SafeNet KMIP and Google Drive Integration Guide

Documentation Version: 20130802

Table of Contents

CHAPTER 1 GOOGLE DRIVE 2

Introduction 2
Prerequisites 2
Creating Google APIs Console Project 2
Enabling Google Drive 3
Authorizing Requests with OAuth 2.0 3
SafeNet Integration 4
Configuring SafeNet Integration 4
Uploading Data Using Keys Stored on KeySecure 4
Downloading Data Using Keys Stored on KeySecure 5

CHAPTER 2 SETTING UP SSL 6

SSL Configuration Procedures 6
Creating a Local CA 6
Creating a Server Certificate Request on the Management Console 7
Signing a Server Certificate Request with a Local CA 7
Downloading the Local CA Certificate 8
Adding the CA Certificate to the Java Keystore 8
SSL with Client Certificate Authentication Procedures 8
Generating a Client Certificate Request with Keytool 9
Signing a Certificate Request and Downloading the Certificate 10
Adding the CA and Client Certificates to the Java Keystore 11
Updating the Properties File 11

CHAPTER 3 SAMPLE APPLICATION 12

Extracting SafeNet Integration 12
Using Sample Application 12
Using SafeNetKmipGoogleDriveSample 13
Creating Encryption Keys 13
Uploading Files to Google Drive 13
Downloading Files from Google Drive 14
Use Cases 14
Uploading/Downloading Files without KeySecure 14
Uploading/Downloading Files Using KeySecure 15

Chapter 1

Google Drive

This chapter presents an overview of Google Drive; steps to create a Google APIs console project; steps to enable Drive API; and how the SafeNet KMIP integration with Google Drive is implemented.

Introduction

Google Drive is an Internet-based service that allows users to store, create, and share data in Google's cloud from any computer, tablet, or phone. Any type of files (documents, pictures, videos, and presentations etc.) can be accessed from anywhere.

Google Drive provides functionality to control access to files. Specific files can be shared with and edited together by authorized people. With Google Drive, no need to worry about the size of e-mail attachments. Simply store files of any size on Google Drive and share them with authorized people.

To use Google Drive, users need to activate Google Drive, upload data, and authorize requests. On Google Drive, the data is stored in plaintext.

Prerequisites

Before implementing SafeNet integration, make sure to activate Google Drive. Google Drive can be enabled on a project-by-project basis.

Enabling Google Drive requires a Google account and a Google APIs console project. At least one project must exist before Google Drive can be activated.

Creating Google APIs Console Project

A Google account is needed to create a Google APIs console project.

To create a Google APIs console:

1. Log on to the **Google APIs Console** (https://code.google.com/apis/console/). If no project already exists, a prompt appears to create a project.
2. Click **Create project...** A new project gets created.
3. Click the **Overview** tab. The Dashboard page displays Project Summary which includes project name, project number, a link to register project ID, and the owner name.
4. Click **Register...** next to Project ID. The Register Project ID dialog box appears.
5. In the **Project ID** field, enter an ID for the project.
6. Click **Check availability**.

7 Click **Choose this ID** if the entered ID is available. The selected project ID now appears next to Project ID under Project Summary.

Enabling Google Drive

After creating a project, Google Drive needs to be enabled.

To enable Google Drive:

1 Log on to the **Google APIs Console** (if needed). In case of multiple projects, enter the project to enable Google Drive for.

2 Click the **Services** tab.

3 In the **All Services** table, click **Off** next to **Drive API**.

4 Accept Google APIs terms of service. Drive API is now enabled.

Authorizing Requests with OAuth 2.0

Any requests to access private data must be authorized by an authenticated user who has access to that data.

To authorize private data requests:

1 Log on to the **Google APIs Console** (if needed). In case of multiple projects, enter the project to authorize requests for.

