> + <extension>
```

Example

```
10000000041774.out
```

Extensions

The extension can be one of the following:

.out

The extension for output, used for statuses “Completed” (script request successfully executed) and “Error” (functional errors have occurred).

.log

The extension for the log file which is populated on the "Failed" status (technical error).

Authorization

File authorization ensures that the output can only be *read*. **System authorization** is used to ensure that this is only done by individuals who are authorized to do so.

File authorization

General

The Oracle Health Insurance Batch Scheduler runs under a specific OS-account (normally '`batch`'). Then, all output files are created and saved under this account.

The default properties of the files created are determined by the OS file mode creation mask `umask`. When this is set to `umask 333` (preferably in `$OZG_ADMIN/ozg_init.env`), all files created are assigned *read-only* properties.

When the setting is used the result is that end users who then open the file at operatings system level can only *read* it. As the basic property of the files is read-only the files *cannot* therefore be changed.
Here it does not matter whether file **browser** functionality (e.g. browsers like MS Internet Explorer or Edge, Chrome) or file **editor** functionality (e.g. vi under UNIX, MS Word, Wordpad, PFE under Windows) is used to display the output.
Even if the files are accessed under Windows (using networking software such as NFS or Samba) through a read-only mounted share they can still only be *read.*

Application authorization

Application authorization means that end users can only access their own output through a created URL, unless he/she has authorizations as an administrator post holder (in the *Maintain functions with roles* screen); in that case they can view the output of other users also.

If a post holder has administrator authorization he/she can also view script requests from other post holders.

The algorithm used to determine the location of the output file that is created is therefore as follows:

```
<set generic output directory> +
<directory-symbol> +
<file name>
```

Where the following also applies:

```
if not administrator post holder then the following must apply
  <current post holder>=<post holder that started the script>
else
  error message
end if
```

Beware, this applies to the standard application functionality. When a URL is rephrased a user can still easily access files from other users through for example the browser. So do not rely on this mechanism to protect sensitive data.

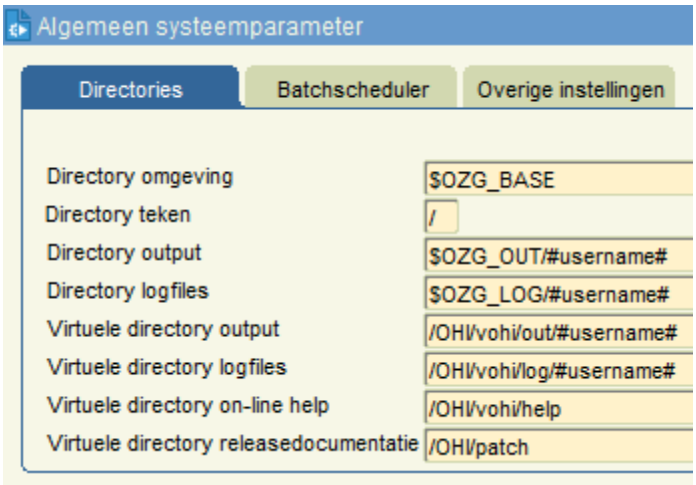
Determining the file location

The location of the directory that is searched is determined at run time.

Writing output

For writing Oracle Health Insurance output determination of the location is done using the locations specified in the "Directory output" and "Directory log files" fields in the SYS1010F screen.

Below a small screenprint that shows these fields with an example configuration (and other fields discussed later in this document).



These locations must be accessible directories on the application server. OS environment variables can also be used in the values for these locations.

The batch scheduler must be restarted before changes made in these locations take effect.

Example

```
Out-files = /u01/app/oracle/product/OHI/prod/out/#username#/#ado#/#merk#
Log-files = /u01/app/oracle/product/OHI/prod/log/#username#/#ado#/#merk#
or (using environment variables $OZG_OUT and $OZG_LOG)

