# Integration of AWS Redshift with CipherTrust REST Data Protection (CRDP)

## AWS Redshift [Overview]

This document describes how to configure and integrate CipherTrust Manager with AWS Redshift.

Amazon Redshift is a data warehouse product which forms part of the larger cloud-computing platform Amazon Web Services. It is built on top of technology from the massive parallel processing (MPP) data warehouse company ParAccel to handle large scale data sets and database migrations. Redshift differs from Amazon's other hosted database offering, Amazon RDS, in its ability to handle analytic workloads on big data data sets stored by a column-oriented DBMS principle. Redshift allows up to 16 petabytes of data on a cluster compared to Amazon RDS Aurora's maximum size of 128 tebibytes. Thales provides a couple of different methods to protect sensitive data in AWS Redshift.

**Bring Your Own Encryption (BYOE)**

- **Data Ingest** – with Thales Batch Data Transformation (BDT)
- **Data Access** – external functions for column level encrypt and decryption using Ciphertrust REST Data Protection (CRDP) Protect/Reveal API's.

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

- **AWS Redshift Customer Managed Keys-** with Thales CM CCKM BYOK and HYOK.

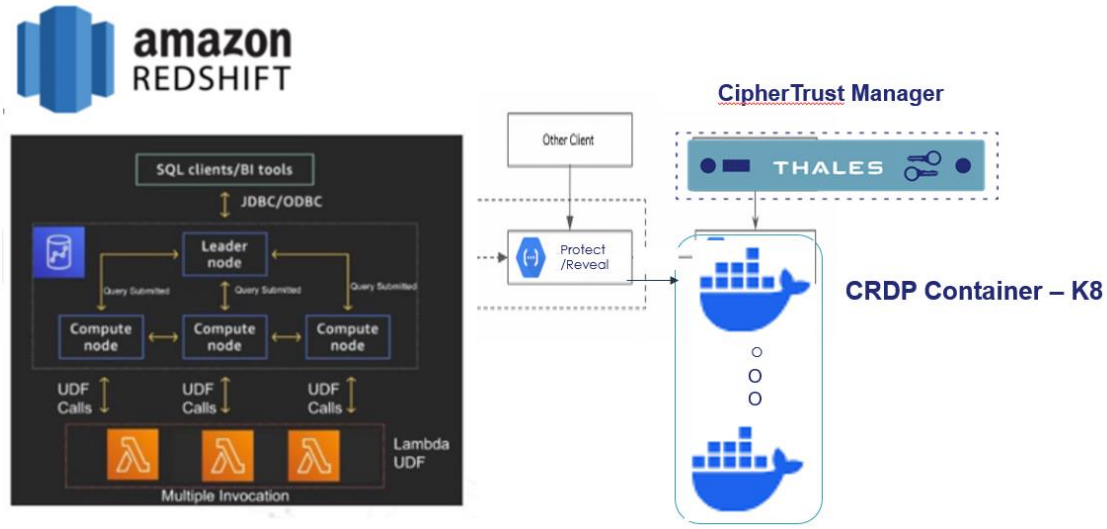
**The above methods are NOT mutually exclusive. All methods can be used to build a strong defense in depth strategy to protect sensitive data in the cloud. The focus of this integration will be on Data Access protecting sensitive data in AWS Redshift columns by using CRDP to create User Defined Functions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

**Architecture**

The examples provided in this document use a capability AWS Redshift called "External Function". A Redshift external function lets you incorporate functionality with software outside of Redshift by providing a direct integration with AWS Lambda Functions. A Redshift external function allows you to implement your function in other languages other than SQL. Listed

below is a diagram of how this integration works.

