Integration of GCP BigQuery with CADP

GCP BigQuery [Overview]

This document describes how to configure and integrate CipherTrust Manager with GCP BigQuery.

BigQuery is a fully managed enterprise data warehouse that helps you manage and analyze your data with built-in features like machine learning, geospatial analysis, and business intelligence. BigQuery's serverless architecture lets you use SQL queries to answer your organization's biggest questions with zero infrastructure management. Federated queries let you read data from external sources while streaming supports continuous data updates. BigQuery's scalable, distributed analysis engine lets you query terabytes in seconds and petabytes in minutes..

Thales provides a couple of different methods to protect sensitive data in GCP BigQuery.

Bring Your Own Encryption (BYOE)

- **Data Ingest** – with Thales Batch Data Transformation (BDT)
- **Data Access** – external remote user defined functions for column level encrypt and decryption using Thales CADP and tokenization using Thales CT-VL.

Bring/Hold Your Own Key (BYOK) (HYOK)

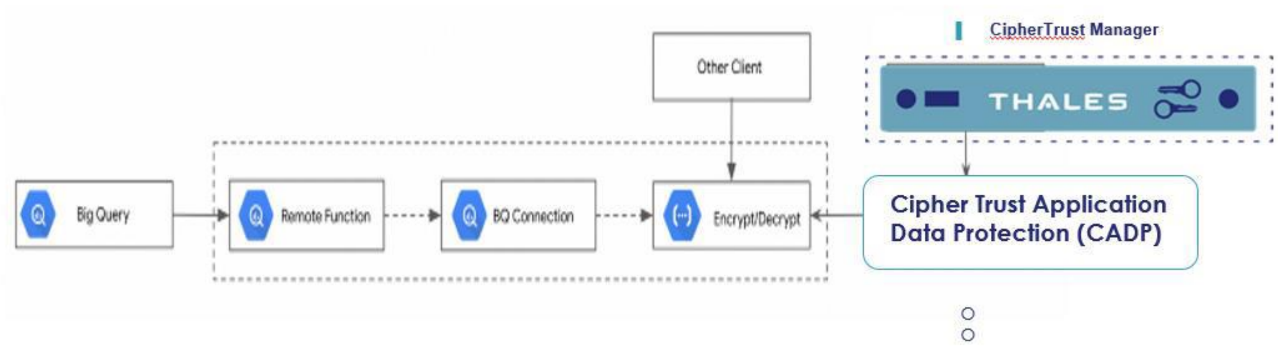
- **GCP BigQuery Customer Managed Keys-** with Thales CM CCKM BYOK and HYOK.

The above methods are NOT mutually exclusive. All methods can be used to build a strong defense in depth strategy to protect sensitive data in the cloud. The focus of this integration will be on Data Access protecting sensitive data in GCP BigQuery columns by using CADP to create User Defined Functions (UDF) for encryption and decryption of sensitive data.

Architecture

The examples provided in this document use a capability GCP BigQuery called "Remote Function". A BigQuery remote function lets you incorporate GoogleSQL functionality with software outside of BigQuery by providing a direct integration with Cloud Functions and Cloud Run. With BigQuery remote functions, you can deploy your functions in Cloud Functions or Cloud Run implemented with any supported language, and then invoke them from GoogleSQL queries.

A BigQuery remote function allows you to implement your function in other languages than SQL and Javascript or with the libraries or services which are not allowed in BigQuery user-defined functions. Listed below is a diagram of how this integration works.



Supported Product Versions

- **CipherTrust Manager** CipherTrust Manager 2.11 and higher (if usersets are used then CM 2.14 or greater should be used)
- **CADP** CADP Java 8.13 and higher
- **GCP BigQuery**

This integration is validated using 2nd generation Google Cloud Functions and Java 11 along with CM 2.14 and CADP 8.16.0.000

Prerequisites

Steps performed for this integration were provided by this GCP link:
https://cloud.google.com/bigquery/docs/reference/standard-sql/remote-functions

https://cloud.google.com/bigquery/docs/remote-function-tutorial#console

- Ensure that CADP for Java is installed and configured. Refer to https://thalesdocs.com/ctp/con/cadp/cadp-java/latest/admin/cadp-for-java-quick-start/cadp-for-java-installer/index.html
- Ensure that the CipherTrust Manager is installed and configured. Refer to the **CipherTrust Manager documentation** for details.

- GCP Cloud function communicates with the CipherTrust Manager using the Network Attached Encryption (NAE) Interface. Ensure that the NAE interface is configured. Refer to the **CipherTrust Manager documentation** for details.