Out-files = $OZG_OUT/#username#/#ado#/#merk#
Log-files = $OZG_LOG/#username#/#ado#/#merk#
```

Reading output

For reading Oracle Health Insurance output the location is determined on the basis of the locations in the "Virtual directory output" and "Virtual directory log files" fields in the SYS1010F screen.

These locations must be accessible virtual directories (within the configuration of the HTTP server) on the application server.
OS environment variables *must not* be used in the values for these locations.

Protocol, servername and portnumber are optional, so the value can start with the name of the virtual directory.

Example

Output

```
http://myhost:7777/OHI/prod/out/#username#/#ado#/#merk#
or
/OHI/prod/out/#username#/#ado#/#merk#
```

Logfiles

```
http://myhost:7777/OHI/prod/log/#username#/#ado#/#merk#
or
/OHI/prod/log/#username#
```

Conditions

1. Locations specified must not end in a directory symbol.
2. The #username# string can be used to mark where the actual username appears in the path.
   The use of substitution variables is optional.
3. Substitution variables are also available for administrative organization, brand and finance company and the source of the claims. The variable names must be in Dutch but for easy understanding the English name is mentioned between brackets:

   - #ado#. The administrative organization for which the output is being generated. The value for the administrative organization is determined from the parameters in the script request or external integration processing. If a script request or external integration processing has no administrative organization parameter then the substitution variable is replaced with empty.
   - #merk# *{#brand#}.* The brand for which output is being generated. The value for the brand is determined from the parameters in the script request or external integration processing. If a script request or external integration processing has no administrative organization parameter then the substitution variable is replaced with empty.
   - #finbedrijf# *{#finco#}.* The finance company for which output is being generated. The value for the finance company is determined from the parameters in the script request or external integration processing. If a script request or external integration processing has no administrative organization parameter then the substitution variable is replaced with empty.
   - #declaratieherkomst# *{#claimsource#}.* This variable is only replaced for claims in return media for external integration. This variable is replaced with the source of the claim for which return information is sent.

   Oracle Health Insurance determines the values for ado, brand and finance company per script request where possible and if applicable.
   The system then replaces the substitution variables with the values that have been determined. Finally, the file is written to the path that has been obtained following substitution.

   For each processing of an outgoing medium Oracle Health Insurance determines the values for ado, brand and finance company where possible and if applicable.
   The system then replaces the substitution variables with the values that have been determined.
   The batch run is created with a file name matching the path that has been obtained following substitution.

4. A location for saving the files that differs from the default location in the file system can be specified using a medium variable for outgoing or return EI medium versions. The aforementioned substitution variables can be used here also. The name of this medium variable is <MEDIUM_CODE>_<MEDIUM_VERSIE>_BESTEMMING *{MEDIUM_CODE>_<MEDIUM_VERSION>_DESTINATION}.* If the medium version with this name is not filled for a specific outgoing EI medium then the output path contained in the system parameters is used. If the medium variable does have a value then this path is used.
5. Operating system commands that are executed by the batch scheduler after a file has been saved can be specified in 'Maintain print layouts'. These can be printer commands or shell scripts that execute another process on the output files. The command defined for processing the output report is executed only for the .out file. The command defined for the processing of data files is executed once for every file produced by the script. The directory path is included in the filename.
   The substitution variables can be used in these commands also.
   If the substitution variable is empty it will be populated with the value 'null':

   ```
   scriptnaam -A #ado# -M #merk# bestandsnaam {scriptname –A #ado# -M
   #brand# filename}
   ```

   becomes the following where ado and brand have empty values:

   ```
   scriptnaam -A null -M null bestandsnaam
   {scriptname -A null -M null filename}
   ```
6. The batch scheduler retrieves the parameter values for ado, brand and finance company that have been specified by the user. The batch scheduler checks if the value for ado, brand and finance company exist. If not, this value is replaced by an empty string.
7. If there is only a parameter (and therefore a parameter value) for brand then the ado is determined based on this brand.
8. By substituting an empty parameter value, if the brand is not applicable for example, a non-valid pathname can be generated that partly comprises a number of consecutive slashes. Multiple consecutive slashes are always replaced with a single slash.
9. The OHI customer organization itself is responsible for the existence of the complete directory structure that must be present for the substitution variables that are used.
10. The #username# string cannot however be used IN such an environment variable! If there is a requirement to use the username in (the middle of) the path then two environment variables can be used for this:

    Example

    ```
    Out-files           = $OZG_OUT_PRE/#username#/$OZG_OUT_POST
    where $OZG_OUT_PRE  = /home
    and     $OZG_OUT_POST = out
    ```

Examples

Scenario 1

- Post holder JJANSEN submits script request 123 under the Health Insurance subsystem. JJANSEN does not have administrator authorization.
- Post holder member KDIJK submits script request 456 under the Financial subsystem. KDIJK does have administrator authorization.

- The output directory to be used has been set to

