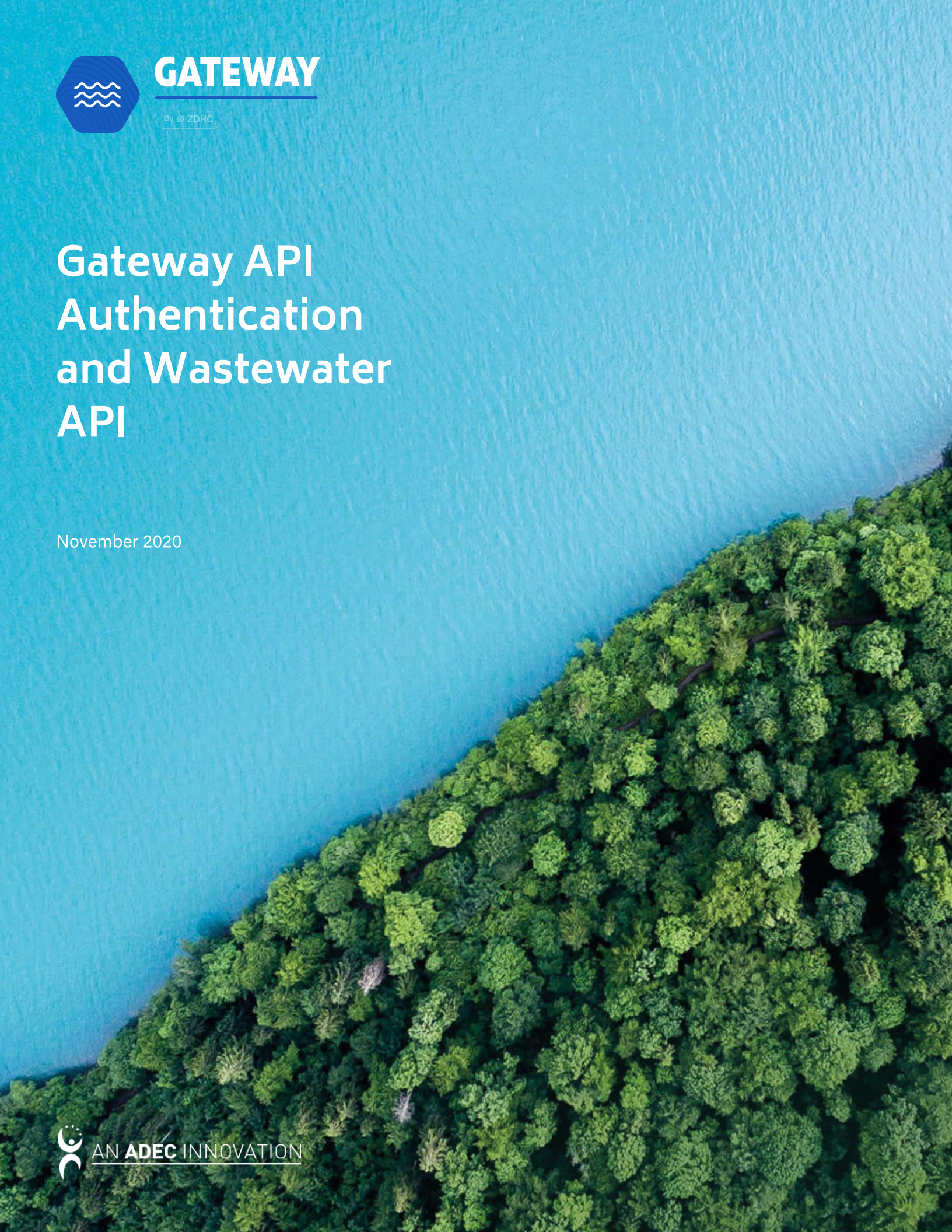
GATEWAY

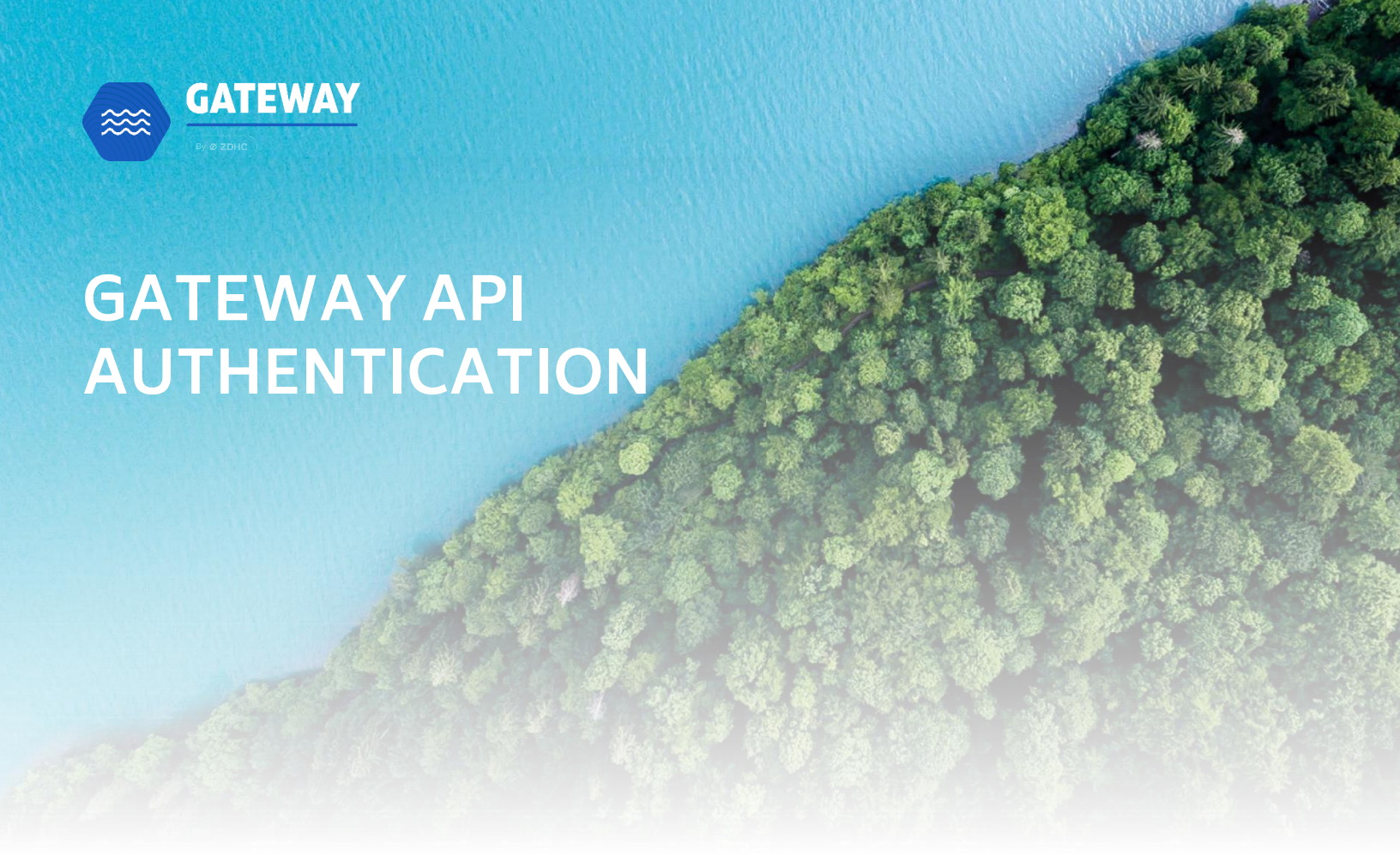
By ZDHC

Gateway API Authentication and Wastewater API

November 2020



AN ADEC INNOVATION

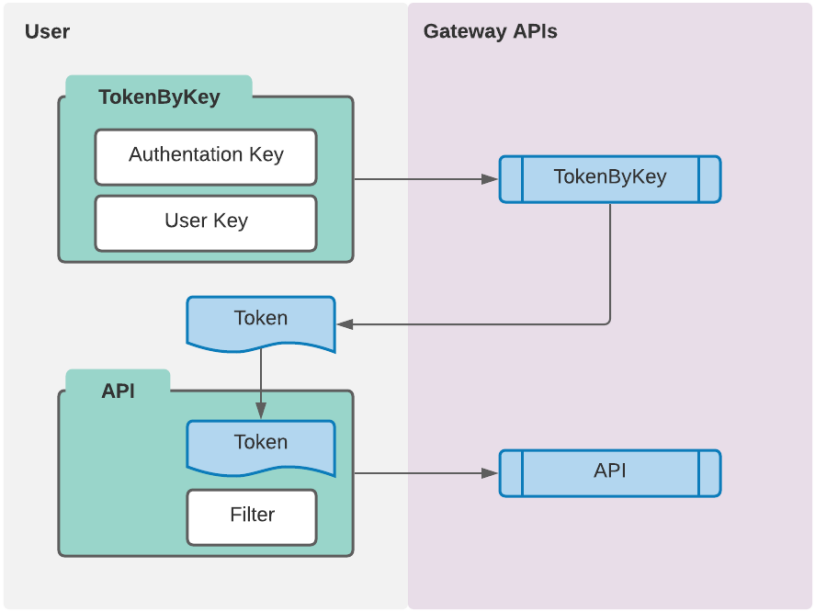


GATEWAY API AUTHENTICATION

Authentication Overview

The Gateway APIs requires two keys for Authentication. The Authentication returns a token that is used in all subsequent API calls.