Steps for Integration

- **[Installing and Configuring Thales CADP for Java]**
- **[Download code from Thales github and compile]**
- **[Publish jar/zip file to GCP Cloud Function (endpoint)]**
- **[Create BigQuery Connection and GCP BigQuery Remote Function]**
- **[Integration with Thales CipherTrust Manager]**
- **[Environment Variables]**

Installing and Configuring CADP for Java

To install and configure **CADP for Java,** refer to Quick Start.

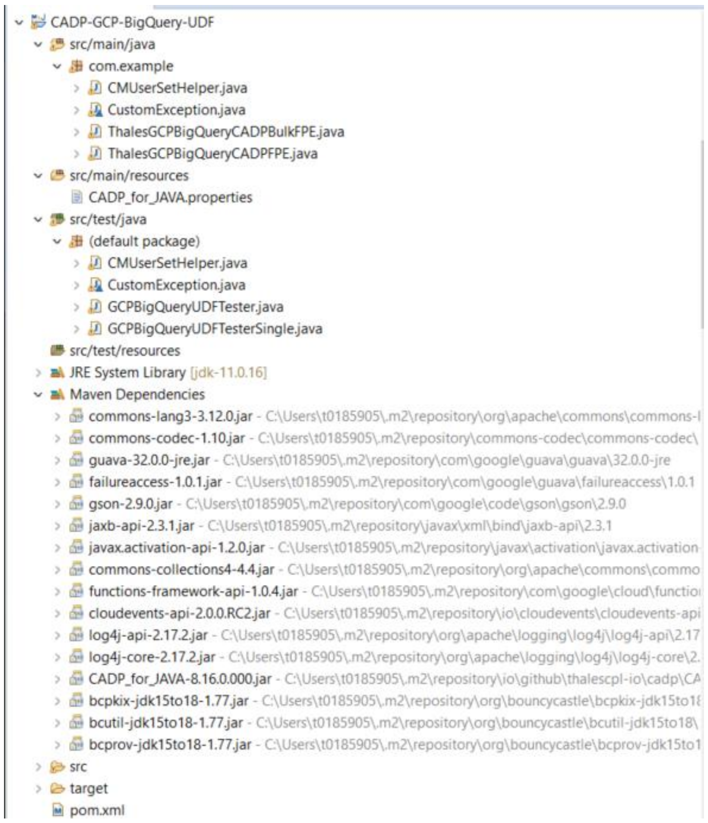
As noted in the link above Thales provides CADP in the Maven repository. This can be found at: https://central.sonatype.com/artifact/io.github.thalescpl-io.cadp/CADP_for_JAVA

Eclipse development tool was used for these examples. Here is the version used for testing along with the Maven plugin for Eclipse.

eclipse.buildId=4.15.0.I20200305-0155

m2e - Maven Integration for Eclipse

Here is a screenshot in eclipse of the pom file used for these examples:



```
CADP-GCP-BigQuery-UDF
  src/main/java
    com.example
      CMUserSetHelper.java
      CustomException.java
      ThalesGCPBigQueryCADPBulkFPE.java
      ThalesGCPBigQueryCADPFPE.java
  src/main/resources
    CADP_for_JAVA.properties
  src/test/java
    (default package)
      CMUserSetHelper.java
      CustomException.java
      GCPBigQueryUDFTester.java
      GCPBigQueryUDFTesterSingle.java
  src/test/resources
  JRE System Library [jdk-11.0.16]
  Maven Dependencies
    commons-lang3-3.12.0.jar - C:\Users\t0185905\.m2\repository\org\apache\commons\commons-l
    commons-codec-1.10.jar - C:\Users\t0185905\.m2\repository\commons-codec\commons-codec\
    guava-32.0.0-jre.jar - C:\Users\t0185905\.m2\repository\com\google\guava\guava\32.0.0-jre
    failureaccess-1.0.1.jar - C:\Users\t0185905\.m2\repository\com\google\guava\failureaccess\1.0.1
    gson-2.9.0.jar - C:\Users\t0185905\.m2\repository\com\google\code\gson\gson\2.9.0
    jaxb-api-2.3.1.jar - C:\Users\t0185905\.m2\repository\javax\xml\bind\jaxb-api\2.3.1
    javax.activation-api-1.2.0.jar - C:\Users\t0185905\.m2\repository\javax\activation\javax.activation-
    commons-collections4-4.4.jar - C:\Users\t0185905\.m2\repository\org\apache\commons\commo
    functions-framework-api-1.0.4.jar - C:\Users\t0185905\.m2\repository\com\google\cloud\functio
    cloudevents-api-2.0.0.RC2.jar - C:\Users\t0185905\.m2\repository\io\cloudevents\cloudevents-api
    log4j-api-2.17.2.jar - C:\Users\t0185905\.m2\repository\org\apache\logging\log4j\log4j-api\2.17
    log4j-core-2.17.2.jar - C:\Users\t0185905\.m2\repository\org\apache\logging\log4j\log4j-core\2.
    CADP_for_JAVA-8.16.0.000.jar - C:\Users\t0185905\.m2\repository\io\github\thalescpl-io\cadp\CA
    bcpkix-jdk15to18-1.77.jar - C:\Users\t0185905\.m2\repository\org\bouncycastle\bcpkix-jdk15to18
    bcutil-jdk15to18-1.77.jar - C:\Users\t0185905\.m2\repository\org\bouncycastle\bcutil-jdk15to18\
    bcprov-jdk15to18-1.77.jar - C:\Users\t0185905\.m2\repository\org\bouncycastle\bcprov-jdk15to1
  src
  target
  pom.xml
```

