ORACLE INSURANCE

Oracle Health Insurance Back Office

Service Callout Installation & Configuration Manual

Version 1.7

Part number: G49637-01

May 19, 2026

CHANGE HISTORY

Release	Version	Changes
10.25.2.0.0	1.0	• Creation
10.25.3.0.0	1.1	• Adjusted paragraph 5.2 for SOAP service consumers via APEX • Table definition for monitoring in paragraph 6.1 extended
10.25.5.0.0	1.2	• Adjusted Apex 'Deployment Instructions'
10.25.6.0.0	1.3	• Some textual adjustments
10.26.1.0.0	1.4	• Adjusted 5.1 for registration of the DB. New part number
10.26.2.0.0	1.5	• Textual adjustment in paragraph 5.1
10.26.3.0.0	1.6	• Added comments describing the additional configuration required for an OAuth2.0 Client Credential setup
10.26.4.0.0	1.7	• Added configuration and troubleshooting for callouts to "SchuldenKnoopPunt" (SKP) • Added grant on `DBMS_SQL` and support for APEX 24.2. • Added grant restricted session

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]	OHI Back Office - Installation & Configuration Manual (docs.oracle.com)
Doc[2]	OHI Back Office - Service Consumer Installation & Configuration Manual (docs.oracle.com)
Doc[3]	APEX Install and Upgrade guide
Doc[4]	Oracle 19c Database Security Guide
Doc[5]	APEX API Reference
Doc[6]	Oracle 26 AI Database Security Guide. Appendix B.1 Introduction to Oracle Database Wallets and Certificates

Contents

1 Introduction 6

1.1 Licenses 6

2 Architectural overview 7

2.1 Design 7
2.2 Alternative configurations 8

3 Prerequisites 10

3.1 APEX 10
3.2 OHI_APEX_OWNER 10
3.3 Database link 10

4 Deployment Instructions 11

4.1 Configure OHI_APEX_OWNER 11
4.2 Configure the Access Control List 11
4.3 Secure storing of credentials 13

5 Back Office configuration 16

5.1 General setting 16
5.2 Service specific settings for Service Callouts (SVLxxxxC) 17
5.3 Service specific settings for SchuldenKnoopPunt (SKP) 18
5.3.1 Background 18
5.3.2 Back Office parameters 18
5.3.3 Wallets 20
5.3.4 Database configuration for the wallets 21
5.3.5 Back Office event definitions 22

6 Monitoring and Troubleshooting 24

6.1 Monitoring 24
6.2 Troubleshooting 24
6.2.1 Testing at the OS level 25
6.2.2 Testing with HTP_UTIL commands 25
6.2.3 Testing with APEX commands 26
6.2.4 Testing with OHI routines from the APEX database 28
6.2.5 Testing with OHI routines from the OHI Back Office database 29
6.2.6 Testing with OHI SKP routines from the OHI Back Office database 30

1 Introduction

The OHI Back Office web service consumers are designed to call third-party web services or web services that have an interface that has been defined by OHI. These web services are present in the world outside OHI Back Office and are called from the PL/SQL code within the OHI Back Office database, with several intermediary steps.

Each web service consumer has a corresponding PL/SQL wrapper package providing a PL/SQL interface to the public methods of the web service consumer.

The PL/SQL wrapper functions are called from within the OHI Back Office database and, when this is configured according to the instructions in this manual, make use of Oracle Application Express (APEX) PL/SQL API packages.

This way of calling external web services is introduced in OHI Back Office release 10.25.2.0.0 and will replace the method where a JMS request and response queue are used. That method is deprecated and will be desupported in a future OHI Back Office release.

OHI Back Office release 10.26.4.0.0 introduces additional web service consumers for the "SchuldenKnoopPunt" (SKP). These do not support delivery via the JMS Queue, but only via the APEX callout described here.

1.1 Licenses

No additional OHI Back Office license option is required to use the web service consumers. The web services (incoming service calls) and web service consumers (outgoing service calls) together constitute the 'Service Layer'. The web services part of the 'Service Layer' DO require an additional OHI license.

For further information, please consult your OHI sales representative.

For the underlying technology, Oracle Database, you must acquire separate technology licenses; these are not included within OHI Back Office.
APEX is available as a free, no-cost feature of the Oracle Database. The entire stack, including Oracle APEX and Oracle REST Data Services (ORDS) are fully supported with your Oracle Database license.

2 Architectural overview

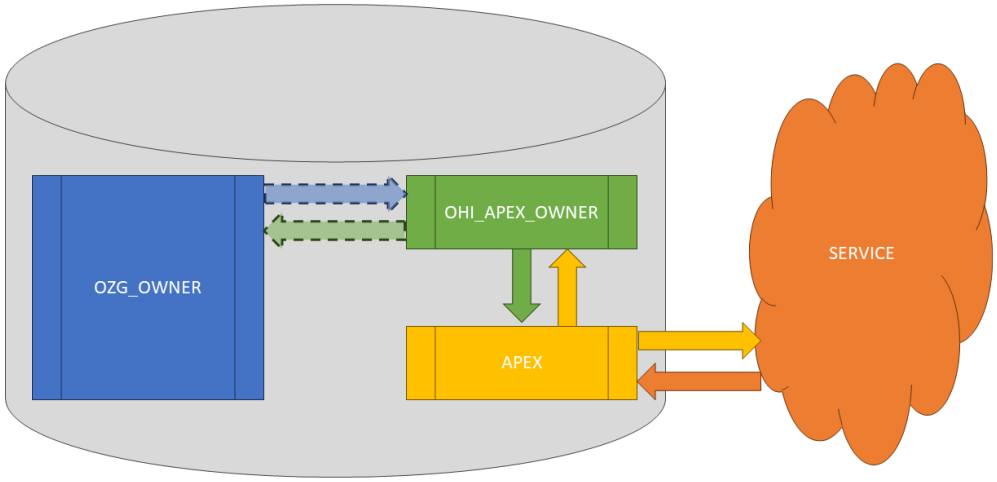
The service consumer functionality as described below offers synchronous calls to external web services. This means OHI calls to the web service are ended when the call returns a response, or a time-out is reached.

This chapter gives a high-level architectural overview of the web service consumers, in short 'SVC'. The abbreviation SVC is used in many OHI module names to make clear the files belong to the **S**er**v**ice **C**onsumer implementation, which makes use of APEX PL/SQL API packages. For the "SchuldenKnoopPunt", the abbreviation SKP is used.

The implementation of calling SOAP service consumers using the Oracle Service Bus (OSB) and JMS-queues, which will be replaced by the method described in this manual, is described in the *Oracle Health Insurance Back Office - Service Consumer Installation and Configuration Manual*. Both methods are currently supported (except for SKP) but the architecture described in this manual will replace the queue-based method. By means of configuration you can switch at a self-chosen moment.

2.1 Design

The diagram below shows how the OHI Back Office schema owner interacts with an external service operation.



The solution is based on the following architecture:

- The OHI Back Office application calls the OHI interface package with the request from within a PL/SQL routine in the OHI Back Office owner schema.
- The interface package in the OHI_APEX_OWNER schema calls the APEX API to perform the actual service callout.
- The OHI_APEX_OWNER schema resides in the same pluggable database as the APEX installation.
- The communication between the schema owner of OHI Back Office (usually OZG_OWNER) and the OHI_APEX_OWNER schema is always via a database link, even if both schemas are in the same pluggable database.

- The response status and message are propagated back to the calling function in the OHI Back Office schema.