**AWS Redshift integration with Thales CRDP (AWS Lambda Function)**




# Supported Product Versions

- **CipherTrust Manager** CipherTrust Manager 2.14 and higher
- **CRDP** 1.0 and higher
- **AWS Redshift**

This integration is validated AWS Lambda Functions and Java 11 along with CM 2.14

# Prerequisites

Steps performed for this integration were provided by this AWS link:
https://docs.aws.amazon.com/redshift/latest/dg/udf-creating-a-lambda-sql-udf.html

https://docs.aws.amazon.com/redshift/latest/dg/r_CREATE_EXTERNAL_FUNCTION.html

- Ensure that CRDP for is installed and configured. Refer to https://thalesdocs.com/ctp/con/crdp/latest/admin/index.html
- Ensure that the CipherTrust Manager is installed and configured. Refer to the **CipherTrust Manager documentation** for details.
- AWS Lambda functions communicates with the CRDP Container using REST API's

# Steps for Integration

- **[Installing and Configuring Thales CRDP for Java]**
- **[Download code from Thales github and compile]**
- **[Publish jar/zip file to AWS Lambda Function (endpoint)]**
- **[Create AWS Redshift External Function]**
- **[Integration with Thales CipherTrust Manager]**
- **[Environment Variables]**

## Installing and Configuring CRDP Container

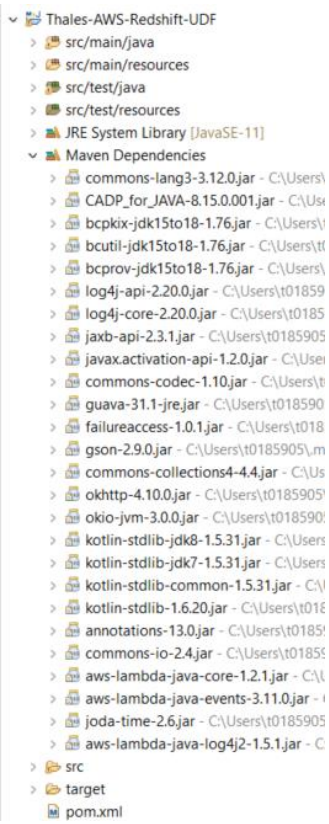
To install and configure **CDRP,** refer to Quick Start.

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

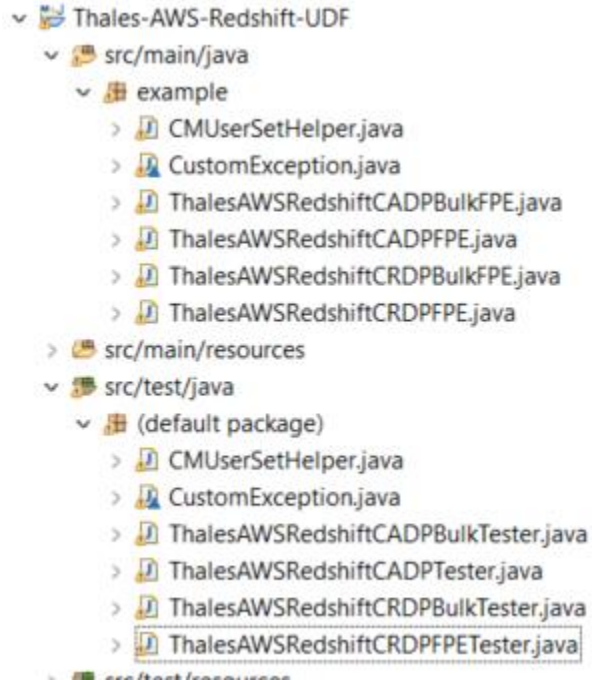
Here is a screenshot in eclipse of the pom file used for these examples:



# Download code from github and compile.

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

The database directory has all the code for AWS Redshift. You should see the classes in your project.




CRDP supports a bulk API which allows for CRDP to batch requests before calling protect or reveal. . A single class file that accepts environment variables will allow the UDF to be used by more than one UDF function.

Assuming you have your CM and your CRDP container already configured the `ThalesAWSRedshiftCRDPBulkTester` can be used to test basic connection to the CRDP container to make sure your CM environment is configured correctly. You will need to modify environment variables such as CRDPIP, BATCHSIZE and other necessary settings. Please see the appendix for all the environment variables and descriptions.