- Authentication Key – The Authentication Key is provided by ZDHC to each Account or Third-Party Provider. If you do not have an Authentication Key email Gateway@zdhc.org to request an Authentication Key.
- User Key – The user key is self-generated and should be treated with the same security and privacy as your username and password.



Create an API User Key

1. Login to Gateway
2. Navigate to Users
3. Click on 'Edit'
4. Click on Create Key

API Development

The full list of APIs, the methods and available filters:

- https://www.my-aip.com/Gateway/rest/v1/
- https://www.my-aip.com/Gateway/rest/v2/

URI

The following are the core URI.

- Base URL

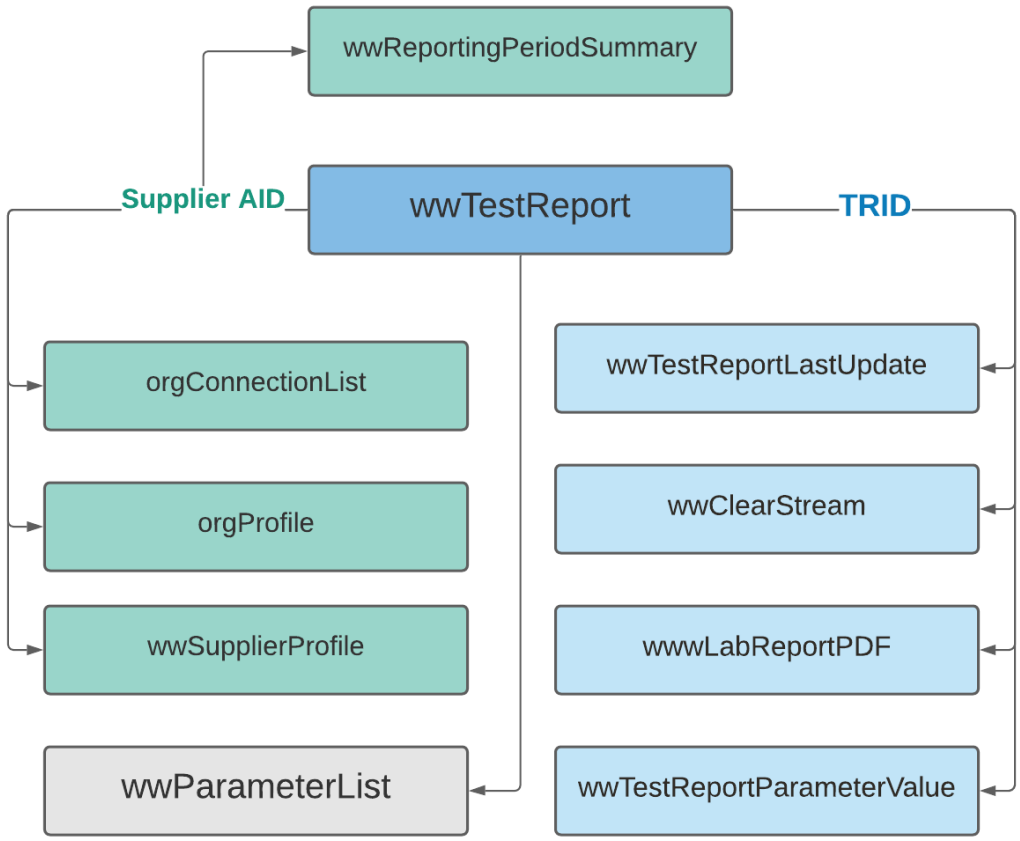
  https://www.my-aip.com/Gateway/rest/v1

- Authorization URL

  https://www.my-aip.com/Gateway/rest/oauth/tokenbykey

WasteWater APIs Overview

The API are connected through the following diagram with the items in blue linked by the Test Report ID (TRID) and the items in green through the Supplier AID.