  ```
  $OZG_OUT/#username#
  ```

- When JJANSEN wants to view his own output the following file is opened as per the algorithm above:

  ```
  $OZG_OUT/jjansen/123.out
  ```

- When KDIJK wants to view his own output the following file is opened:

  ```
  $OZG_OUT/kdijk/456.out
  ```

- If JJANSEN wants to view KDIJK's output the following applies: JJANSEN is not an administrator post holder and JJANSEN (=current post holder) <> KDIJK (post holder who started the script); therefore, an error message is generated.
- If KDIJK wants to view JJANSEN's output the following applies: KDIJK is an administrator post holder and the following file is opened:

  ```
  $OZG_OUT/jjansen/123.out
  ```

Scenario 2

- Out files: `/home/#username#/#ado#/#merk#/#finbedrijf#`
  `{/home/#username#/#ado#/#brand#/#finco#`

  Suppose that Pietersen uses the username PPIETERS to submit a script request for medium ZRG8092E 'Genereren afrekenspec. natura/restitutie naar bestand' *{Generate payment spec. in kind/repayment to file}* with parameter value '1' for ado and parameter value 'TOP' for the brand. Following substitution the batch run will be generated using the filename `/home/ppieters/1/top`.

Scenario 3

- Out files: `/home/#ado#/#merk#/#finbedrijf#/#username#`
  `{/home/#ado#/#brand#/#finco#/#username#}`

  Suppose that Pietersen uses the username PPIETERS to submit a script request for medium ZRG8092E 'Genereren afrekenspec. natura/restitutie naar bestand' *{Generate payment spec. in kind/repayment to file}* with parameter value '2' for ado and no parameter value for the brand. Following substitution the batch run will be generated using the filename `/home/2/ppieters`.

Scenario 4

- Out files: `/home/#ado#/#merk#/#finbedrijf#/#username#`
  `{/home/#ado#/#brand#/#finco#/#username#}`

  Suppose that Pietersen uses the username PPIETERS to submit a script request for medium ZRG6055E 'Genereren polisbladen naar bestand' *{Generate policy pages to file}* with the parameter value TOP for the brand (there is no parameter value for ado). The system now determines the associated ado set '1'. Following substitution the batch run will be generated using the filename `/home/1/top/ppieters`.

Scenario 5

- Out files: `/home/#ado#/#merk#/#finbedrijf#/#username#`
  `{/home/#ado#/#brand#/#finco#/#username#}`

  Suppose that Pietersen uses the username PPIETERS to submit a script request for medium FIN2020E 'Aanmaken aanmaanbestand externe incasso' *{Create reminder file external collection}* with the parameter value 6 for the finance company. Following substitution the batch run will be generated using the filename `/home/6/ppieters`.

XML output files

Starting with the first OHI release in 2005 and onwards a set of batches has been provided that produce XML output. If necessary, an associated XSD file can also be created. This functionality is outlined below.

General functioning

A number of default parameters exist for an XML output product. These are described below:

- **XML and/or XSD file**
  This parameter is used to specify whether the script request should produce the requested XML file only (this will be the default use) or if the associated XSD file should be created in addition or on its own.
  A batch that produces an XML output product can, after all, also produce an associated XSD file if required. The file 'describes' the structure of the XML file to be produced. It is, as it were, a 'contract' which must be met by the XML output and can often be used to control and/or direct XML processing programs.
  In a script request you can specify whether you want this XSD file created. The content of the XSD file will always be the same for the output product concerned. It is only when a new version of the output product or the underlying object structure is produced that the content may differ in relation to the previous version. The same parameters are used for naming the XSD file (see below) as for the XML file, but the extension .XSD is used instead of .XML.
  The content of the XML file depends on the other functional parameters to be specified.
  In the unlikely event that you only want to create the XSD file then still valid values must be specified for the functional parameters. The script request is, for that matter, primarily intended for generating XML output.

  NOTE: When you only request XML output a more efficient (clearly faster performing) algorithm is used for writing the output to file (output is buffered to write larger pieces at once). So it is wise to only generate the XSD output when required to prevent unnecessary write delays.

- **Name of XML/XSD file**