2 Click the **API Access** tab.

3 Click **Create an OAuth 2.0 client ID...** The Create Client ID dialog box appears.

4 Specify branding information including product name, product logo, URL to home page.

5 Click **Next**.

6 Select **Installed application** under **Application type**.

7 Select **Other** under **Installed application type**.

8 Click **Create client ID**. The entered branding information and generated Google client secrets (Client ID and Client secret) appear on the screen.

9 Click **Download JSON**. The downloaded `client_secrets.json` file is required while using the sample application. This file contains Google client secrets used to retrieve access token from Google. The retrieved access token is stored in the `credential.json` file in the working directory. The access token is used for Google authentication.

If needed, another client ID can be created by clicking **Create another client ID...**

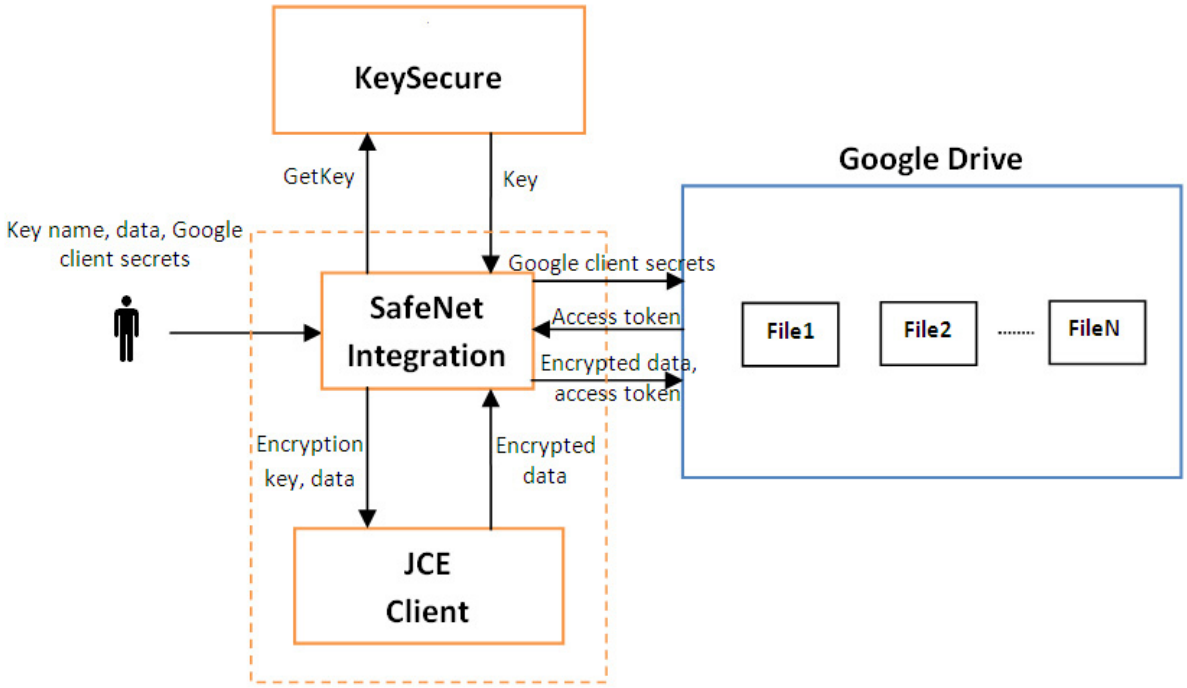
SafeNet Integration

SafeNet integrates Google Drive with the KeySecure server for key management services. With SafeNet integration, users can manage encryption keys on a KeySecure server. Encryption keys can be generated, managed, and stored on the KeySecure server.

Configuring SafeNet Integration

There are some configuration steps needed before using SafeNet integration. Certain parameters in the `IngrianNAE.properties` (also known as properties file) file need to be modified to define how the provider interacts with KeySecure. SSL setup is also needed for secure communication with KeySecure. For details on SSL setup, refer to Chapter 2, “Setting up SSL”.