Download code from github and compile.

git clone https://github.com/ThalesGroup/CipherTrust_Application_Protection.git

The database directory has all the code for GCP BigQuery. You should see the above classes in your project.

CADP supports a bulk API which allows for CADP to batch requests before calling encrypt or decrypt. A separate class file for each datatype is available for testing with this API. There is a limit of 10,000 items to encrypt but it also depends on the size of the items as well so review the CADP documentation to make sure those thresholds are not exceeded.

Assuming you have your CM already configured the `GCPBigQueryUDFTester` can be used to test basic connection to CM to make sure your CM environment is configured correctly. You will need to modify the keyname to make sure it exists in CM and there are environment variables that must be set as well. If you plan on using UserSets the user must also be in the Application Data Protection Client Group. See section on Environment Variables for more details. If you have already installed CM then you will need to update the CADP_for_JAVA.properties file with all the necessary settings such as IP/NAE Port, etc. The file is located under the resource's directory in the java project for eclipse.

Generate the jar file to upload to the CSP.

To compile and generate the target jar file to be uploaded to AWS Lambda select the project and choose “Run As” “maven install” to generate the target.

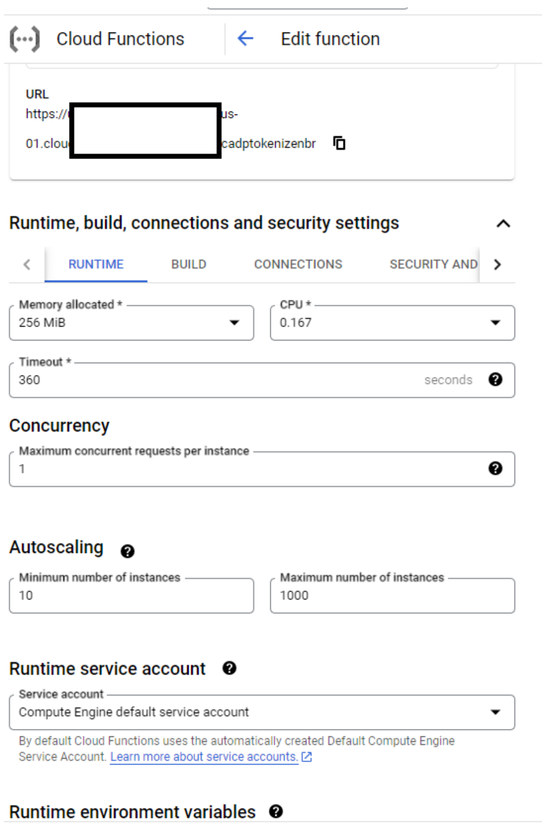
```
[INFO] --- maven-install-plugin:3.0.1:install (default-install) @ CADP-GCP-Function -
--
[INFO] Installing C:\Users\t0185905\workspace\CADP-GCP-Function\pom.xml to
C:\Users\t0185905\.m2\repository\Thales\CADP-GCP-Function\0.0.1-SNAPSHOT\CADP-GCP-
Function-0.0.1-SNAPSHOT.pom
[INFO] Installing C:\Users\t0185905\workspace\CADP-GCP-Function\target\CADP-GCP-
Function-0.0.1-SNAPSHOT.jar to C:\Users\t0185905\.m2\repository\Thales\CADP-GCP-
Function\0.0.1-SNAPSHOT\CADP-GCP-Function-0.0.1-SNAPSHOT.jar
[INFO] Installing C:\Users\t0185905\workspace\CADP-GCP-Function\target\CADP-GCP-
Function-0.0.1-SNAPSHOT-jar-with-dependencies.jar to
C:\Users\t0185905\.m2\repository\Thales\CADP-GCP-Function\0.0.1-SNAPSHOT\CADP-GCP-
Function-0.0.1-SNAPSHOT-jar-with-dependencies.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  7.250 s
[INFO] Finished at: 2024-02-28T10:56:43-05:00
[INFO] ------------------------------------------------------------------------The
```

The code provided has uses Googles userDefinedContext capability. This code accepts a mode and a datatype as a keyvalue pair.