- A timeout will occur if the response from the service callout is not returned within the given timeout period.

NB. For this configuration, the database parameter GLOBAL_NAMES must be set to false for the OHI Back Office database.

2.2 Alternative configurations

Besides the configuration where the APEX schema is in the same pluggable database, a configuration with APEX in a separate pluggable database is also an option. OHI prefers the option where the APEX schema is in the same pluggable database as the OHI Back Office application.

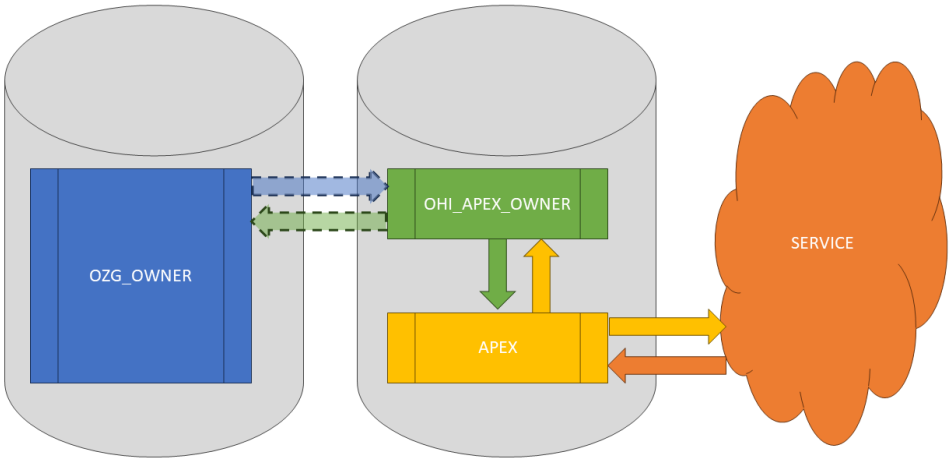
Future implementations may no longer support the configuration with APEX in a separate pluggable database.

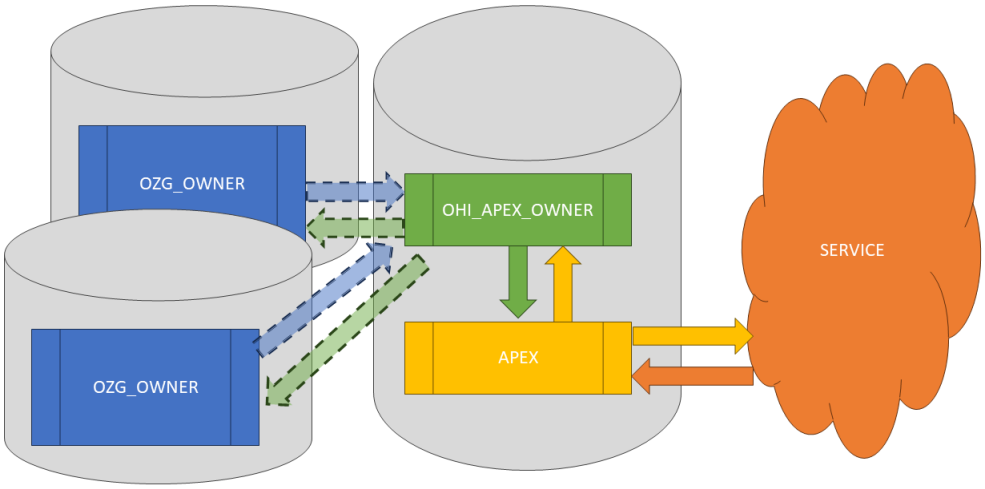
A separate pluggable database with an APEX installation may be required if you currently use APEX in the pluggable database of OHI Back Office and an upgrade is not (yet) possible.

A separate pluggable database with an APEX installation can also be used to service multiple non-production environments and so limit the amount of configuration required.

The diagrams below show the different configurations, where one or more OHI Back Office environments are connected to one APEX environment. The first one is a typical production configuration and the second one is a possible configuration for test environments.

The configuration of the endpoints (URLs) is done in OHI Back Office. This way one OHI Back Office environment can call a test endpoint and another one can call the acceptance endpoint of the same service using the same APEX installation.





3 Prerequisites

The functionality of the service consumers may not be required in all OHI BO environments. It is certainly needed in the Production environment and in some integration test environments. In environments where the service consumers are needed, this functionality must be enabled by fulfilling the prerequisites described below.

The deployment itself is described in the next chapter.

3.1 APEX

Oracle Application Express (APEX) should be installed in a pluggable database. This can be the same pluggable database (PDB) where the OHI Back Office application is installed, or a separate pluggable database, either in the same or a different Container Database (CDB).
The minimum required APEX version is 24.1. APEX version 24.2 is also certified for use with OHI Back Office. OHI may require higher versions of APEX with different OHI releases or interim patches.

More on the recommendation to install APEX in a pluggable database and alternatives can be found in chapter 5 (Utilizing Multitenant Architecture) of the Doc[3] *APEX install and Upgrade* guide.

As OHI Back Office solely uses the APEX PL/SQL APIs, only the installation of APEX described in chapter 6.3.1 of the Doc[3] *APEX Install and Upgrade* guide is required. The creation of the instance administration account, public user and router account can be omitted.

Currently the installation of the Oracle REST Data Services (ORDS) is not required.

For more information, please use the database documentation and consult the Doc[3] *APEX Install and Upgrade* guide.

3.2 OHI_APEX_OWNER

In the same pluggable database where APEX is installed, a database account OHI_APEX_OWNER must be created/present. This account will handle the communication between the OHI Back Office and the APEX schemas.
The required grants and settings for this database account are described in the next chapter.

The required software for this database account to handle the communication will be installed via OHIPATCH.

3.3 Database link

Create a private database link from the OHI Back Office schema owner to the OHI_APEX_OWNER. Register the name of this database link in the OHI Back Office parameter value for parameter 'Database link' in the Back Office parameter group 'Service Callouts'. This parameter will be used by OHIPATCH to determine the correct database link and perform the installation of the applicable sources in the OHI_APEX_OWNER schema. OHIPATCH does not require direct access to the OHI_APEX_OWNER schema.

4 Deployment Instructions

This chapter explains:

- how to configure OHI_APEX_OWNER
- how to configure the Access Control List (ACL)
- how to store credentials

4.1 Configure OHI_APEX_OWNER

In the same pluggable database where APEX is installed, a dedicated database account OHI_APEX_OWNER is required for use with OHI Back Office.

The database account OHI_APEX_OWNER needs the following system privileges. Grant these via a user with DBA privileges.

```
grant create session to ohi_apex_owner;
grant alter session to ohi_apex_owner;
grant select any dictionary to ohi_apex_owner;
grant create procedure to ohi_apex_owner;
grant execute on DBMS_NETWORK_ACL_ADMIN to ohi_apex_owner;
grant execute on DBMS_SQL to ohi_apex_owner;
grant restricted session to ohi_apex_owner;
```

Some configuration in the APEX schema is also required for OHI_APEX_OWNER to be able to perform the callouts via the APEX APIs.

```
begin
  apex_instance_admin.add_workspace
  ( p_workspace      => 'OHIBO_WORKSPACE'
  , p_primary_schema => 'OHI_APEX_OWNER'
  );

  apex_instance_admin.set_workspace_parameter
  ( p_workspace => 'OHIBO_WORKSPACE'
  , p_parameter => 'MAX_WEBSERVICE_REQUESTS'
  , p_value     => 10000000
  );
end;
/
```

These statements are also available in the script OZGI008S.sql. That script is part of the 10.25.2.0.0 release and will be installed in the /sql directory of the OHI Back Office application.