Uploading Data Using Keys Stored on KeySecure



With SafeNet integration, the encryption process is as follows:

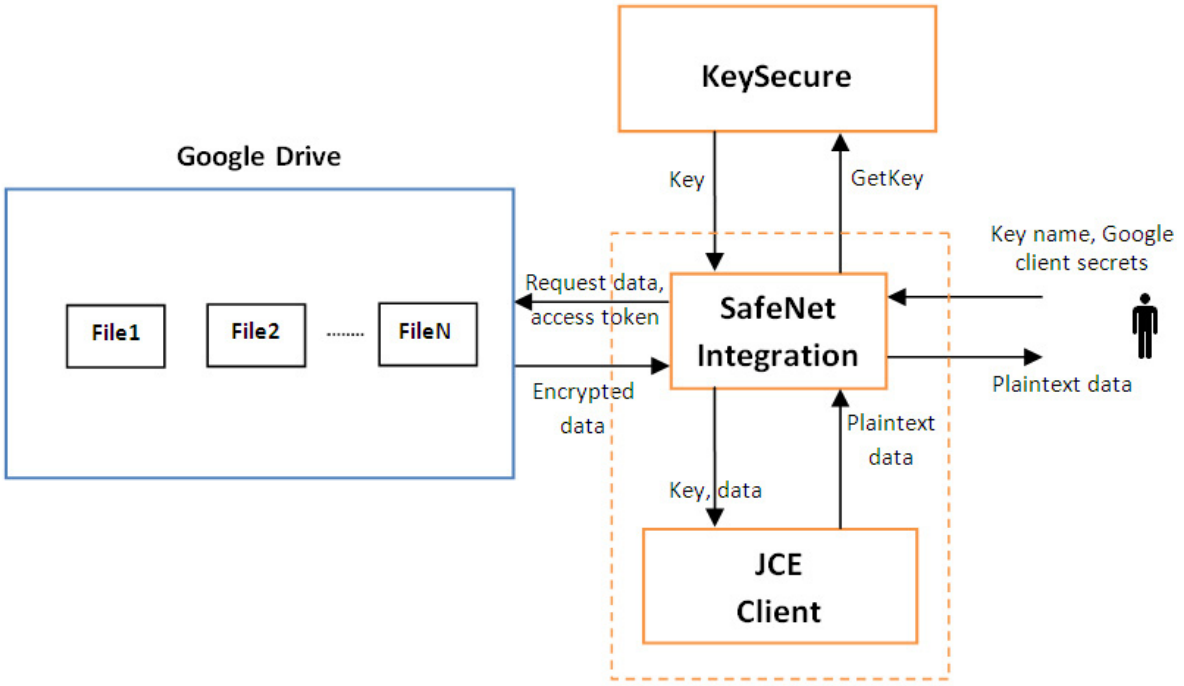
1. User provides the key name, data, and Google client secrets to SafeNet integration.
2. SafeNet integration sends Google client secrets to retrieve access token from Google.
3. SafeNet integration connects to KeySecure, locates the key name on KeySecure, and fetches the key.
4. SafeNet integration sends the data and the encryption key to the JCE client.

5 JCE client encrypts the data using the encryption key.

6 JCE client sends the encrypted data to SafeNet integration.

7 SafeNet integration sends the encrypted data and access token to Google Drive.

Downloading Data Using Keys Stored on KeySecure



With SafeNet integration, the decryption process is as follows:

1 User provides the key name and Google client secrets to SafeNet integration.

2 SafeNet integration connects to KeySecure, locates the key name on KeySecure, and fetches the key.

3 SafeNet integration requests the data from Google Drive using the already retrieved access token.

4 SafeNet integration retrieves encrypted data from Google Drive.

5 SafeNet integration sends the encrypted data and the key retrieved from KeySecure to the JCE client.

6 JCE client decrypts the encrypted data using the key received from SafeNet integration.

7 JCE client sends the decrypted data to SafeNet integration.

8 SafeNet integration sends the decrypted data to the user.

Chapter 2

Setting up SSL

This chapter provides an overview of SafeNet SSL and SSL with Client Certificate Authentication features.