Publish jar/zip file Google Cloud Function. (endpoint)

Once you have generated the jar file to upload you can then create the CSP function.   Google requires a zip file so zip up the jar file in the target directory of your eclipse project.

GCP Cloud Function


Cloud Functions | Edit function

URL
https://[redacted]us-
01.clou[redacted]cadptokenizenbr

Runtime, build, connections and security settings

RUNTIME | BUILD | CONNECTIONS | SECURITY AND

Memory allocated *: 256 MiB
CPU *: 0.167
Timeout *: 360 seconds

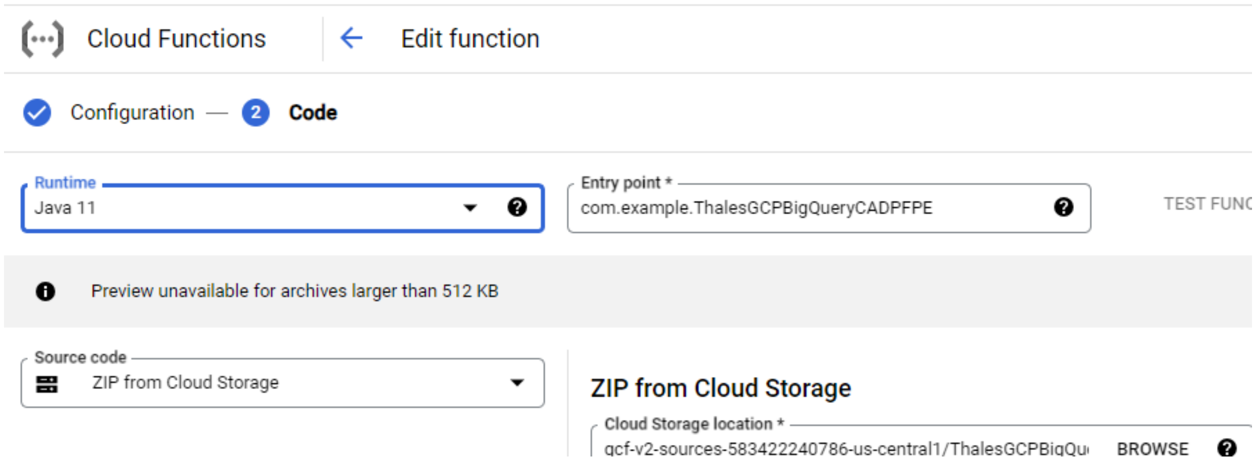
Concurrency
Maximum concurrent requests per instance: 1

Autoscaling
Minimum number of instances: 10
Maximum number of instances: 1000

Runtime service account
Service account: Compute Engine default service account
By default Cloud Functions uses the automatically created Default Compute Engine Service Account. Learn more about service accounts.

Runtime environment variables

Set environment variables appropriate values (See section on Environment Variables for details) and then select Next. Upload the zip file on the next screen. Be sure to change the entry point to reflect your class name. ***The example below is com.example.ThalesGCPBigQueryCADPFPE This should match your code path.***


Cloud Functions | Edit function

Configuration — 2 Code

Runtime: Java 11
Entry point *: com.example.ThalesGCPBigQueryCADPFPE
TEST FUNC

Preview unavailable for archives larger than 512 KB

Source code: ZIP from Cloud Storage

ZIP from Cloud Storage
Cloud Storage location *: gcf-v2-sources-583422240786-us-central1/ThalesGCPBigQu
BROWSE

Click Deploy to deploy the function.

Once you have created the functions above and if you have already configured and setup CM with the key and all the environment variables you can test the function with the test tab. You will need to provide the appropriate json to test. Here is an example:

```
{
  "requestId": "124ab1c",
  "caller":
"//bigquery.googleapis.com/projects/myproject/jobs/myproject:US.bquxjob_5b4c112c_17961fafeaf",
  "sessionUser": "test-user@test-company.com",
  "userDefinedContext": {
    "mode": "decrypt",
    "datatype": "nbr"
  },
  "calls": [
    [
      93309296
    ],
    [
      74705755
    ],
        [
      39056597430
    ],
    [
      6621883
    ]
        ,
        [
      2662402956
    ],
    [
      17506289853
    ]
  ]
}
```

Create and configure BigQuery connection and GCP BigQuery Remote Function.