4.2 Configure the Access Control List

To be able to call a service from inside the database, a schema or user must be granted access to the service location. This is done via the access control list (ACL).

The configuration of the access control list is done via the database package DBMS_NETWORK_ACL_ADMIN.

See the documentation for this package in the *PL/SQL Packages and Types Reference* for further information.

The DBMS_NETWORK_ACL_ADMIN package is available via the OHI_APEX_OWNER account.
The 'principal' for the Access Control Entry (ACE) should be the owner account of the current APEX schema, e.g., APEX_240100 for APEX version 24.1.

The statement to allow a http(s) connection to, for instance, host myhost.company.com on port 6512 can look like this:

```
declare
  l_principal varchar2(20) := 'APEX_240100';
begin
  dbms_network_acl_admin.append_host_ace
  ( host       => 'myhost.company.com'
  , lower_port => 6512
  , upper_port => 6512
  , ace        => xs$ace_type
                  ( privilege_list => xs$name_list
                                      ( 'http'
                                      , 'connect'
                                      )
                  , principal_name => l_principal
                  , principal_type => xs_acl.ptype_db
                  )
  );
end;
/
```

If you need to communicate to the external endpoint via a proxy server, you need to repeat the statement above for the proxy server:

```
declare
  l_principal varchar2(20) := 'APEX_240100';
begin
  dbms_network_acl_admin.append_host_ace
  ( host       => 'myproxy.mycompany.com'
  , lower_port => 80
  , upper_port => 443
  , ace        => xs$ace_type
                  ( privilege_list => xs$name_list
                                      ( 'http'
                                      , 'connect'
                                      )
                  , principal_name => l_principal
                  , principal_type => xs_acl.ptype_db
                  )
  );
end;
/
```

When connecting to a TLS/SSL/https secured endpoint, the certificate of the endpoint needs to be validated. The certificates needed for the validation (e.g., the CA certificate chain or the CA root certificate) should be stored in a wallet, accessible by the database instance.
This can also be configured using the DBMS_NETWORK_ACL_ADMIN package:

```
declare
  l_principal varchar2(20) := 'APEX_240100';
begin
  dbms_network_acl_admin.append_wallet_ace
  ( wallet_path => 'file:/ohi/keyBase/wallets/httpscerts'
  , ace         => xs$ace_type
                   ( privilege_list => xs$name_list
                      ( 'use_client_certificates'
                      , 'use_passwords'
                      )
                   , principal_name => l_principal
                   , principal_type => xs_acl.ptype_db
                   )
  );
end;
/
```

How to create such a wallet (httpscerts in this example) with orapki is described in **Doc[4]** *Oracle 19c Database Security Guide* Appendix F "Managing Public Key Infrastructure (PKI) Elements".

The setup can be checked via the following database views:
select * from DBA_HOST_ACES where principal = [APEX OWNER];

select * from DBA_WALLET_ACES where principal = [APEX OWNER];

☞ When a setup with OAuth2.0 is used, additional configuration is required. The host used to retrieve the access token - the "Token url" configured as part of the Service specific settings - needs to also be configured via `dbms_network_acl_admin.append_host_ace`, as shown above. Secondly, if this host is secured via SSL using a different root certificate, the wallet configured in the `dbms_network_acl_admin.append_wallet_ace` statement shown above must also include this other root certificate.

4.3 Secure storing of credentials

The endpoint of the service operation can require different forms of authentication:

- Basic Authentication
- OAuth2.0 Client Credentials
- A client certificate issued by the endpoint.

☞ If you use an intermediate component between APEX and the external endpoint (such as a Service Bus), this intermediate component is the endpoint for APEX. Proxy server are an exception. Proxy servers can be configured as part of the endpoint configuration in Back Office parameters.

☞ Certificates can be used here for different purposes:

- Transport Layer Security (TLS) where the endpoint server is authenticated using its public certificate
- Client authentication, where the client (APEX) is authenticated by a private certificate that was issued by the endpoint server

For Basic Authentication and OAuth2.0 Client Credentials, the credential for a specific endpoint can be stored in a credential store in the APEX database. The static_id of the credential store can be used by the application to pass these credentials with the callout to the service operation, hiding the actual credentials from the OHI Back Office application.

An example to configure a basic authentication credential:

```
begin
  apex_util.set_workspace(p_workspace =>'OHIBO_WORKSPACE');
  apex_credential.create_credential
  ( p_credential_name      => 'OHIBO_CREDENTIALS'
  , p_credential_static_id => 'OHIBO_CREDENTIALS'
  , p_authentication_type  => apex_credential.C_TYPE_BASIC
  , p_allowed_urls         => apex_t_varchar2
                              ('https://myhost.company.com'
                               ,'http://myhost.company.com'
```

```
                                    )
      , p_prompt_on_install    => true
      , p_credential_comment   => 'OHIBO basic store'
      );

      apex_credential.set_persistent_credentials
      ( p_credential_static_id => 'OHIBO_CREDENTIALS'
      , p_username             => 'my_user'
      , p_password             => 'mySecretPassword123'
      );
    end;
    /
```

An example to configure a OAuth2.0 Client Credential:

```
begin
  apex_util.set_workspace(p_workspace =>'OHIBO_WORKSPACE');
  apex_credential.create_credential
  ( p_credential_name       => 'OHIBO OAuth2 Client Credential'
  , p_credential_static_id  => 'OHIBO_CLIENT_CREDENTIAL'
  , p_authentication_type   => APEX_CREDENTIAL.C_TYPE_OAUTH_CLIENT_CRED
  , p_credential_comment    => 'Client credential used for ...'
  );
  apex_credential.set_persistent_credentials
  ( p_credential_static_id => 'OHIBO_CLIENT_CREDENTIAL'
  , p_client_id            => 'abcdefghijk'
  , p_client_secret        => 'idcscs-1234567-a123-1234-ab12-a12345678'
  );
end;
/
```

Note the use of the correct workspace (OHIBO_WORKSPACE) that was created above.

Client certificates can be stored in a wallet on the operating system. This can be the same wallet that is used for the TLS certificate (`'httpscerts'` in the example above).

For using the “SchuldenKnoopPunt” with different identities/roles, some limitations apply. See paragraph 5.3 Service specific settings for SchuldenKnoopPunt (SKP)

Client certificates have a limited validity. Be sure to acquire a new certificate before the current one expires and replace the current one with the new one. Client certificates usually have an overlap in validity periods.

Access to a wallet containing https/SSL/TLS certificates can be stored in a credential store.
This credential store will not be passed in each call from OHI Back Office but is set at system level in the APEX schema. The access to this wallet is described in the previous paragraph.

An example of configuring the use for such a wallet credential store for a wallet with the name httpscerts can be as follows:

```
begin
   apex_instance_admin.set_parameter
  ( p_parameter => 'WALLET_PATH'
  , p_value     => 'file:/ohi/keyBase/wallets/httpscerts'
  );

  apex_instance_admin.set_parameter
  ( p_parameter => 'WALLET_PWD'
```

```
  , p_value     => 'mySecretPassword123'
  );
end;
/
```

See the **Doc[5]** *Oracle APEX API Reference* for more information about configuring and maintaining the credential store(s).