SSL Configuration Procedures

This section describes the procedures you will follow when configuring SSL. It explains the following processes:

1. Creating a Local CA
2. Creating a Server Certificate Request on the Management Console
3. Signing a Server Certificate Request with a Local CA
4. Downloading the Local CA Certificate
5. Adding the CA Certificate to the Java Keystore

Creating a Local CA

To create a local CA:

1. Log on to the Management Console as an administrator with Certificate Authorities access control.
2. Navigate to the Create Local Certificate Authority section on the Certificate and CA Configuration page (Security, Local CAs).
3. Modify the fields as needed.
4. Select either *Self-signed Root CA* or *Intermediate CA Request* as the **Certificate Authority Type**.
5. Click **Create**.

Note: Only a local CA can sign certificate requests on KeySecure. If you are using a CA that does not reside on KeySecure, you *cannot* use the Management Console to sign certificate requests.

Important! Local CA certificates must be added to a trusted CA list to be recognized by the Cryptographic Key Server. Local CA certificates should be backed up for protection.

To add a local CA to a trusted CA list:

1. Navigate to the Certificate and CA Configuration page. (Security, Trusted CA Lists).
2. Select **Default** under Profile Name.
3. Click **Edit** under the Trusted Certificate Authority List section.
4. Select the local CA in the **Available CAs** list.

5 Click **Add**. The local CA now appears in the **Trusted CAs** list.

6 Click **Save**.

Creating a Server Certificate Request on the Management Console

To create a server certificate request on the Management Console:

1 Log on to the Management Console as an administrator with Certificates access control.

2 Navigate to the Create Certificate Request section of the Certificate and CA Configuration page (Security, SSL Certificates) and modify the fields as needed.

3 Click **Create Certificate Request**. This creates the certificate request and places it in the Certificate List section of the Certificate and CA Configuration page. The new entry shows that the **Certificate Purpose** is *Certificate Request* and that the **Certificate Status** is *Request Pending*.

Signing a Server Certificate Request with a Local CA

To sign a server certificate request with a local CA:

1 Log on to the Management Console as an administrator with Certificates and Certificate Authorities access control.

2 Navigate to the Certificate List section on the Certificate and CA Configuration page (Security, SSL Certificates).

3 Select the certificate request and click **Properties**.

4 Copy the text of the certificate request. The copied text must include the header (-----BEGIN CERTIFICATE REQUEST-----) and footer (-----END CERTIFICATE REQUEST-----).

5 Navigate to the Local Certificate Authority List (Security, Local CAs). Select the local CA and click **Sign Request** to access the Sign Certificate Request section.

6 Modify the fields as shown:

- **Sign with Certificate Authority** - Select the CA that signs the request.
- **Certificate Purpose** - Select *Server*.
- **Certificate Duration (days)** - Enter the life span of the certificate.
- **Certificate Request** - Paste all text from the certificate request, including the header and footer.

7 Click **Sign Request**. This will take you to the CA Certificate Information section.

8 Copy the actual certificate. The copied text must include the header (-----BEGIN CERTIFICATE-----) and footer (-----END CERTIFICATE-----).

9 Navigate back to the Certificate List section (Security, SSL Certificates). Select your certificate request and click **Properties**.

10 Click **Install Certificate**.

11 Paste the actual certificate in the Certificate Response text box. Click **Save**. The Management Console returns you to the Certificate List section. The section will now show that the **Certificate Purpose** is *Server* and that the **Certificate Status** is *Active*.

The certificate can now be used as the server certificate for the NAE Server.

Downloading the Local CA Certificate

To download a local CA certificate from KeySecure:

1 Log on to the Management Console as an administrator with Certificate Authorities access control.

2 Navigate to the Local Certificate Authority List section of the Certificates and CA Configuration page (Security, Local CAs).