  A file name that you set (without the associated extension) can be specified for XML output products. This is so that a useful name can be used that can easily be recognized later and so that the file is easier to locate later (of course this depends on what kind of easily recognizable name is selected).
  Additionally to this, the default/standard output (.out file) from the script request, identified by the number of the request, which contains error messages and information messages, is saved in the usual manner.
  A name will be suggested but it is recommended that you specify your own name. Should an XML file already exist with the name that is specified then the number of the script request is appended behind the name so that it is still unique. The file that already exists is not, therefore, overwritten. Any existing XSD file will be overwritten though (the reason behind this choice is that this will normally be identical or it will be simple to recreate it).

- **File location**
  A location or a 'directory' can be selected from a list of pre-defined locations

(so called 'database directories') to be set up by the database administrator which have been given a logical name (a list of the values to be used can be requested). Using this database directory functionality XML output intended for different purposes can be saved to different locations which can be determined by the organization.

- **Reference date**
  For certain, time-valid data which are retrieved when generating the XML output the situation must be determined as per a specific date. By default the date of creation of the output file (the 'system date') will be used as that date but if necessary a different date can be specified. The date is for example used to determine the marital status for an individual, which is valid on that date.

By changing the default parameter values for a script definition you can influence the standard values for the parameters, possibly making the submission of a script request more efficient.

In addition, it is important to realize that creating an XML file is a relatively intensive action: the data that is present in the database is translated into a functional model and this model is 'rich' as far as data is concerned.

The reason for this is that the XML file must contain all data that may be required so that a 'selection' can be made from this when using the XML file. At the same time, the techniques used to generate the XML output are more intensive than those that are used to generate the traditional ASCII output product. And the final reason is, of course, that the size of the XML files is considerably larger than a 'data only' file as every data component has an 'open' and 'close' tag that comprises the name of the data component. An example: <NAME>Smith</NAME>.

Still there is a way to influence the size of the XML output by including or excluding specific data elements. There is a special screen which can be used to view and alter this XML output dynamic content.

Technical management of database directories

Certain 'database directories' have to be created in the database in order to facilitate the creation of XML and any associated XSD output files. These database directories specify a 'logical directory name' that reference physical file system directories to which the XML output can be written. The files concerned are, for that matter, created from the database processes in file system directories that have been created for this purpose.

NOTE: This also means that in the case of a separated application and database server environment a file system has to be shared between the application and database servers if database created files need to be accessed also directly through the application server. The database directory or directories to be created point to such a shared file system location.

When generating the XML output products the user can select from a number of database directories that are available to the user account concerned. These database directories can be created in the database using a DBA account and can be granted to the user accounts that are permitted access to them. The account concerned must be granted write access.

If necessary a directory can be granted to PUBLIC or assigned to a role or directly to a user account. You as an OHI customer can therefore determine the privileges on directories that reference a file system directory structure completely under your own control.

By default the OZG_TMP database directory has already been granted and represented in a file system directory that you determined during the initial database side installation of the application. This database directory must remain granted.

A couple of example commands:

```
CREATE DIRECTORY PGB_UITVOER AS '/u01/xml/ozg_uitvoer/pgb';
{CREATE DIRECTORY PGB_OUTPUT AS '/u01/xml/ozg_OUTPUT/pgb'}