If the wallet (with only the public CA root certificate) is created with the **-auto_login_only** option, no wallet password is required and thus can be omitted

```
apex_instance_admin.set_parameter
  ( p_parameter => 'WALLET_PWD'
  , p_value     => null
  );
```

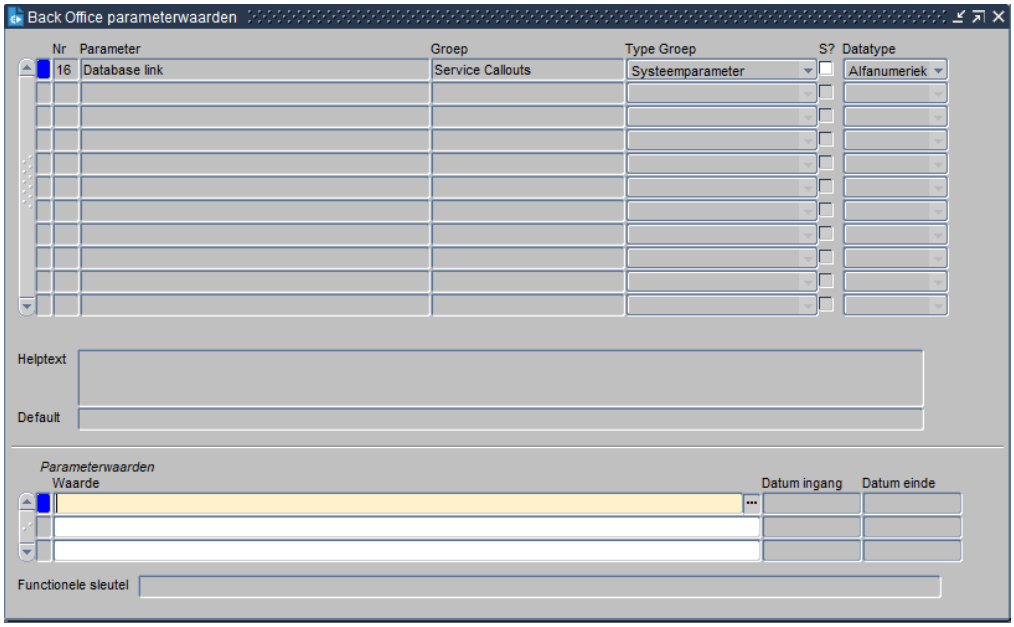
The wallet that has been registered with `apex_instance_admin` can be overruled when making the actual requests. You would typically use a wallet registered with `apex_instance_admin` to secure the communication with a service bus.

5 Back Office configuration

Part of the configuration is done in the OHI Back Office application.

5.1 General setting

The name of the database link from the OHI Back Office schema to the OHI_APEX_OWNER schema needs to be entered in the OHI Back Office parameter "Database link" in the system parameter group "Service Callouts".



The registered database link name will be used in OHIPATCH in step 110 to install and configure the interface between the OHI Back Office schema and the OHI_APEX_OWNER schema.

If it is not possible to switch from database link during a release installation, the installation and/or configuration can also be executed via sqlplus under the OHI Back Office schema account or via a DBA-role.

```
begin
  sys_install_svc_pck.run;
end;
/
```

☞ `sys_install_svc_pck.run` will only install a new version of the interface package in the OHI_APEX_OWNER schema if the version to be installed is higher than the version already present. In principle, the interface package will be downwards compatible, so OHI Back Office environments running with a lower (interim) OHI release can use the new, higher version of the interface package installed in the (central) OHI_APEX_OWNER schema as part of a higher (interim) OHI release for another OHI BO environment.

Configure the OHI database to send requests through the APEX API by registering the database identifier. This ensures controlled service callouts and prevents unintended communication from cloned environments.

This is done via the following procedure. Execute as OHI schema owner or DBA:

```
begin
  sys_install_svc_pck.registreer_dbid;
```

```
end;
/
```

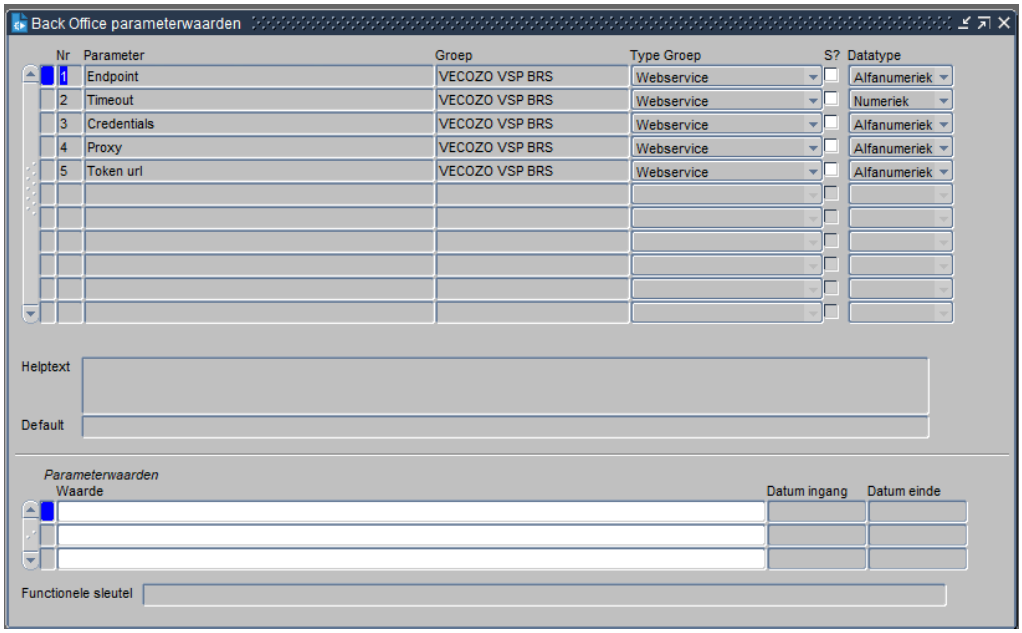
When a database is cloned, it receives a different database identifier.
Registration ensures a clone does not accidentally call production endpoints.
Also, requiring explicit registration prompts administrators to review and adjust endpoints and related service callout settings before enabling outbound communication.
Run this procedure in each environment after provisioning or cloning.

To undo/reset the registration and disable the callout possibility the following procedure is available:

```
begin
  sys_install_svc_pck.reset_dbid;
end;
/
```

5.2 Service specific settings for Service Callouts (SVLxxxxC)

The configuration for the SVC services can be set in the Back Office parameter values screen (SYS1145F) of the application instance. All parameters can be found in an own group for each individual web service that can be consumed, e.g., "VECOZO VSP BRS" for the VECOZO VPS service in REST:



Each service has five parameters:

- Endpoint: The URL endpoint of the web service to be consumed.
- Timeout: The amount of time in seconds to wait for a response.
- Credentials: The static_id of the credential to be used (see paragraph 4.2).
- Proxy: The proxy server to use for the request
- Token url: For token-based authentication flows (like OAuth2.0): The URL where to get the access token from.

The endpoint should include the path parameters like the parameter contractId in the next sample: "/polis/v1/collectievecontracten/{contractId}/polissen".

Query parameters should be omitted from the endpoint. During the actual call the path parameters will be replaced with the correct value and the query parameters will be added to the URL, with the correct values if applicable.

Depending on your network configuration, you may need to communicate to the outside world via a proxy server. In some configurations, you can communicate with this proxy server using HTTP, and you do not need a local database wallet, because the proxy server communicates with the final endpoint using its own HTPS/TLS configuration.

The value set for the timeout parameter is a maximum. Setting a high value could cause a screen or batch session to seem to "hang" waiting for a maximum of the specified number of seconds.