3 Select the Local CA and click the **Download** button to download the file to your client.

Adding the CA Certificate to the Java Keystore

To add the CA certificate to the keystore:

1 Open a command prompt window on the client and navigate to the Java security directory. This is typically `<JRE-Home>/lib/security`.

2 Use the keytool utility to import the CA certificate by issuing the command below. This statement selects `cacerts` as the keystore. You create an alias for the CA at this time.

```
keytool -import -keystore cacerts -alias <CAAlias> -file <CertFileName.crt>
```

Here, `<CertFileName.crt>` represents the name of the certificate file with the path.

3 Enter the keystore password when prompted. By default, the password is “changeit”.

4 Indicate that the CA is trusted, when prompted.

The utility will then issue a message confirming that the certificate has been added to the keystore. For information about the keytool utility, please refer to Sun’s documentation at:
http://java.sun.com/j2se/1.4.2/docs/tooldocs/solaris/keytool.html

SSL with Client Certificate Authentication Procedures

This section describes the procedures you will follow when configuring SSL with Client Certificate Authentication. It explains the following processes:

1 Generating a Client Certificate Request with Keytool

2 Signing a Certificate Request and Downloading the Certificate

3 Adding the CA and Client Certificates to the Java Keystore

Generating a Client Certificate Request with Keytool

Note: You cannot authenticate the client IP address if you use keytool to generate the client certificate request.

To generate a client certificate request:

1. Open a command prompt window on your client and navigate to the Java security directory (`<Java_Home>\lib\security`).
2. Generate a public/private key pair by issuing the command below. You create an alias for the key pair at this time.

   ```
   keytool -keystore <KeystoreName> -genkey -alias <KeyPairAlias> -keyalg RSA
   ```

   The key generation process will then request the following data:

   - A keystore password.
   - The distinguished name.

     This is a series of fields whose values are incorporated into the certificate request. These fields include country name, state or province name, city or locality name, organization name, organizational unit name, and the users first and last name.

     Important! The user name specified here must exist on KeySecure. If the user does not exist, create it on the User & Group Configuration page. (Security, Local Authentication, Local Users & Groups)
   - The key password.

     The certificate password *must* be the same as the keystore password. You can simply press Return to set the password. You need not retype the keystore password.

     The sample output looks similar to the following:

     ```
     C:\Program Files\Java\jdk1.7.0_07\jre\lib\security>keytool -keystore ca-
     certs -genkey -alias KeyPairAlias1 -keyalg RSA
     Enter keystore password:
     What is your first and last name?
       [Unknown]:  AB
     What is the name of your organizational unit?
       [Unknown]:  ENGG
     What is the name of your organization?
       [Unknown]:  SFNT
     What is the name of your City or Locality?
       [Unknown]:  NOIDA
     What is the name of your State or Province?
       [Unknown]:  UP
     ```

```
What is the two-letter country code for this unit?
  [Unknown]:  IN
Is CN=AB, OU=ENGG, O=SFNT, L=NOIDA, ST=UP, C=IN correct?
  [no]:  yes

Enter key password for <KeyPairAlias1>
        (RETURN if same as keystore password):
```

Important! The user name, **AB**, specified above must exist on KeySecure. If the user does not exist, create it on the User & Group Configuration page. (Security, Local Authentication, Local Users & Groups)

3 Create the certificate request by issuing the command below. Reference the Key Pair Alias you created above.

```
keytool -certreq -alias <KeyPairAlias> -file <CertReqFileName>
-keystore <KeystoreName>
```

You will now have a certificate request in the `<CertReqFileName>` file.

Signing a Certificate Request and Downloading the Certificate

This section describes how to sign a certificate request with a local CA and then download the certificate. You must download the certificate *immediately* after it is signed by the CA.

To sign a certificate request with a local CA:

1 Open the certificate request in a text editor.

2 Copy the text of the certificate request. The copied text must include the header (-----BEGIN CERTIFICATE REQUEST-----) and the footer (-----END CERTIFICATE REQUEST-----).