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.

**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.

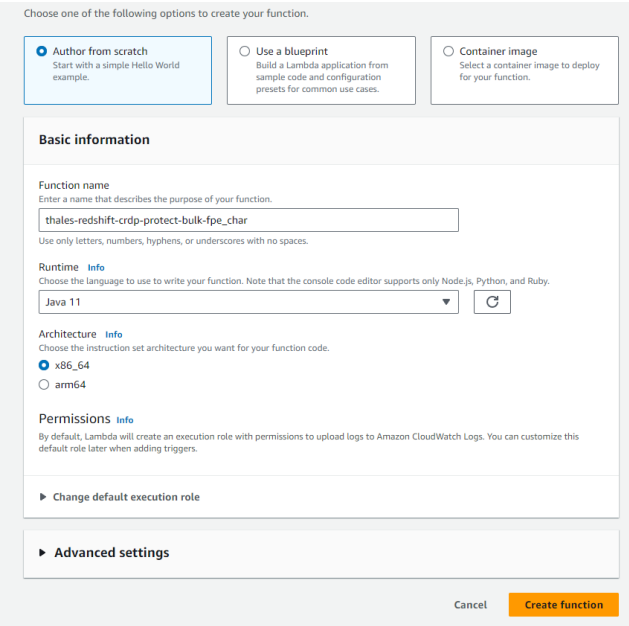
```
[INFO] Replacing original artifact with shaded artifact.
```

```
[INFO] Replacing C:\Users\t0185905\workspace\Thales-AWS-Redshift-UDF\target\Thales-
AWS-Redshift-UDF-0.0.1-SNAPSHOT.jar with C:\Users\t0185905\workspace\Thales-AWS-
Redshift-UDF\target\Thales-AWS-Redshift-UDF-0.0.1-SNAPSHOT-shaded.jar
[INFO]
[INFO] --- maven-install-plugin:3.0.1:install (default-install) @ Thales-AWS-
Redshift-UDF ---
[INFO] Installing C:\Users\t0185905\workspace\Thales-AWS-Redshift-UDF\pom.xml to
C:\Users\t0185905\.m2\repository\ThalesCPL\Thales-AWS-Redshift-UDF\0.0.1-
SNAPSHOT\Thales-AWS-Redshift-UDF-0.0.1-SNAPSHOT.pom
[INFO] Installing C:\Users\t0185905\workspace\Thales-AWS-Redshift-UDF\target\Thales-
AWS-Redshift-UDF-0.0.1-SNAPSHOT.jar to
C:\Users\t0185905\.m2\repository\ThalesCPL\Thales-AWS-Redshift-UDF\0.0.1-
SNAPSHOT\Thales-AWS-Redshift-UDF-0.0.1-SNAPSHOT.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  6.056 s
[INFO] Finished at: 2024-07-10T12:26:14-04:00
[INFO] ------------------------------------------------------------------------
```

# Publish jar/zip file to AWS Lambda Function. (endpoint)

Once you have generated the jar file to upload you can then create the CSP function.

### AWS Lambda Function




Click Create function.

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


Be sure to have a role that allows for AWSLambdaBasicExecutionRole. 512MB of RAM is plenty. You can monitor your tests and possibly lower it.

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:

```
{
  "request_id": "23FF1F97-F28A-44AA-AB67-266ED976BF40",
  "cluster": "arn:aws:redshift:xxxx",
  "user": "adminuser",
  "database": "db1",
  "external_function": "public.foo",
  "query_id": 5678234,
  "num_records": 3,
  "arguments": [
    [
      "The. king. and I me"
    ],
    [
      "The mroe data "
    ],
    [
      "test dada"
    ]
  ]
}
```

# Create AWS Redshift External Function.

Here are some links that provide details.

https://docs.aws.amazon.com/redshift/latest/dg/udf-creating-a-lambda-sql-udf.html

https://docs.aws.amazon.com/redshift/latest/dg/r_CREATE_EXTERNAL_FUNCTION.html

https://aws-dojo.com/excercises/excercise31/

As noted above the steps are:

1. Create the AWS Lambda Function. (should already be done from above)
2. Create external function object in AWS Redshift

**Examples:**

Redshift Function Definitions.