GRANT WRITE ON DIRECTORY PGB_UITVOER TO ksmith;
{GRANT WRITE ON DIRECTORY PGB_OUTPUT TO ksmith}

CREATE DIRECTORY NOCLAIM_UITVOER AS '/u01/xml/ozg_noclaim';
{CREATE DIRECTORY NOCLAIM_OUTPUT AS '/u01/xml/ozg_noclaim'}

GRANT WRITE ON DIRECTORY noclaim_uitvoer TO noclaim_gebruikers;
{GRANT WRITE ON DIRECTORY noclaim_output TO noclaim_users}
```

The operating system account that owns the Oracle database software must have write permission on the physical directories concerned.

Logical names for the directories must not begin with OZG_BASE.

You can only grant READ and WRITE privilege. WRITE privileges are needed to be able to write XML.

The SQL Reference Manual should be consulted for further information in relation to granting object privileges or on creating database directories.

Beware that since database release 12.2 database directories in a pluggable database can be restriced to a specific pre-defined path like for example ‘/u01/envs/test1’. Database directories need to be relative to such a prefixed path.

In such a way the output from a PDB can be better isolated from other output and it is in no way possible to create a database directory that might reference an Oracle Home folder or a folder where database files are stored.

The PATH_PREFIX variable can be used to specify this during for example the plugin operation of a pluggable database.

JSON and XML messages

The OHI Back Office application offers functionality to 'publish' individual, OHI defined, standard messages for separate intended receivers of such a message (for example an insured member or a health care provider).

This message based output is different than the common file based output, where multiple messages for multiple receivers are typically stored in one or more larger files, containing many individual messages.

Goal is to support a transition over the years from file based output to message based output.

In the current situation standard messages published by OHI Back Office are all published as JMS (Java Message Service) messages on a single OHI queue in the database, named ALG_OHI_JMS_QUEUE.

This might change over time, additional queues may be introduced. Currently it is not supported to publish such a message directly as a file.

These standard messages should clearly be distinguished from 'custom' JMS messages that your organization, using the OHI application, may publish on a separate queue for this purpose, named ALG_JMS_QUEUE. The messages on this 'custom' publish queue are not created and published by the OHI application, this queue is only offered, on request of multiple OHI customers as a standard general purpose queue for publishing custom messages.

For more information how to access the database Advanced Queuing ALG_OHI_JMS_QUEUE as JMS queue please see the relevant Appendix in **Doc[1].**

General functioning

Different type of messages will be published on the ALG_OHI_JMS_QUEUE. The set of messages that can be published on the queue can be found in the technical OHI Back Office table ALG#TYPE_QUEUE_BERICHTEN.

In a future version of the online help information the different messages will be documented including a more detailed description.

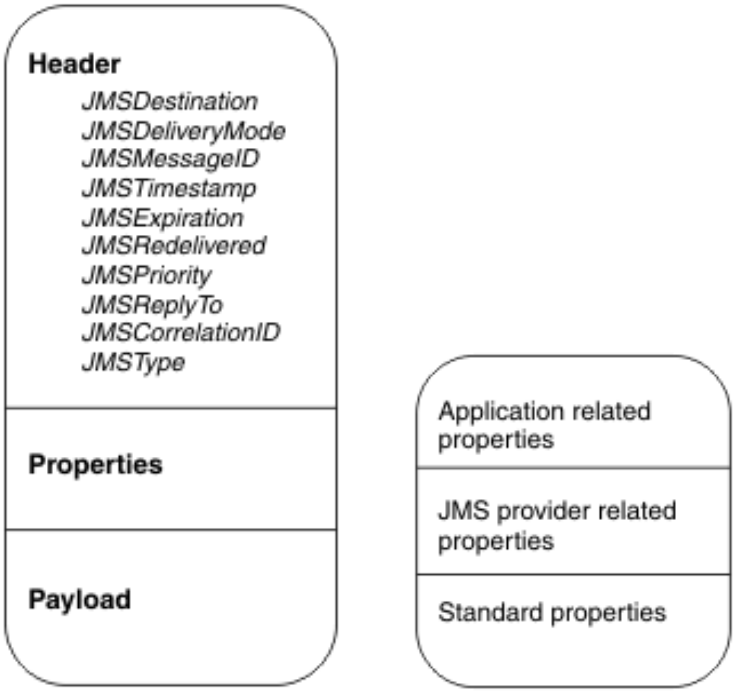
The mentioned table contains per message type a record with the following columns:

- HEADERTYPE: The value that identifies the message in the JMS Header as it appears in the JMSType property.
- SOORT_BERICHT: The OHI kind of message where two kinds are supported:
    - The value N for notification, meaning a JMS message without payload, only meant to 'signal' to the dequeuer a certaing event has occurred. Such a message will only provide some properties.