3 Log on to KeySecure as an administrator with Certificate Authorities access control.

4 Navigate to the Local Certificate Authority List (Security, Local CAs). Select the local CA and click **Sign Request** to access the Sign Certificate Request section.

5 Modify the fields as shown:

- **Sign with Certificate Authority** – Select the CA that signs the request.
- **Certificate Purpose** – Select *Client*.
- **Certificate Duration (days)** – Enter the life span of the certificate.
- **Certificate Request** – Paste all text from the request, including the header and footer.

6 Click **Sign Request**. This will take you to the CA Certificate Information section.

7 Click the **Download** button to save the certificate on your local machine.

Adding the CA and Client Certificates to the Java Keystore

To add the client certificate to the Java keystore:

1. Open a command prompt window on your client and navigate to the Java security directory (`<Java_Home>\lib\security`).
2. Import the CA certificate that signed the client certificate using the command below. You create an alias for the CA at this time. When prompted, enter the keystore password and indicate that the CA is trusted.

   ```
   keytool -keystore <KeystoreName> -import -alias <CAAlias> -file
   <CertFileName.crt>
   ```

   The above step is required if a CA other than the Local CA is to be used.
3. Import the signed client certificate using the following command. Use the key pair alias you used to create the certificate request. When prompted, enter the keystore password.

   ```
   keytool -keystore <KeystoreName> -alias <KeyPairAlias> -import -file
   <CertFileName.crt>
   ```
4. Verify that the client certificate was properly imported by executing the following command. Reference the key pair alias you used above. The system should display the certificate.

   ```
   keytool -keystore <KeystoreName> -alias <KeyPairAlias> -list -v
   ```

Important! To enable Client Certificate Authentication, your keystore must have a copy of the CA certificate that signed the server certificate.

Updating the Properties File

After setting up the SSL, the SSL-related parameters must be updated in the properties file. The following parameters in the `IngrianNAE.properties` file need to be updated as follows:

- `NAE_IP.1=<IP address of the server.>`
- `KMIP_Port=<KMIP SSL port on the server.>`
- `Protocol=ssl`
- `Key_Store_Location=<path and name of keystore that contains a copy of the server's local CA, the client certificate, and the CA that signed the client certificate.>`
- `Key_Store_Password=<keystore password>`
- `Client_Cert_Alias=<client certificate alias>`
- `Client_Cert_Passphrase=<client certificate password, if used>`

Chapter 3

Sample Application

A sample application, `SafeNetKmipGoogleDriveSample`, is included to demonstrate the data upload/download with and without SafeNet key management.

Extracting SafeNet Integration

Extracting the `SafeNetKMIPGDrive.zip` file creates a `SafeNetKMIPGDrive` folder. This folder contains the following items:

- `src` – Source file for the sample application.
- `bin` – Package and class file generated by compiling the source file.
- `lib` – Library (jar) dependencies, Google API, SafeNet KMIP library (jar), and the properties file.
- `doc` – JavaDoc html and resource files.
- `UserGuide` – SafeNet KMIP and Google Drive Integration Guide.
- `readme.txt` – Instructions to implement SafeNet integration.

Using Sample Application

1. Add the `lib` and `bin` folder to the `CLASSPATH` environment variable.

   For example, set `CLASSPATH=%CLASSPATH%;"D:\SafeNetKMIPGDrive\lib\*;D:\SafeNetKMIPGDrive\bin";`

2. Set the properties file `IngrianNAE.properties`. For details, refer to “Updating the Properties File” on page 11.

3. Copy the downloaded `client_secrets.json` file to the working directory.

4. Check the command usage for available options.

   a. Navigate to the path where SafeNet integration is extracted.

   b. Execute `java SafeNetKmipGoogleDriveSample`. The command usage is displayed:

```
D:\SafeNetKMIPGDrive\bin>java SafeNetKmipGoogleDriveSample
Usage:

-createENCKey <KeyName> <KeySize>
-uploadFile <filename> <ENCKey>*
```

```
-downloadFile <filename> <destpath> <ENCKey>*


*ENCKey should be provided in case of encrypting data before uploading to
Google Drive
```

5 Use the sample `SafeNetKmipGoogleDriveSample` with required parameters shown in the command usage above.

Using SafeNetKmipGoogleDriveSample

Use `SafeNetKmipGoogleDriveSample` to upload and download data from Google Drive with or without SafeNet KeySecure. Symmetric keys can also be created on KeySecure using sample application `SafeNetKmipGoogleDriveSample`.