The functions are created using the following format:

```
CREATE OR REPLACE EXTERNAL FUNCTION
public.thales_crdp_protect_bulk_fpe_char(sensitivedata varchar)
 RETURNS varchar
 STABLE
lambda 'thales-redshift-crdp-protect-bulk-fpe-char'
iam_role 'arn:aws:iam::yoruproject:role/redshiftpoweruseraccess'

CREATE OR REPLACE EXTERNAL FUNCTION
public.thales_crdp_protect_fpe_char(sensitivedata varchar)
 RETURNS varchar
 STABLE
lambda 'thales-redshift-crdp-protect-fpe-char'
iam_role 'arn:aws:iam:: yoruproject:role/redshiftpoweruseraccess'
```

# Integration with CipherTrust Manager.

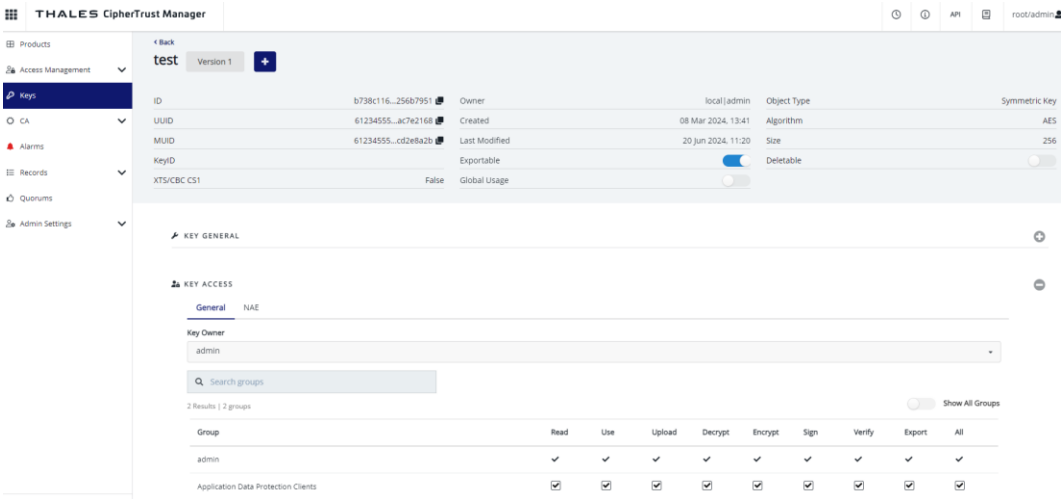
Here is a link to a demo that shows all the steps required to setup CRDP in CM.

thales.navattic.com/thalesprotectreveal

Here is a list of the steps.

1. Create an Encryption Key
2. Create a CRDP Application
3. Create a User Set
4. Create an Access Policy
5. Create a Protection Policy
6. Perform Integration

When creating a key to be used by the protection policy make sure that it is in the Application Data Protection Clients Group.