OHI Back Office webservice parameter groups starting with 'SVL', then 4 digits followed by a 'C' are the parameter groups for the SOAP service consumers. If at least a value for the 'Endpoint' parameter is provided the callout will be performed via the APEX callout instead of via the service consumer queues as described in the 'Oracle Health Insurance Back Office - Service Consumer Installation and Configuration Manual'.

5.3 Service specific settings for SchuldenKnoopPunt (SKP)

5.3.1 Background

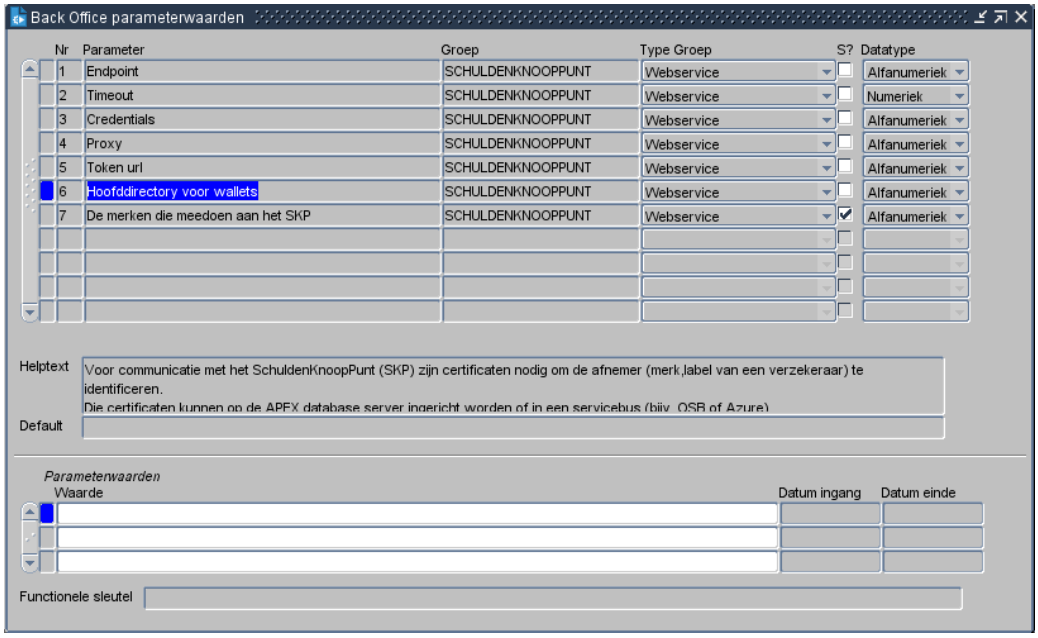
Your company may participate in the SKP with multiple identities (brands). Each identity/brand needs to communicate with its own client certificate that was received from the SKP. That means each brand needs its own wallet to identify itself with the SKP in each request.

Your company may or may not allow direct communication with the SKP from the APEX database server. Especially for production systems, you will probably have to communicate via an intermediary service bus (Oracle Service Bus, Azure Service Bus, MuleSoft, etc.). In that case the service bus is responsible for adding the correct certificate to each request. Because the message definition from the SKP does not include an identification of the brand/identity that sends a message, the service bus cannot determine the correct certificate from the message. To overcome this, OHI adds a custom HTTP Header to each request: x-***ohilabel*** that holds the brand code from OHI. The service bus needs to implement logic to translate this brand code to the correct certificate and attach that to the request before sending it to the SKP.

If you do allow direct communication (optionally via a proxy server) to the SKP (e.g., for acceptance testing), you need to configure the wallets to be used on the APEX database server. This could also apply for communication with an internal mockup service.

5.3.2 Back Office parameters

The configuration for the SchuldenKnoopPunt can be set in the Back Office parameter values screen (SYS1145F) of the application instance. All parameters can be found in the group "SCHULDENKNOOPPUNT". The parameters are slightly different than those for the SVLxxxC callouts, to implement the wallet configuration.



- Endpoint: The URL endpoint of the web service to be consumed.
    - If you communicate directly with the SKP, the Endpoint will be the URL of the SKP environment, e.g.

      https://api.preproductie-schuldenknooppunt.nl
    - If you communicate via a service bus, the Endpoint will be the URL of that servicebus, eg.

      https://{your-namespace}.servicebus.windows.net

      Your service bus will have to accept all the SKP operations used by OHI:

Tag	Path
MessageBoxV3	/system/v3/messagebox/*
OrganizationV3	/system/v3/organization/*
SbEnSkV2	/flow/sb-en-sk/v2/se/*
SbEnSkV3	/flow/sb-en-sk/v3/se/*

      These can be found in the SKP swagger.
- Timeout: The amount of time in seconds to wait for a response.

  The value set for the timeout parameter is a maximum wait time. Setting a high value could cause a screen or batch session to seem to "hang" waiting for a maximum of the specified number of seconds.
- Credentials: The static_id of the credential to be used (see paragraph 4.2).
- Proxy: The proxy server to use for the request.

  Depending on your network configuration, you may need to communicate to the outside world via a proxy server.

- Token url: For token-based authentication flows (like OAuth2.0): The URL where to get the access token from.

- Main directory for wallets: the directory on the APEX database server that holds the different wallets for different SKP identities.

  If this parameter has a value, OHI will try to find a brand-specific wallet for each request, by constructing a wallet subdirectory name as <main directory>/'skp_'||lower(<brand name>).

- The brands that participate in the SchuldenKnoopPunt. This parameter accepts multiple values. They are time valid.

Some examples for the main directory for wallets in combination with the brands and the endpoint:

- Suppose you register 2 brands as participants in SKP: DDZ1 and DDZ2 and you leave the main wallet directory empty. OHI will not add a brand-specific certificate to the request. The endpoint (you service bus or a mockup service) needs to use the HTTP header property "***x-ohilabel***" (which will have either DDZ1 or DDZ2) to determine the brand and attach the correct certificate.

- Suppose you register 2 brands as participants in SKP: DDZ1 and DDZ2 and enter "`/u01/mywallets/OHI/myenvironment`" for the main wallet directory. OHI will try to find a wallet "`/u01/mywallets/OHI/myenvironment/skp_ddz1`" or "`/u01/mywallets/OHI/myenvironment/skp_ddz2`" and add the client certificate in that wallet to the request. The HTTP header property "***x-ohilabel***" will be filled (with either DDZ1 or DDZ2) in this case too but will be ignored by the endpoint SKP.

5.3.3 Wallets

In contrast with the public certificates used for the SVLxxxxC Service callouts to a TLS/SSL/https secured endpoint, the SKP requires both public certificates and a private certificate (issued by the SKP) for each identity/brand that takes part in the SKP.

If you have multiple brands participating in the SKP, a different client certificate needs to be added for each brand.

As explained above, you can either communicate directly from APEX to the SKP or use a service bus in-between.

This paragraph assumes direct communication, optionally via a proxy server.

Even if you have only one participant/brand, you need to follow the setup described here.

Be sure to read [Doc6] Introduction to Oracle Database Wallets and Certificates about the different kinds of wallet. Auto-login wallets are a security risk, but password-protected wallets are not practical. Consider using local single sign-on (LSSO) auto-login wallets (if possible). In any case, make sure you restrict access to the wallets to the OS account that runs the database software (usually "`oracle`").

These are the steps for creating a brand-specific wallet.

