HTTP Utility User's Guide v24.0

Date: 01/10/2024

Author: Jean-Marc MALMEDY

Document version: 24.0

1. Generalities

The HTTP Utility is a PL/SQL utility designed to simplify the process of sending HTTP requests within Oracle Database. It provides a convenient interface for invoking web services and triggering GitLab pipelines through HTTP requests.

2. Description of the solution

The solution is built on top of the *UTL_HTTP* Oracle native package. Its functionalities are encapsulated in the HTTP Utility to make it easier to use.

The solution is mainly implemented in the *HTTP_UTILITY_KRN* PL/SQL package and proposes two main functionalities:

- Sending a general HTTP request from the database.
- Sending an HTTP request to a GitLab server to trigger the execution of a pipeline or pipeline job.

3. Technical implementation

3.1. HTTP_UTILITY_VAR package

This package contains the declaration of all types, variables and exceptions that are used by the packages that implements the logic.

3.2. HTTP_UTILITY_KRN package

This package is the core of the utility and implements its functionalities.

3.2.1. Procedures and functions

3.2.1.1. SET_VERBOSE procedure

This public procedure activates or deactivates the *verbose* mode. When activated, applicative and debug messages are displayed in the *DBMS Output*.

Parameters:

Name	Type	Description
p_verbose	BOOLEAN	whether the verbose mode must be activated

3.2.1.2. LOG_MSG procedure

This private procedure displays a message in the *DBMS Output* if the verbose mode is activated.

Name	Type	Description
p_message	VARCHAR2	the message to be displayed

3.2.1.3. RAISE_ERROR procedure

This private procedure displays an error message, if the verbose mode is activated, and raises an application error.

Parameters:

Name	Type	Description
p_code	SIMPLE_INTEGER	application error code
p_msg	VARCHAR2	error message

3.2.1.4. SEND_HTTP_REQUEST procedure

This public procedure sends an HTTP request.

Parameters:

Name	Type	Description
p_url	VARCHAR2	the URL the HTTP request must be sent to
p_wallet_path	VARCHAR2	optional parameter containing the path of the wallet the certificate is stored in
p_proxy	VARCHAR2	optional parameter containing the possible proxy server address
p_verbose	BOOLEAN	optional parameter indicating whether the verbose mode must be activated

Exceptions and errors:

Error	Description
-20001	invalid parameter
-20002	sending HTTP request failure

3.2.1.5. SEND_GITLAB_HTTP_REQUEST procedure

This public procedure sends an HTTP request to a GitLab server in order to trigger the execution of a pipeline or pipeline job.

Parameters:

Name	Type	Description
p_url_root	VARCHAR2	root part of the URL the HTTP request must be sent to
p_token	VARCHAR2	the token of the GitLab pipeline that needs to be triggered
p_var_name	VARCHAR2	optional, this parameter contains the name of a GitLab variable that needs to be initialized
p_var_value	VARCHAR2	optional, this parameter contains the value of a GitLab variable that needs to be initialized
p_wallet_path	VARCHAR2	optional parameter containing the path of the wallet the certificate is stored in
p_proxy	VARCHAR2	optional parameter containing the possible proxy server address
p_verbose	BOOLEAN	optional parameter indicating whether the verbose mode must be activated

Exceptions and errors:

Error	Description
-20001	invalid parameter
-20002	sending HTTP request failure

4. Using the solution

4.1. Sending an HTTP request

The send_http_request procedure sends an HTTP request to the URL passed as parameter. If a certificate is needed, the path of the wallet containing the certificate needs to be passed as parameter too. If the request needs to be sent through a proxy server, its address must be passed also. A last parameter indicates whether the verbose mode needs to be activated.

```
BEGIN
   http_utility_krn.send_http_request(
      p_url         => 'https://www.test.com/abcd'
   );
END;
/
```

In the example above, the HTTP request is sent to the *https://www.test.com/abcd* URL. No certificate or proxy server are used.

```
BEGIN
   http_utility_krn.send_http_request(
      p_url             => 'https://www.test.com/abcd'
    , p_wallet_path     => 'file://ab/cd/ef'
    , p_proxy           => 'proxy.test.com:8012'
    , p_verbose         => TRUE
   );
END;
/
```

In the example above, the HTTP request is sent to the *https://www.test.com/abcd* URL. A certificate is mandatory and is stored in the */ab/cd/ef* wallet directory. The request will be sent through the *proxy.test.com* proxy server, using the *8012* IP port. Finally, the verbose mode is activated.

4.2. Sending an HTTP request to a GitLab server

An HTTP request can also be sent to a GitLab server in order to trigger the execution of a GitLab pipeline or a pipeline job. Comparing to a classical HTTP request, additional information can be passed to the send_gitlab_http_request procedure:

- the token of the GitLab pipeline trigger
- the name of a GitLab variable that needs to be initialized
- the value of a GitLab variable that needs to be initialized

Those parameters are used and combined by the procedure to build and format the complete URL.

```
BEGIN
   http_utility_krn.send_gitlab_http_request(
      p_url             => 'https://www.test.com/abcd'
    , p_token           => 'aabbccddeeff'
   );
END;
/
```

In the example above, a simple HTTP request is sent to the GitLab server, to the *https://www.test.com/abcd* URL. The *aabbccddeeff* token is passed to the procedure. No variable will be initialized and no certificate or proxy servers are needed. The complete URL generated by the procedure is `https://www.test.com/abcd?token=aabbccddeeff` .

```
BEGIN
   http_utility_krn.send_gitlab_http_request(
      p_url             => 'https://www.test.com/abcd'
    , p_token           => 'aabbccddeeff'
    , p_var_name        => 'NEXT_STAGE'
    , p_var_value       => 'FINALIZE'
    , p_wallet_path     => 'file://ab/cd/ef'
    , p_proxy           => 'proxy.test.com:8012'
    , p_verbose         => TRUE
   );
END;
/
```

In the example above, an HTTP request is sent to the GitLab server, to the *https://www.test.com/abcd* URL. The *aabbccddeeff* token is passed to the procedure.The *NEXT_STAGE* variable is initialized with the *FINALIZE* value. A certificate is needed and it is stored in the */ab/cd/ef* wallet directory. The request will be sent through the *proxy.test.com* proxy server, using the *8012* IP port. Finally, the verbose mode is activated. The complete URL generated by the procedure is `https://www.test.com/abcd?token=aabbccddeeff&variables[NEXT_STAGE]=FINALIZE` .

5. Installation

5.1. Pre-requisites

The HTTP Utility is a low-level service designed to be integrated or used in any development or higher-level services.

Therefore, it does not depend on any other tool or utility.

5.2. Privileges and roles

Since the solution is built on top of the *UTL_HTTP* package, the schema user the utility will be installed in must be granted with the *EXECUTE* privilege on this package.

5.3. Installation procedure

The HTTP Utility is now integrated in the **DBCoE PL/SQL Toolkit** and is then deployed and upgraded using the DBM_Utility deployment tool.

No parameters must then be passed to the installer.

Once the **DBCoE PL/SQL Toolkit** is downloaded, the HTTP Utility can be installed using the following commands:

- Deploying the HTTP Utility: `@dbm-cli install http_utility`
- Migrating to the the last version of the HTTP Utility: `@dbm-cli migrate http_utility`
- Uninstalling the HTTP Utility: `@dbm-cli uninstall http_utility`