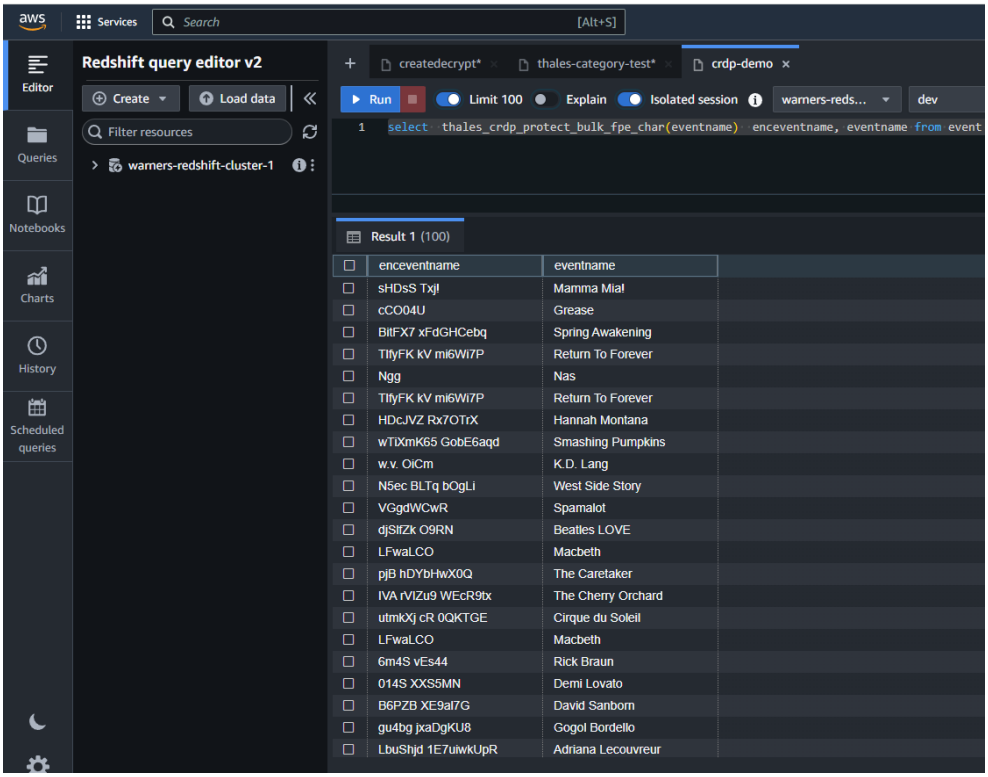



If userset lookups are desired, then when creating a user it should have the ability to export the key and also be in the Application Data Protection Clients Group. This user will also be needed to be provided as an environment variable for the function along with the password The examples provided have 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.

When all the above steps are performed you should see your UDF's in AWS Redshift UI. Here is a sample query using one of the UDF's.

*Sample Results:*




# Environment Variables

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

| Key | Value | Desc |
|---|---|---|
| BATCHSIZE | 200 | Nbr of rows to chunk when using batch mode |
| CMPWD | Yourpwd! | CM PWD if using CADP |
| CMUSER | apiuser | CM USERID if using CADP |
| CRDPIP | 20.221.216.666 | CRDP Container IP if using CRDP |
| datatype | charint | datatype of column (char or charint) |
| keymetadata | 1001000 | policy and key version if using CRDP |
| keymetadatalocation | external | location of metadata if using CRDP (internal,external) |
| mode | revealbulk | mode of operation(protect,reveal,protectbulk,revealbulk) CRDP |
| protection_profile | plain-nbr-ext | protection profile in CM for CRDP |
| returnciphertextforuserwithnokeyaccess | yes | if user in CM not exist should UDF error out or retur ciphertext |
| usersetidincm | 716f01a6-5cab-4799-925a-6dc2d8712fc1 | userset in cm if user lookup is done |
| usersetlookup | no | should uselookup be done (yes,no) |
| usersetlookupip | 20.241.70.666 | userset lookup |
| showrevealinternalkey | yes | show keymetadata when issuing a protect call (CRDP) |