Here are some links that provide details.

https://cloud.google.com/bigquery/docs/remote-function-tutorial#console

https://cloud.google.com/bigquery/docs/remote-functions

As noted above the steps are:

1. Create the GCP Cloud Function. (should already be done from above)
2. Create GCP BigQuery Connection.
3. Create remote function object in GCP BigQuery

Examples:

```
bq mk --connection --display_name='warnerscorner' --connection_type=CLOUD_RESOURCE --
project_id=yourprojectid --location=US mw-remote-add-conn
```

Big Query Function Definitions.

The functions are created using the following format:

data type in db/actual data format/return type

Examples:

```
SELECT `abc-sales-l-app-us-
02.your_demo_dataset_US.thales_cadp_encrypt_char_int_char`
('123382742348293479')
093381171100859737

SELECT `abc-sales-l-app-us-
02.your_demo_dataset_US.thales_cadp_encrypt_char_int`
('123382742348293479')
93381171100859737

SELECT `abc-sales-l-app-us-
02.your_demo_dataset_US.thales_cadp_encrypt_nbr` (123382742348293479)
93381171100859737
```

```
CREATE or replace FUNCTION `abc-sales-l-app-us-
02.your_demo_dataset_US.thales_cadp_encrypt_char`(x String)
RETURNS String
REMOTE WITH CONNECTION `abc-sales-l-app-us-02.us.mw-remote-add-conn`
OPTIONS (
endpoint = 'https://us-central1-abc-sales-l-app-us-
02.cloudfunctions.net/ThalesGCPBigQueryCADPFPE',
user_defined_context = [("mode", "encrypt"),("datatype", "char")]
);
CREATE or replace FUNCTION `abc-sales-l-app-us-
02.your_demo_dataset_US.thales_cadp_decrypt_char`(x String)
RETURNS String
REMOTE WITH CONNECTION `abc-sales-l-app-us-02.us.mw-remote-add-conn`
OPTIONS (
endpoint = 'https://us-central1-abc-sales-l-app-us-
02.cloudfunctions.net/ThalesGCPBigQueryCADPFPE',
user_defined_context = [("mode", "decrypt"),("datatype", "char")]
);

CREATE or replace FUNCTION `abc-sales-l-app-us-
02.your_demo_dataset_US.thales_cadp_decrypt_char_int`(x String)
RETURNS int64
REMOTE WITH CONNECTION `abc-sales-l-app-us-02.us.mw-remote-add-conn`
OPTIONS (
endpoint = 'https://us-central1-abc-sales-l-app-us-
02.cloudfunctions.net/ThalesGCPBigQueryCADPFPE',
user_defined_context = [("mode", "decrypt"),("datatype", "charint")]
);

CREATE or replace FUNCTION `abc-sales-l-app-us-
02.your_demo_dataset_US.thales_cadp_encrypt_char_int`(x String)
RETURNS int64
REMOTE WITH CONNECTION `abc-sales-l-app-us-02.us.mw-remote-add-conn`
OPTIONS (
endpoint = 'https://us-central1-abc-sales-l-app-us-
02.cloudfunctions.net/ThalesGCPBigQueryCADPFPE',
user_defined_context = [("mode", "encrypt"),("datatype", "charint")]
);

CREATE or replace FUNCTION `abc-sales-l-app-us-
02.your_demo_dataset_US.thales_cadp_decrypt_char_int_char`(x String)
RETURNS String
REMOTE WITH CONNECTION `abc-sales-l-app-us-02.us.mw-remote-add-conn`
OPTIONS (
endpoint = 'https://us-central1-abc-sales-l-app-us-
02.cloudfunctions.net/ThalesGCPBigQueryCADPFPE',
user_defined_context = [("mode", "decrypt"),("datatype", "charintchar")]
);

CREATE or replace FUNCTION `abc-sales-l-app-us-
02.your_demo_dataset_US.thales_cadp_encrypt_char_int_char`(x String)
RETURNS String
REMOTE WITH CONNECTION `abc-sales-l-app-us-02.us.mw-remote-add-conn`
OPTIONS (
endpoint = 'https://us-central1-abc-sales-l-app-us-
02.cloudfunctions.net/ThalesGCPBigQueryCADPFPE',
user_defined_context = [("mode", "encrypt"),("datatype", "charintchar")]
);

CREATE or replace FUNCTION `abc-sales-l-app-us-
02.your_demo_dataset_US.thales_cadp_encrypt_nbr`(x int64)
RETURNS int64
REMOTE WITH CONNECTION `abc-sales-l-app-us-02.us.mw-remote-add-conn`
OPTIONS (
endpoint = 'https://us-central1-abc-sales-l-app-us-
02.cloudfunctions.net/ThalesGCPBigQueryCADPFPE',
user_defined_context = [("mode", "encrypt"),("datatype", "nbr")]
);
CREATE or replace FUNCTION `abc-sales-l-app-us-
02.your_demo_dataset_US.thales_cadp_decrypt_nbr`(x int64)
RETURNS int64
REMOTE WITH CONNECTION `abc-sales-l-app-us-02.us.mw-remote-add-conn`
OPTIONS (
endpoint = 'https://us-central1-abc-sales-l-app-us-
02.cloudfunctions.net/ThalesGCPBigQueryCADPFPE',
user_defined_context = [("mode", "decrypt"),("datatype", "nbr")]
);
```