1. Acquire the root Certificate Authority of the public certificate of the SKP:

In a browser, go to
https://schuldenknooppunt.atlassian.net/wiki/spaces/skp/pages/109477891/Publieke+delen+certificaten+Schuldenknooppunt

and follow the link to DigiCert Global Root G2: DigiCert Trusted Root Authority Certificates or go directly to

https://www.digicert.com/kb/digicert-root-certificates.htm#otherroots

Download the DigiCert Global Root G2 PEM file and copy it to your APEX database server, to e.g., `/tmp/DigiCertGlobalRootG2.crt.pem`

2. Acquire the client certificate for the participant/brand from the contact person who requested it from the SKP. This is a file in pkcs12f format, called something like `SE_DDZ1.pfx` and copy it to your APEX database server, to e.g. `/tmp/SE_DDZ1.pfx`.

3. On the APEX database server with the account `oracle`, create the main wallet directory. Following the example above:

```
mkdir /u01/mywallets/OHI/myenvironment
```

4. Create a wallet for the first brand:

```
cd /u01/mywallets/OHI/myenvironment

orapki wallet create -wallet skp_ddz1 -auto_login_only
```

5. Import the public Root CA certificate for the HTTPS security:

```
orapki wallet add -wallet skp_ddz1 -cert
/tmp/DigiCertGlobalRootG2.crt.pem -trusted_cert -
auto_login_only
```

6. Import the brand-specific client certificate:

```
orapki wallet import_pkcs12 -wallet skp_ddz1 -pkcs12file
/tmp/SE_DDZ1.pfx -auto_login_only
```

7. Check the contents of the wallet:

```
orapki wallet display -wallet skp_ddz1
```

8. Limit access to the wallet to account `oracle`.

For each additional brand, repeat steps 2,4,5,6,7,8.

5.3.4 Database configuration for the wallets

Connect as OHI_APEX_OWNER in the APEX database:

- Register the SKP endpoint and optionally the proxy server using

  `dbms_network_acl_admin.append_host_ace`

  as described in paragraph 4.2 Configure the Access Control List.

- For each brand, register the wallet as described in paragraph 4.2 Configure the Access Control List. For the example above, execute the following command:

```
declare
  l_principal varchar2(20) := 'APEX_240100';
```

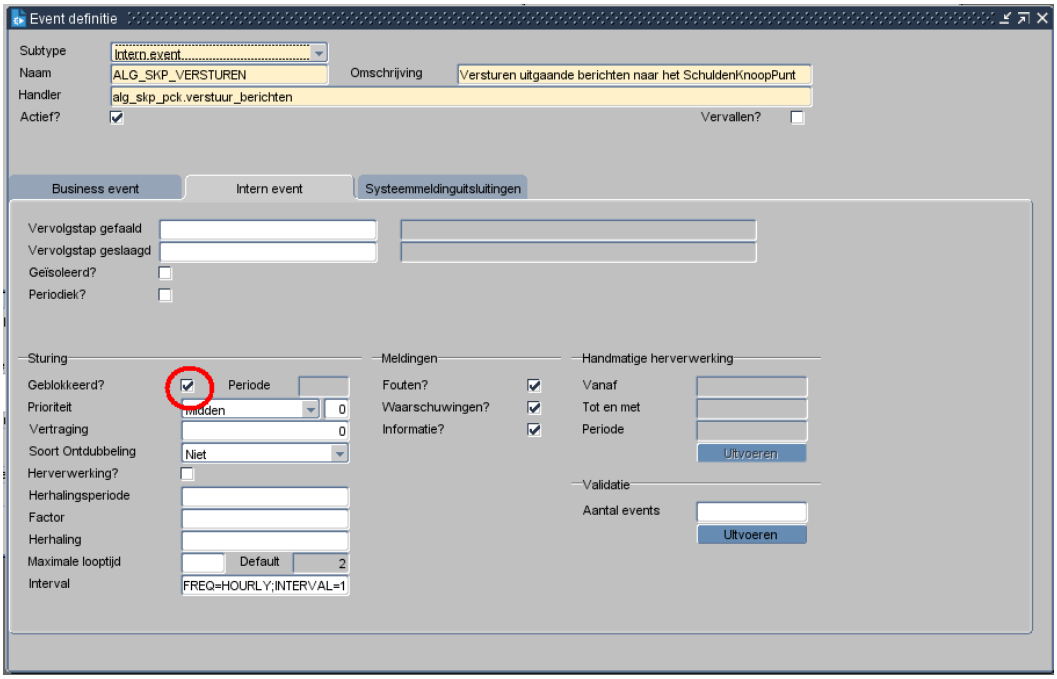
```
begin
  dbms_network_acl_admin.append_wallet_ace
  ( wallet_path => 'file: /u01/mywallets/OHI/myenvironment/skp_ddz1'
  , ace         => xs$ace_type
                   ( privilege_list => xs$name_list
                      ( 'use_client_certificates'
                      , 'use_passwords'
                      )
                   , principal_name => l_principal
                   , principal_type => xs_acl.ptype_db
                   )
  );
end;
/
```

5.3.5 Back Office event definitions

All communication with the SKP is initiated from within OHI Back Office. Even with all the configuration in place, no SKP communication will take place until it is triggered by OHI Back Office periodic events. Initially, these will be blocked, to avoid large numbers of error messages until the configuration has been completed.

Once everything else is ready, you can adjust and unblock these event definitions using the screen "Event definitie" (SYS1149F).

Search for Subtype "Intern event" and Naam "ALG_SKP%".



This will return three event definitions:

Name	Function
ALG_SKP_MESSAGEBOX	Retrieves all waiting messages from the SKP endpoint, in batches
ALG_SKP_VERSTUREN	Sends all waiting outgoing messages to the SKP endpoint, one by one.
ALG_SKP_OPHALEN_ORGANISATIES	Retrieves all relevant organisations participation in the SKP.

For all three,

- go to the tab “Intern event” and check if the default Priority and Interval are OK. If not, change them.
- uncheck the field “Geblokkeerd?”
- press the Save button.

6 Monitoring and Troubleshooting

This chapter contains some troubleshooting paragraphs. These can be used if the service consumer calls do not work and result in errors that are not functional by nature.

6.1 Monitoring

All calls are logged in the table ALG#SERVICE_CONSUMER_LOG

Column	Description
ID	Unique identification for each row
SERVICENAAM	The functional (OHI) name of the service
BERICHT_TYPE	Functional message code; for the VECOZO service this is the actual attribute 'bericht type'
BERICHT_SUBTYPE	For the VECOZO service this is the actual attribute 'bericht subtype'
BERICHT_ACTIE	For the VECOZO service this is the actual attribute 'bericht actie'
STARTTIJD_AANVRAAG	Time the call to the endpoint was made
EINDTIJD_AANVRAAG	Time a response (or timeout) was received from the endpoint or (APEX)API
TYPE_OPERATIE	Type of operation, e.g., GET or POST
RESPONSE_STATUS	The response (http) status of the request
LENGTE_AANVRAAG	Size in bytes of the request
LENGTE_ANTWOORD	Size in bytes of the response
AANVRAAG	The endpoint (URL) used in the callout including the resource and query parameters
GEBRUIKER	User initiating the callout
AANVRAAG_TEKST	Request payload in ASCII format if applicable
AANVRAAG_BINAIR	Request payload in binary format if applicable
ANTWOORD_TEKST	Response payload in ASCII format
ERROR_TEKST	The error message, if an error occurred during the callout

The same information is available in the batch report "Overzicht autonome verwerking" (SYS5002R) and in the dashboard of the Autonomous Processing Framework (AVF) in the OHIJET application. Look for the Service Consumers sections.

