ORACLE
INSURANCE

# Oracle Insurance for Health

## Reading, Writing and Authorizing Oracle Health Insurance application files

version 1.2

Part number: E39878_01

March 2013

# CHANGE HISTORY

| Release | Version | Changes |
|---|---|---|
| 10.12.2.0.0.0 | 1.2 | • Changed examples in Reading output |
| | | |
| | | |
| | | |

# CONTENTS

# Introduction

Output is generated within the Oracle Health Insurance application using the so-called *batch scheduling mechanism.*
The files involved are stored on the server in a directory that is to be specified.

An exception to this is the XML output which is to be generated by certain batches from release 2005.01 onwards. You can specify your own file name and directory for the XML output (and XSD output if applicable) per script request. The log messages from the batch are, however, saved to the regular output file.

In addition, online help information in HTML format is used within the application.

Finally, the documentation for the delivered (patch) releases can be viewed from within the application.

This document deals with the functional aspects of these application files (layout, configuration and use).

# Requesting output from a script

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

The output for XML output-producing batches, which are being presented in increasing numbers from 2005.01, is written to a separate file. 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 Oracle*Reports modules, shell scripts, SQL*Plus modules, 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.

See the following document for the mandatory directory structure:

Oracle Health Insurance Installation, Configuration and DBA Manual for UNIX

## Naming

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

### Filename structure

Up until release 2004.01 the file name was built up from a prefix for the subsystem, a script request number and an extension:

```
<abbreviation for the subsystem> + <script number in 5 positions> + <extension>
```

***Example***

```
ZRG00123.out
```

From release 2004.02 the filename comprises the script number only, followed by an extension.

### 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), when the ASCII output type has been selected in the script definition.

*.log*

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

*.xml*

The extension for output used for statuses "Completed" and "Error" when the XML output type has been selected in the script definition.

## Authorization

**File authorization** ensures that the output can only be *read*. **System authorization** is used to ensure that 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. Exceptions to this are scripts of the *Oracle Reports* type; these run under the account used to start the Reports Server (normally `oracle`).

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 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 **viewer** functionality (e.g. MS Internet Explorer, Netscape, pg under UNIX, MS Wordviewer under Windows) 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) they can still only be *read.*

### System authorization

System authorization means that end users can only view their own output, 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> +
(<abbreviation for the subsystem> + ) ←  up to and incl. 2004.01
<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
```

## 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 made on the basis of the locations in the "Directory output" and "Directory log files" fields in the SYS1010F screen.

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

The batchscheduler must be restarted before changes 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 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:
   - #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'. This can be printer commands or shell scripts that execute another process on the output files. The command defined for processing the processing 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 client organization itself is responsible for the existence of the complete directory structure that must exist 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/ZRG00123.out
  ```

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

  ```
  $OZG_OUT/kdijk/FIN00456.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/ZRG00123.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

With effect from release 2005.01 a new form of batches is being provided which can produce XML output. If necessary, an associated XSD file can also be created. How this works 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 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 XSD file depends on the other functional parameters to be specified. In the unlikely event that you only want to create the XSD file then valid values must be specified for the functional parameters. The script request is, for that matter, primarily intended for generating XML output.

- **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 name can be used that can be recognized later and so that the file is easier to locate later (if an easily recognizable name is selected). In addition to this, the default output 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 locations 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 structure 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 will be used as that date but if necessary a

different date can be specified. The date is used to determine the marital status for an individual which is valid on that date for example.

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 output product. And the final reason is, of course, that the size of the XML files is considerably larger as every data component has an 'open' and 'close' mark that comprises the name of the data component. An example: <NAME>Smith</NAME>.

## Technical management of pointers

Certain 'directories' have to be created in the database in order to facilitate the creation of XML and any associated XSD files. These are directories to which the XML output can be written. The files concerned are, for that matter, created from the database in 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. The directory or directories to be created point here.

When generating the XML output products the user can select from a number of directories that are available to the user account concerned. These 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 can therefore arrange the rights on directories that are to be assigned yourself.

By default the OZG_TMP directory has already been granted and represented in a directory that you determined. This 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 UNIX 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 directories.

# Online help information

In order to view the online help information within the application the “Online help virtual directory” should be set in the SYS1010F screen to the virtual directory containing the online help files.

The virtual directory must point to the physical directory `$OZG_BASE/help`.

***Example***

```
http://myhost:7777/Zorg/prod/help
```

# Release documentation

In order to 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/Zorg/patch
```