Integration with CipherTrust Manager.

Assuming the NAE interface has already been setup based on the prerequisites the only other setup required is to have a user and key created in CM. This user should have the ability to export the key. This user will also be needed to be provided as an environment variable for the function. The examples provided have the key as a hardcoded value, but this can be easily altered to be provided as an environment variable, obtained from a secrets manager or in the userDefinedContext of the create function statement.

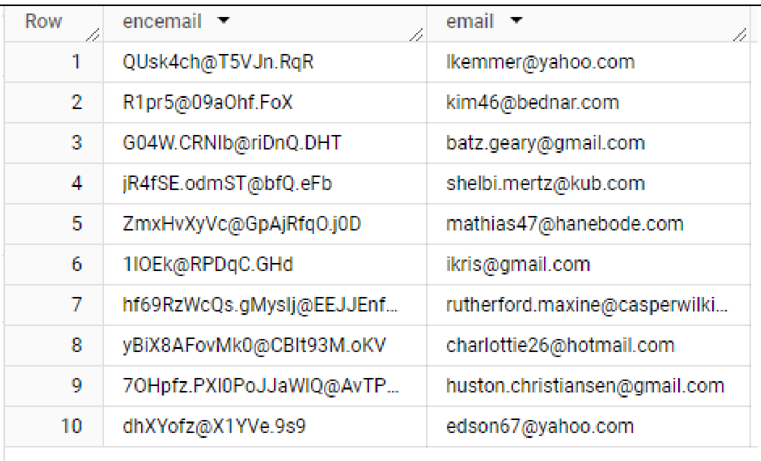
As noted above there is a test class that can be used to test connectivity with CM without having to publish the Function. If you have not updated the CADP_for_JAVA.properties file with the CM settings such as IP/NAE Port, etc. then do so now. The file is located under the resource’s directory in the eclipse project. These properties can also be overwritten with CADP code as well if your desire is to pass them in as environment variables or headers of the JSON request. Here is an example:

```
System.setProperty("com.ingrian.security.nae.NAE_IP.1", "10.20.1.9");
```

When all the above steps are performed you should see your UDF’s in GCP BigQuery under Routines in the UI. Here is a sample query using one of the UDF’s.

Sample Results:

```
select `abc-sales-l-app-us-02.your_demo_dataset_US.thales_cadp_encrypt_char` (email)
as encemail, email from `abc-sales-l-app-us-02.your_demo_dataset_US.plaintext` limit
10
```



Row	encemail	email
1	QUsk4ch@T5VJn.RqR	lkemmer@yahoo.com
2	R1pr5@09aOhf.FoX	kim46@bednar.com
3	G04W.CRNIb@riDnQ.DHT	batz.geary@gmail.com
4	jR4fSE.odmST@bfQ.eFb	shelbi.mertz@kub.com
5	ZmxHvXyVc@GpAjRfqO.j0D	mathias47@hanebode.com
6	1IOEk@RPDqC.GHd	ikris@gmail.com
7	hf69RzWcQs.gMyslj@EEJJEnf...	rutherford.maxine@casperwilki...
8	yBiX8AFovMk0@CBIt93M.oKV	charlottie26@hotmail.com
9	7OHpfz.PXI0PoJJaWIQ@AvTP...	huston.christiansen@gmail.com
10	dhXYofz@X1YVe.9s9	edson67@yahoo.com