The retention for the data in this table defaults to 7 days, but this can be altered via the OHI Back Office parameter 'Retentie log' in the parameter group 'Service Callouts'.

The functional programs that use the APEX callouts will log additional messages in table ALG#MELDINGEN, as do the other program units in OHI BO.

6.2 Troubleshooting

Setting up the connections to endpoints can be tricky, especially because of the different certificates.

Testing your connection can be done at different levels:

- At the OS level, on the server where the APEX database runs. This tests the network connectivity and the certificates.

- Testing with HTP_UTIL commands, from the APEX Owner schema in the database. This adds tests of the privileges (Host ACEs and Wallet ACEs) and the wallet setup.

- Testing with APEX commands from the APEX Owner schema in the database. This adds tests of the APEX wrappers. The same tests can be repeated from the OHI_APEX_OWNER schema.

- Testing with OHI routines from the OHI Back Office database. This also tests the database link and validates the whole communication chain.

The remainder of this paragraph will give examples for a direct connection to the SchuldenKnoopPunt (SKP) using wallets, as this is the most complex case because of the required client certificate. If you communicate via a service bus, the communication with that service bus may require OAuth, and the certificates may have to be added to the request by the service bus. In that case, the instructions below will not apply, but some parts may still be useful.

6.2.1 Testing at the OS level

Testing at the OS level requires:

- access to the APEX database server as "oracle"

- utility "curl"

- the client certificate file for the participant/brand for an environment at SKP. This is a file in pkcs12f format that is password protected. Let's assume this is the file /tmp/SE_DDZ1.pfx

- network access to the SKP environment, possibly via a proxy server.

Do the following to test connectivity with the SKP pre-production server:

1. Convert `/tmp/SE_DDZ1.pfx to a` *.pem file:

   ```
   openssl pkcs12 -in /tmp/SE_DDZ1.pfx -out /tmp/SE_DDZ1.pem -nodes
   ```

   This will ask for the password that was given by SKP.

2. Set the proxy server to the correct values for your company (optional):

   ```
   export HTTP_PROXY=http://myproxy.mycompany.com:80

   export HTTPS_PROXY=http://myproxy.mycompany.com:80
   ```

3. Send a request with curl to get the organization that is linked to the wallet from the SKP:

   ```
   curl -v
   --header "Content-Type: application/json" \
   --request GET
   --cert /tmp/SE_DDZ1.pem \
   https://api.preproductie-schuldenknooppunt.nl:443/system/v3/organization/
   ```

This should return an organization in JSON format. If it does not, the output should give you a clue to fix your setup.

6.2.2 Testing with HTP_UTIL commands

Testing with with HTP_UTIL commands requires:

- Access to the APEX owner schema in the database

- A wallet set up on the database server OS according to the instructions in paragraph 5.3.3 Wallets
- Database configuration for the wallet according to the instructions in paragraph 5.3.4 Database configuration for the wallets.
- Database configuration for the SKP host and the proxy server in the APEX database (Host ACEs)

Do the following to test connectivity with the SKP pre-production server:

1. Log on as the APEX schema owner.
2. Adapt the following code for your environment and execute. Make sure you have the same filename as in the wallet_ace.
3. Execute:

```
set serveroutput on;
declare
  l_http_request  utl_http.req;
  l_http_response utl_http.resp;
  l_text          varchar2(4000);
  l_wallet        varchar2(200) :=
      'file:///u01/mywallets/OHI/myenvironment/skp_ddz1';
begin
  -- extra debug output:
  --utl_http.set_detailed_excp_support(TRUE);
  utl_http.set_transfer_timeout(15000);
  utl_http.set_wallet(l_wallet);
  utl_http.set_proxy('http://myproxy.mycompany.com:80');

  l_http_request := utl_http.begin_request('https://api.preproductie-
schuldenknooppunt.nl/system/v3/organization','GET', 'HTTP/1.1');
  utl_http.set_header(l_http_request, 'Host', 'api.preproductie-
schuldenknooppunt.nl');
  utl_http.set_header(l_http_request, 'User-Agent', 'Mozilla/4.0');
  utl_http.set_header(l_http_request, 'content-type', 'application/json');

  l_http_response := utl_http.get_response(l_http_request);
  begin
    loop
      utl_http.read_line(l_http_response, l_text);
      dbms_output.put_line (l_text);
    end loop;
  exception
    when utl_http.end_of_body
    then
      utl_http.end_response(l_http_response);
  end;
end;
/
```

This should output organization data in JSON format for the brand/participant of the wallet.

You can also set additional tracing in the database. See My Oracle Support document Doc ID 2288646.1 “SRDC - Providing Supporting Information for UTL_SMTP, UTL_MAIL, UTL_HTTP and ACL Related Issues”:

```
ALTER SESSION SET EVENTS = '10937 TRACE NAME CONTEXT FOREVER, LEVEL 4';
ALTER SESSION SET EVENTS = '10590 TRACE NAME CONTEXT FOREVER, LEVEL 14';
```

6.2.3 Testing with APEX commands

Testing with APEX commands requires:

- Access to the APEX owner schema in the database
- Access to the OHI_APEX_OWNER schema in the database, set up according to paragraph 4.1 Configure OHI_APEX_OWNER
- A wallet set up on the database server according to the instructions in paragraph 5.3.3 Wallets
- Database configuration for the wallet according to the instructions in paragraph 5.3.4 Database configuration for the wallets.
- Database configuration for the SKP host and the proxy server in the APEX database

Do the following to test connectivity with the SKP pre-production server:

1. Log on as the APEX schema owner.
2. Adapt the following code for your environment and execute. Make sure you have the same filename as in the wallet_ace.

```
set serveroutput on;
declare
  l_response clob;
  l_obj json_object_t;
  l_clob clob;

  procedure print_clob(p_clob clob)
  is
    l_pos integer := 1;
    l_len integer := dbms_lob.getlength(p_clob);
    l_chunk varchar2(32767);
  begin
    while l_pos <= l_len
    loop
      l_chunk := dbms_lob.substr(p_clob, 32767, l_pos);
      dbms_output.put_line(l_chunk);
      l_pos := l_pos + 32767;
     end loop;
  end;

begin
  -- extra debug output:
  -- apex_debug.enable_dbms_output;
  -- utl_http.set_detailed_excp_support(true);
  apex_web_service.clear_request_headers;
  apex_web_service.g_request_headers(1).name := 'Content-type';
  apex_web_service.g_request_headers(1).value := 'application/json';
  apex_web_service.g_request_headers(2).name :=  'User-Agent';
  apex_web_service.g_request_headers(2).value := 'Mozilla/4.0';

  l_response := apex_web_service.make_rest_request
    ( p_url => 'https://api.preproductie-schuldenknooppunt.nl/system/v3/organization'
    , p_http_method => 'GET'
    , p_wallet_path => 'file:///u01/mywallets/OHI/myenvironment/skp_ddz1'
 -- , p_https_host => 'api.preproductie-schuldenknooppunt.nl'
    , p_proxy_override  => 'http://myproxy.mycompany.com:80';
    , p_transfer_timeout => 15000
    );
  -- print_clob(l_response);
  select json_serialize(l_response returning clob pretty)
  into l_clob
  from dual;
  print_clob(l_clob);
end;
/
```

This should output organization data in JSON format for the brand/participant of the wallet.

If the previous step is successful, repeat the same steps for OHI_APEX_OWNER.