# 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 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.plaintext50CRDPemailprotected_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: 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_CRDP_encrypt_char`(email)
  END AS email
FROM
  mw_demo_dataset_US.plaintext50CRDPemailprotected_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.plaintext50CRDPemailprotected_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_CRDP_decrypt_char`(email_enc)
  END  like "%gmail%"
```

| Row | name | email | email_enc |
|---|---|---|---|
| 1 | Dr. Lemmie Zboncak | ikris@gmail.com | 1IOEk@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@qAlV9.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.CRNIb@riDnQ.DHT |
| 9 | Dr. Cristy Schinner | schulist.garfield@gmail.com | HII90wCU.LPtU3p6o@F9Nb7.G... |
| 10 | Vena Douglas | huston.christiansen@gmail.com | 7OHpfz.PXI0PoJJaWIQ@AvTP... |

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

Please review the method checkvalid for the logic of handling these use cases.

*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'.*

# Sample CRDP docker commands.

As noted in the link above there are number of ways to deploy the CRDP container. For demo purposes here are the steps that were performed for a standalone use case.

```
sudo apt update
sudo apt install apt-transport-https ca-certificates curl
software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo
apt-key add -
sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu bionic stable"
sudo apt update
```

```
apt-cache policy docker-ce
sudo apt install docker-ce
sudo systemctl status docker
sudo usermod -aG docker crdpuser

su - crdpuser
docker ps
docker pull thalesciphertrust/ciphertrust-restful-data-
protection:latest
exit
//Back on as root
```

```
docker run -e KEY_MANAGER_HOST=192.168.159.134 -e
REGISTRATION_TOKEN=vw886rxzMolgtL1Gz950Ta9kVbDDvo0qgawGHthOc7iKn
m8b1VP6R2ps3qifVDTN -p 8090:8090 -e SERVER_MODE=no-tls
thalesciphertrust/ciphertrust-restful-data-protection
```

{"level":"info","time":"Thu, 20 Jun 2024 14:56:26 +0000","msg":"going to initialize shield"}

{"level":"info","time":"Thu, 20 Jun 2024 14:56:26 +0000","msg":"registerClient: Going to register the client"}

{"level":"info","time":"Thu, 20 Jun 2024 14:56:26 +0000","msg":"registerClient: Register the client successfully"}

{"level":"info","time":"Thu, 20 Jun 2024 14:56:32 +0000","msg":"","app_name":"crdpapp1","audit":true,"client_id":"be458360-e656-4faf-b7ee-0d210626b214","endpoint":"/v1/protect","jwt_username":"","key_name":"test","key_version":"1","method":"POST","protection_policy_name":"plain-alpha-internal","protection_policy_version":"4","source_ip":"192.168.159.1","status":"Success"}

{"level":"info","time":"Thu, 20 Jun 2024 14:56:45 +0000","msg":"","app_name":"crdpapp1","audit":true,"client_id":"be458360-e656-4faf-b7ee-0d210626b214","endpoint":"/v1/protect","jwt_username":"","key_name":"test","key_version":"1","method":"POST","protection_policy_name":"plain-alpha-internal","protection_policy_version":"4","source_ip":"192.168.159.1","status":"Success"}

{"level":"info","time":"Thu, 20 Jun 2024 14:56:51 +0000","msg":"","app_name":"crdpapp1","audit":true,"client_id":"be458360-e656-4faf-b7ee-0d210626b214","endpoint":"/v1/protect","jwt_username":"","key_name":"test","key_version":"1","method":"POST","protection_policy_name":"plain-alpha-internal","protection_policy_version":"4","source_ip":"192.168.159.1","status":"Success"}

{"level":"error","time":"Thu, 20 Jun 2024 14:56:57 +0000","msg":"error while encrypting the data, err: Input buffer is too short (len=1), it has to be at least 2 characters long."}

{"level":"error","time":"Thu, 20 Jun 2024 14:56:57 +0000","msg":"Input buffer is too short (len=1), it has to be at least 2 characters long.","app_name":"crdpapp1","audit":true,"client_id":"be458360-e656-4faf-b7ee-0d210626b214","endpoint":"/v1/protect","jwt_username":"","key_name":"test","key_version":"1","method":"POST","protection_policy_name":"plain-alpha-internal","protection_policy_version":"4","source_ip":"192.168.159.1","status":"Error"}