API EndPoint Details

v2/orgProfile

Profiles basic information about the facility such the name, location and industries.

v1/wwSupplierProfile

The wwSupplierProfile is specific to supplier that submit wastewater (ww) lab test results. The information on the ww outlines the discharge profile.

v1/orgConnectionList

The orgConnectionList is used by Brands and Supplier to determine the suppliers where they do and do not have a connection. A Brand will only see wastewater data for supplier if there is an accepted connection.

v1/wwTestReport

The wwTestReport returns the header portion of the published test reports such as the Supplier, Lab, Date Collected and the TRID. Test reports are limited to either a supplier's own test reports or a brand may view test report if there is an accepted connection between the brand and a supplier.

v1/wwTestReportParameterValue

Using the TRID, a user can obtain all test report values. Key columns include

- result_na

  If the result was lost, bad (didn't meet the labs quality standards), or not requested.

- result_nd

  The non-detect (nd) flag is either 'Yes' or 'No'. An ND is always a pass.

- result_numeric

  If the lab was able to obtain a numeric result. The numeric result will never include a qualififer such as < or >.

- Evaluation

  The evaluation is a simply way to determine if the result is above or below the limit

  - MR – The result is at or below the MRSL reporting threshold.
  - MA – The result is at or below the Aspirational limit.
  - MP – The result is at or below the Progressive limit.
  - MF – The result is at or below the Foundational Limit.
  - ALERT - If the value is above the Foundational Limit (MF) for Conventionals or the MRSL Reporting Limit (MR) the evaluation is alert.
  - NA – When an NA is submitted.

v1/wwLabReportPdf

Returns the actual lab PDF report as a binary string.

v1/wwTestReportLastUpdate

A list of all test reports and the last updated flag. This is used to synchronize local databases to the Gateway values without having to pull back the entire data set.

v1/wwClearstreamReport

The summary percentage presented on the Clearstream report.

v1/wwReportingPeriodSummary

Combines multiple test reports reported during a Reporting Period into 1 set of results per Supplier.

v1/wwParameterList

The list of ZDHC Parameters and the limits by ZDHC reporting guideline.

Code Examples

All code examples are written in R and assumes the following set-up.

```
## R libraries referenced in the code examples

# PARSE JSON
library(jsonlite)

# CALL APIs
library(httr)

# DATA SHAPING
library(tidyverse)

# DATA CLEANING
library(janitor)

# SET THE DEFAULT BASE URL
url_base = "https://www.my-aip.com/Gateway/rest/"
```

TokenByKey POST

The TokenByKey

```
## TokenByKey

f_api_oauth_tokenbykey <- function() {


  # SET THE CREDENTIALS
  cred='{
    "UserAccessKey":"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA",
    "authenticationKey":"ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ"
    }'

  # SET THE PATH
  path = "https://www.my-aip.com/Gateway/rest/oauth/tokenbykey"


  # POST THE REQUEST
  request <- POST(url = path,body=cred,
                  httr::add_headers('accept' = 'application/json'),
                  httr::content_type('application/json')
          )

  response <- content(request, as = "parsed")
  return(response$token$accessToken)

}
```

Generic GET Function

The following function can be used to call the GET APIs

```
# GENERIC FUNCTION FOR GET WASTEWATER APIS


f_ww_api <- function(end_point,param_set,version='v1/') {

  # AUTHENTICATE
  auth_token =  f_api_oauth_tokenbykey()

  # BUILD THE PATH
  path = str_c(url_base,version,end_point,param_set)

  # CALL API AND CONVERT TO A DATA FRAME
  request <- GET(url = path,verbose(),
                 httr::add_headers('accessToken' = auth_token),
                 httr::content_type('application/json'))

  response <- content(request, as = "text")

  df <- fromJSON(response, flatten = TRUE) %>% data.frame() %>% tibble()


  # ERROR HANDLING
  if (names(df[1]) =='errorMessage') { stop(df[1]) }


  # CLEAN
  df <- df %>% select(-success) %>%
      rename_all(funs(str_remove(.,"data."))) %>%
      janitor::clean_names()

  # RETURN DATA
  return(df)

}
```

Generic POST Function

The following function can be used to call the POST APIs

```
# GENERIC FUNCTION FOR POST WASTEWATER APIS

f_ww_api_post <- function(end_point,param_set,version='v1/') {

  auth_token =  f_api_oauth_tokenbykey()
  path = str_c(url_base,version,end_point)
  body <- list(ParameterCodeList = param_set)

  request <- POST(url = path,verbose(),body =param_set,
                  httr::add_headers('accessToken' = auth_token),
                  httr::content_type('application/json'))

  response <- content(request, as = "text")


  df <- fromJSON(response, flatten = TRUE) %>%
      data.frame() %>% tibble()

  if (names(df[1]) =='errorMessage') { stop(df[1]) }

  df <- df %>% select(-success) %>%
      rename_all(funs(str_remove(.,"data."))) %>%
      janitor::clean_names()

  return(df)

}
```