6.2.4 Testing with OHI routines from the APEX database

Testing with the OHI wrapper from the APEX database requires:

- A wallet set up on the database server according to the instructions in paragraph 5.3.3 Wallets
- Database configuration for the wallet according to the instructions in paragraph 5.3.4 Database configuration for the wallets.
- Database configuration for the SKP host and the proxy server in the APEX database
- Access to the OHI_APEX_OWNER schema in the APEX database set up according to paragraph 4.1 Configure OHI_APEX_OWNER
- A database link between the OHI BO database and the APEX database (for the installation only, not for this test)
- Installation of OHI BO release 10.26.4.0.0 or higher (for SKP) including execution of `sys_install_svc_pck.run` and configuration of the Back Office parameters in the group "SCHULDENKNOOPPUNT".

Do the following to test connectivity with the SKP pre-production server:

1. Log on as the OHI_APEX_OWNER.
2. Adapt the following code for your environment. Make sure you have the same filename as in the wallet_ace.

```
declare
    l_txt_lst       ohi_apex_interface_pck.a_txt_lst_tp :=
    ohi_apex_interface_pck.a_txt_lst_tp();
    l_bin_lst       ohi_apex_interface_pck.a_bin_lst_tp :=
    ohi_apex_interface_pck.a_bin_lst_tp();
    l_response_status    number;
    l_reason_phrase      varchar2(32767);
    l_response_headers   ohi_apex_interface_pck.nvp_lst_tp;
    l_clob clob;
begin
  ohi_apex_interface_pck.rest_request
  ( pi_url=> 'https://api.preproductie-schuldenknooppunt.nl/system/v3/organization'
  , pi_http_method       => 'GET'
  , pio_payload          => l_txt_lst
  , pio_payload_bin      => l_bin_lst
  , pi_content_type      => 'application/json'
  , pi_content_type_bin  => 'application/octet-stream'
  , pi_proxy_override    => 'http://myproxy.mycompany.com:80'
  , pi_wallet_path       => 'file:///u01/mywallets/OHI/myenvironment/skp_ddz1'
  , po_response_status   => l_response_status
  , po_reason_phrase     => l_reason_phrase
  , po_response_headers  => l_response_headers
  );

  dbms_output.put_line('response_status = '||l_response_status);
  if l_response_status <> 200
  then
    dbms_output.put_line('reason = '||l_reason_phrase);
  end if;

  if l_bin_lst is not null
  and l_bin_lst.count > 0
  then
    for i in 1 .. l_bin_lst.count
    loop
```

```
      dbms_output.put_line(utl_raw.cast_to_varchar2(l_bin_lst(i)));
    end loop;
  end if;
end;
/
```

This should output “response_status = 200” and organization data in JSON format for the brand/participant of the wallet.

If you get errors, consider enabling the APEX debugging:

```
begin
APEX_DEBUG.ENABLE(p_level => 4);
end;
```

```
select * from APEX_DEBUG_MESSAGES;
```

6.2.5 Testing with OHI routines from the OHI Back Office database

Testing with OHI routines from the OHI Back Office database will also test the database link. It requires:

- A wallet set up on the database server according to the instructions in paragraph 5.3.3 Wallets
- Database configuration for the wallet according to the instructions in paragraph 5.3.4 Database configuration for the wallets.
- Database configuration for the SKP host and the proxy server in the APEX database
- Access to the OHI BO owner schema in the database (usually OZG_OWNER)
- A database link between the OHI BO database and the APEX database.
- Installation of OHI BO release 10.26.4.0.0 or higher (for SKP) including execution of `sys_install_svc_pck.run`.

Do the following to test connectivity with the SKP pre-production server:

1. Log on as the OHI Back Office schema owner (usually OZG_OWNER).
2. Adapt the following code for your environment. Make sure you have the same filename as in the wallet_ace.

```
declare
  l_txt_lst       OHI_SERVICE_CALLOUT.a_txt_lst_tp :=
    OHI_SERVICE_CALLOUT.a_txt_lst_tp();
  l_bin_lst       OHI_SERVICE_CALLOUT.a_bin_lst_tp :=
    OHI_SERVICE_CALLOUT.a_bin_lst_tp();
  l_response_status    number;
  l_reason_phrase      varchar2(32767);
  l_response_headers   OHI_SERVICE_CALLOUT.nvp_lst_tp;
  l_clob clob;
begin
  OHI_SERVICE_CALLOUT.rest_request
  ( pi_url    => 'https://api.preproductie-schuldenknooppunt.nl/system/v3/organization'
  , pi_http_method       => 'GET'
  , pio_payload          => l_txt_lst
  , pio_payload_bin      => l_bin_lst
  , pi_content_type      => 'application/json'
  , pi_content_type_bin  => 'application/octet-stream'
  , pi_proxy_override    => 'http://myproxy.mycompany.com:80'
  , pi_wallet_path       => 'file:///u01/mywallets/OHI/myenvironment/skp_ddz1'
  , po_response_status   => l_response_status
  , po_reason_phrase     => l_reason_phrase
  , po_response_headers  => l_response_headers
  );

  dbms_output.put_line('response_status = '||l_response_status);
  if l_response_status <> 200
```

```
  then
    dbms_output.put_line('reason = '||l_reason_phrase);
  end if;

  if l_bin_lst is not null
  and l_bin_lst.count > 0
  then
    for i in 1 .. l_bin_lst.count
    loop
      dbms_output.put_line(utl_raw.cast_to_varchar2(l_bin_lst(i)));
    end loop;
  end if;
 end;
/
```

This should output "response_status = 200" and organization data in JSON format for the brand/participant of the wallet, just like the previous paragraph.

In case of errors, check for (new) records in the tables

- ALG#SERVICE_CONSUMER_LOG
- ALG#MELDINGEN
- ALG#MELDING_XMLCONTEXTS

6.2.6 Testing with OHI SKP routines from the OHI Back Office database

Testing with OHI SKP routines from the OHI Back Office database will also test the settings of the the Back Office parameters in the group "SCHULDENKNOOPPUNT". This requires:

- A wallet set up on the database server according to the instructions in paragraph 5.3.3 Wallets
- Database configuration for the wallet according to the instructions in paragraph 5.3.4 Database configuration for the wallets.
- Database configuration for the SKP host and the proxy server in the APEX database
- Access to the OHI BO owner schema in the database (usually OZG_OWNER)
- A database link between the OHI BO database and the APEX database.
- Installation of OHI BO release 10.26.4.0.0 or higher (for SKP) including execution of `sys_install_svc_pck.run` and configuration of the Back Office parameters in the group "SCHULDENKNOOPPUNT".

Do the following to test connectivity with the SKP pre-production server:

1. Log on as the OHI Back Office schema owner (usually OZG_OWNER).
2. Execute the following code:

```
begin
  alg_skp_pck.get_organisaties;
end;
/

select naam, woonplaats, organisatietype
,      nvl(laatste_mutatie_moment, creatie_moment) as laatste_wijziging
from   alg#skp_organisaties
order by nvl(laatste_mutatie_moment, creatie_moment) desc;
```

The query should return multiple records, with a very recent `laatste_wijziging`.

If you repeat the call, no new updates will be retrieved, and the `laatste_wijziging` will not change, but you will see a new record in table ALG#SERVICE_CONSUMER_LOG with message_type = `organization`:

```
select bericht_type, eindtijd_aanvraag, response_status
from alg#service_consumer_log
order by id  desc;
```

In case of errors, check for (new) records in the tables

- ALG#SERVICE_CONSUMER_LOG
- ALG#MELDINGEN
- ALG#MELDING_XMLCONTEXTS