Environment Variables

Listed below are the environment variables currently required for the Cloud Function.

```
String userName = System.getenv("CMUSER");
// If you plan on using UserSets the cmuser must also be in the Application Data
Protection Client Group.
String password = System.getenv("CMPWD");
// returnciphertextforuserwithnokeyaccess = is na environment variable to express
// how data should be returned when the user above does not have access to the key
//and if doing a lookup in the userset and the user does not exist. If
//returnciphertextforuserwithnokeyaccess = null
// then an error will be returned to the query, else the results set will provide
ciphertext.
// validvalues are 1 or null
// 1 will return cipher text
// null will return error.
String returnciphertextforuserwithnokeyaccess =
System.getenv("returnciphertextforuserwithnokeyaccess");
// usersetlookup = should a userset lookup be done on the user from Big Query? 1
// = true 0 = false.
String usersetlookup = System.getenv("usersetlookup");
```

```
// usersetID = should be the usersetid in CM to query.
String usersetID = System.getenv("usersetidincm");
// usersetlookupip = this is the IP address to query the userset. Currently it
// is the userset in CM but could be a memcache or other in memory db.
String userSetLookupIP = System.getenv("usersetlookupip");
```

Advanced Topics.

Google also provides the ability to setup a secure perimeter with VPC Service Controls. See link below for more information.

https://cloud.google.com/bigquery/docs/remote-functions#using_vpc_service_controls

Application Data Protection UserSets

Application Data Protection UserSets are currently used for DPG to control how the data will be revealed to users. These UserSets can also be independent of any Access Policy. Most cloud databases have some way to capture who is running the query and this information can be passed to CM to be verified in a UserSet to ensure the person running the query has been granted proper access. In github there is a sample class file called CMUserSetHelper that can be used to load a userset with values from an external identity provider such as LDAP. The name of this method is `addAUserToUserSetFromFile`. Once users have been loaded into this userset the usersetid must be captured and used as an environment variable to the Function. The function has a number of environment variables that must be provided for the function to work. Please review the section on Environment Variables for more details.

Options for keyname

Currently the sample code uses a hard coded key name. Other options to be considered are:

1.)keyname as an environment variable.

2.)keyname as a GCP secret.

3.)keyname defined as a userDefinedContect. This would enable one Cloud Function with multiple BigQuery Functions. Users would then only be granted access to the function that is relevant to them.

```
CREATE or replace FUNCTION `abc-sales-l-app-us-
02.your_demo_dataset_US.thales_cadp_decrypt_nbr`(x int64)
RETURNS int64
REMOTE WITH CONNECTION `abc-sales-l-app-us-02.us.mw-remote-add-conn`
OPTIONS (
endpoint = 'https://us-central1-abc-sales-l-app-us-
02.cloudfunctions.net/ThalesGCPBigQueryCADPFPE',
user_defined_context = [("mode", "decrypt") ("keyname","yourkey"),
("datatype", "nbr")]);
```

4.)keyname passed in an attribute to the function. Example with a database view.

create view employee as select first_name, last_name,
thales_cadp_aws_decrypt_char('testfaas', email) as email from emp_basic

Options for handling null values.

Since it is not possible to encrypt a column that contains null values or any column that has 1 byte it is necessary to skip those to avoid getting an error when running the query. There are a couple of ways to handle this use case.

Option 1. Modify the queries.