    - The value B for message ('bericht' in Dutch) in which situation a payload with the actual message, typically in XML or JSON format, will be present.

- OMS: The description of the message, a short functional description of when the message will be 'published' and for a notification which application specific properties will be present in the notification message.

For a better understanding the figure below shows the standard structure of a JMS message. A JMS message can consist of three parts, the standard JMS header, the additional properties and the payload.

The properties can be separated into three groups as is also shown. For OHI the application related properties differ per message type, so per record as present in the above mentioned table.



JMS messages are published on an Advanced Queuing database queue which supports JMS structured data.

Typically the table behind such a queue looks like:

Name	Type
Q_NAME	VARCHAR2(128)
MSGID	RAW(16)
CORRID	VARCHAR2(128)
PRIORITY	NUMBER
STATE	NUMBER
DELAY	TIMESTAMP(6)
EXPIRATION	NUMBER
TIME_MANAGER_INFO	TIMESTAMP(6)
LOCAL_ORDER_NO	NUMBER
CHAIN_NO	NUMBER
CSCN	NUMBER
DSCN	NUMBER
ENQ_TIME	TIMESTAMP(6)
ENQ_UID	VARCHAR2(128)
ENQ_TID	VARCHAR2(30)
DEQ_TIME	TIMESTAMP(6)
DEQ_UID	VARCHAR2(128)
DEQ_TID	VARCHAR2(30)
RETRY_COUNT	NUMBER
EXCEPTION_QSCHEMA	VARCHAR2(128)
EXCEPTION_QUEUE	VARCHAR2(128)
STEP_NO	NUMBER
RECIPIENT_KEY	NUMBER
DEQUEUE_MSGID	RAW(16)
SENDER_NAME	VARCHAR2(128)
SENDER_ADDRESS	VARCHAR2(1024)
SENDER_PROTOCOL	NUMBER
USER_DATA	SYS.AQ$_JMS_MESSAGE
USER_PROP	SYS.ANYDATA

Please be aware of the following aspects:

- The technical table for the ALG_OHI_JMS_QUEUE is queue table ALG#OHI_JMS_QUEUE_TAB.
- This table is the store for the queue itself but also for the associated error/exception queue.
- Messages that were processed still remain on the queue for in case there are issues which need investigation. The retention time for the queue is default configured as 7 days.
- Message processing can be monitored by querying the queue table, columns ENQ_TIME and DEQ_TIME contain the timestamp for when the message was enqueued and dequeued, always in UTC.
- The USER_DATA column, a SQL type column, contains the standard JMS header properties as separate columns within this SQL type column, the actual application properties are stored in USER_DATA.HEADER.PROPERTIES as a property list, and it contains also the payload, the actual message contents.
- The textual payload itself is stored in USER_DATA.TEXT_LOB unless it does fit within 4000 characters, in which situation it is stored in USER_DATA.TEXT_VC.
- The STATE column shows the actual queue processing state of the message.

For an overview of the relevant state values see the table below:

Value	Name	Meaning
0	READY	The message is ready to be processed, i.e., either the delay time of the message has passed or the message did not have a delay time specified
1	WAITING or WAIT	The delay specified by message_properties_t.delay while executing dbms_aq.enqueue has not been reached.
2	RETAINED OR PROCESSED	The message has been successfully processed (dequeued) but will remain in the queue until the retention_time specified for the queue while executing dbms_aqadm.create_queue has been reached.
3	EXPIRED	The message was not successfully processed (dequeued) in either 1) the time specified by message_properties_t.expiration while executing dbms_aq.enqueue or 2) the maximum number of dequeue attempts (max_retries) specified for the queue while executing dbms_aqadm.create_queue.

You can query the queue table to monitor whether expected messages are present and are being dequeued or not.

An example query for a specific message type named 'RESTITUTIEDECLARATIE' where as application and message specific property 'DECLARATIENUMMER' is fetched, next to the actual payload:

```
select t.enq_time    enqueue_tijd
,      t.deq_time    dequeue_tijd
,      t.enq_uid     enqueue_user
,      t.retry_count
,      t.priority
,     (select prp.num_value
       from   table (t.user_data.header.properties) prp
       where  prp.name = 'DECLARATIENUMMER'
      )    declaratie_nr
,     nvl(t.user_data.text_lob
         ,to_clob(t.user_data.text_vc))  bericht
,     t.state
from   alg#ohi_jms_queue_tab t
where  t.user_data.header.type = 'RESTITUTIEDECLARATIE'
order by
      t.enq_time desc
```

Message format

Messages that contain a payload (SOORT_BERICHT = 'B') are typically formatted as a JSON or XML message.

The messages are created in the same manner and by the same functional code base as which is used for the JSON responses that are created by RESTful web services. This to make sure the structure of a message is identical when returned by a web service call or when published through an OHI JMS queue.

Please note:

- OHI standard messages as published on an OHI JMS queue are created typically by 'get' operations.
- Not each RESTful web service 'get' operation may be used to publish messages on a JMS queue.
- Some messages will only be available by publishing them on the queue and the related web service operation should not be used as it will run too long and the message result will be too large. This typically applies for potentially very large messages.
- For a specific message type OHI will, when publication is activated, publish the message in XML or in JSON format. In a future release you as customer may centrally specify whether the format should be JSON or XML as both formats can be produced.
- For the standard XML messages an XSD file is delivered within the $OZG_BASE folder that contains the response message definition. The name of such a file is always HSL_<three character abbreviation>.xsd. This three character abbreviation is the same as used in the modules that are shown when the HSLBOWS deployment is chosen.
- For JSON structured messages the Swagger information as delivered for the REST webservices can be used to determine the format. The Swagger definition is published in the OHI Online Help for each web service.
- One web service definition can contain multiple operations which each can have its own response message.

For JSON structured messages the API and SDK documentation per web service can be used to retrieve a Swagger file containing all the response messages.

For the XML structured messages the link between the list of three character abbreviations and web service names as used in the Online Help may not be clear. For that reason an association table is shown below:

File name used for XSD	Resource name in Online Help
HSL_BSN	BSN
HSL_C2B	C2B
HSL_CLA	Claim
HSL_FIN	Financial
HSL_NAT	Natura Declaratie
HSL_POL	Policy
HSL_REL	Relation
HSL_ZKR	Zorgkantoor
HSL_ZPN	Zorgplan

HSL_ZZV	Machtiging

Beware, as with the Swagger files one .xsd file name may contain the message definitions for the responses of many operations present in that web service.

At this moment it is still a little hard to determine which response message definition is associated with a certain message published on the OHI JMS message queue. This will be improved in a future release of OHI and as such be documented.

Online help information

In order to access and view the online help information from within the application the "Online help virtual directory" should be set in the SYS1010F screen to the virtual directory in Oracle HTTP server (OHS) containing the online help files.

The virtual directory must point to the physical directory `$OZG_BASE/help`. It might be a double forward slash is needed before the 'help' directory.

Example

`http://myhost:7777/OHI/prod/help` (or `http://myhost:7777/OHI/prod//help`)

Release documentation

In order to access and view the release documentation within the application the "Release documentation virtual directory" should be set in the SYS1010F window to the virtual directory containing the release document files.

The virtual directory must point to the physical directory `$OZG_PATCH`.

Example

```
http://myhost:7777/OHI/patch
```