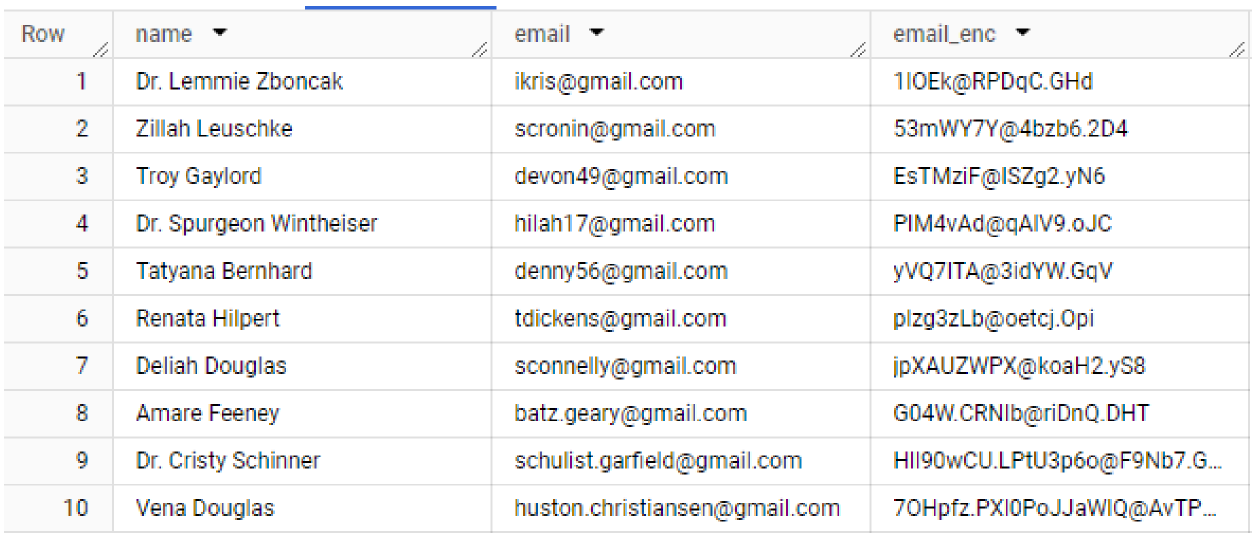
Many times, simply adding a where clause to exclude values that have nulls or less than 2 bytes can avoid query errors. For example: select * from FROM
  mw_demo_dataset_US.plaintext50cadpemailprotected_nulltest where email is not null and length(email) > 1; For those scenarios where that is not suffice some other examples are listed below.   Here is an example of a select statement that can be modified to handle null values:

```
SELECT
  name,
  CASE
    WHEN email IS NULL THEN 'null'
    WHEN length(email) < 2 then email
    WHEN email = 'null' then email
    ELSE `cpl-sales-l-app-us-01.mw_demo_dataset_US.thales_cadp_encrypt_char`(email)
  END AS email
FROM
  mw_demo_dataset_US.plaintext50cadpemailprotected_nulltest;
```

The above use case was for situations where the column contained both null and the word 'null'.

Here is an example of a select statement that can be modified to handle null values in the where clause.

```
    SELECT name, email, email_enc
FROM
  mw_demo_dataset_US.plaintext50cadpemailprotected_nulltest
  where CASE
    WHEN email_enc IS NULL THEN 'null'
    WHEN length(email_enc) < 2 then email_enc
    WHEN email_enc = 'null' then email_enc
    ELSE  `cpl-sales-l-app-us-
01.mw_demo_dataset_US.thales_cadp_decrypt_char`(email_enc)
  END  like "%gmail%"
```



Row	name	email	email_enc
1	Dr. Lemmie Zboncak	ikris@gmail.com	1lOEk@RPDqC.GHd
2	Zillah Leuschke	scronin@gmail.com	53mWY7Y@4bzb6.2D4
3	Troy Gaylord	devon49@gmail.com	EsTMziF@ISZg2.yN6
4	Dr. Spurgeon Wintheiser	hilah17@gmail.com	PIM4vAd@qAIV9.oJC
5	Tatyana Bernhard	denny56@gmail.com	yVQ7ITA@3idYW.GqV
6	Renata Hilpert	tdickens@gmail.com	plzg3zLb@oetcj.Opi
7	Deliah Douglas	sconnelly@gmail.com	jpXAUZWPX@koaH2.yS8
8	Amare Feeney	batz.geary@gmail.com	G04W.CRNlb@riDnQ.DHT
9	Dr. Cristy Schinner	schulist.garfield@gmail.com	HIl90wCU.LPtU3p6o@F9Nb7.G...
10	Vena Douglas	huston.christiansen@gmail.com	7OHpfz.PXI0PoJJaWIQ@AvTP...

Option 2. Modify the Cloud Function to skip encrypting these values.

Include an if statement checking for null values the word ‘null’ or any value less than 2.

```
JsonArray bigquerytrow = bigquerydata.get(i).getAsJsonArray();
if (bigquerytrow != null && bigquerytrow.size() > 0) {
        JsonElement element = bigquerytrow.get(0);
        if (element != null && !element.isJsonNull()) {
                sensitive = element.getAsString();
                if (sensitive.isEmpty() || sensitive.length() < 2) {
                        encdata = sensitive;
                } else {
                if (sensitive.equalsIgnoreCase("null")) {
                                encdata = sensitive;
                        } else {
                        byte[] outbuf = thalesCipher.doFinal(sensitive.getBytes());
                                encdata = new String(outbuf);
                        }
                }
        } else {
                encdata = sensitive;
        }
} else {
        encdata = sensitive;
}
```

Note: When using this logic it is important to know that for any column that is null will return ‘null’. This should be fine for use cases where the query is not updating the column. For use cases where the source system is expecting null vs the word ‘null’ additional testing should be conducted on the systems that rely on this data type as being null vs the word ‘null’.