The following topics describe the tasks that can be performed using the sample application `SafeNetKmipGoogleDriveSample`.

Important! Performing operations on Google Drive requires access token for authentication. While using the sample application, the application may prompt to retrieve access token from Google (if the access token does not already exist). Allow the application to retrieve the access token. The retrieved access token is stored in the `credential.json` file downloaded at the working directory. The same access token will be used for all future authentications.

Creating Encryption Keys

To create an encryption key, execute:

```
java SafeNetKmipGoogleDriveSample -createENCKey <KeyName> <KeySize>
```

Where:

- `<KeyName>`: Name for the symmetric key.
- `<KeySize>`: Size for the symmetric key.

Uploading Files to Google Drive

To upload a file to Google Drive, execute:

```
java SafeNetKmipGoogleDriveSample -uploadFile <filename> <ENCKey>*
```

Where:

- `<filename>` – Name and path of the file to be uploaded.
- `<ENCKey>` – Name of the key for encryption.

Downloading Files from Google Drive

To download a file from Google Drive, execute:

```
java SafeNetKmipGoogleDriveSample -downloadFile <filename> <destpath> <ENCKey>*
```

Where:

- `<filename>` – Name of the file to be downloaded.
- `<destpath>` – Path to store the downloaded file.
- `<ENCKey>` – Name of the key used for encryption.

Use Cases

This section describes how files are uploaded to and downloaded from Google Drive in various real life scenarios.

Uploading/Downloading Files without KeySecure

This use case describes how the files are uploaded to and downloaded from Google Drive without using SafeNet integration.

1. Uploading Files to Google Drive without KeySecure

This use case describes how files are uploaded to Google Drive without using SafeNet integration.

To upload a file to Google Drive, execute:

```
java SafeNetKmipGoogleDriveSample -uploadFile <filename>
```

Where:

- `<filename>` – Name and path of the file to be uploaded.

2. Downloading Files from Google Drive without KeySecure

This use case describes how the files are downloaded from Google Drive without using SafeNet integration.

To download a file from Google Drive, execute:

```
java SafeNetKmipGoogleDriveSample -downloadFile <filename> <destpath>
```

Where:

- `<filename>` – Name of the file to be downloaded.
- `<destpath>` – Path to store the downloaded file.

Uploading/Downloading Files Using KeySecure

This use case describes how the files are uploaded to and downloaded from Google Drive using SafeNet integration when the key is created on KeySecure.

1. Creating Symmetric Key on KeySecure

To create a symmetric key on KeySecure using SafeNet integration, execute the following command:

```
java SafeNetKmipGoogleDriveSample -createENCKey <KeyName> <KeySize>
```

Where:

- `<KeyName>` – Name for the symmetric key.
- `<KeySize>` – Size for the symmetric key.

2. Uploading Files to Google Drive Using KeySecure

This use case describes how the files are uploaded to Google Drive using SafeNet integration.

To upload a file to Google Drive, execute:

```
java SafeNetKmipGoogleDriveSample -uploadFile <filename> <ENCKey>*
```

Where:

- `<filename>` – Name and path of the file to be uploaded.
- `<ENCKey>` – Name of the key for encryption.

Note: *`ENCKey` is needed to encrypt the file before uploading it to Google Drive.

3. Downloading Files from Google Drive Using KeySecure

This use case describes how the files are downloaded from Google Drive using SafeNet integration.

To download a file from Google Drive, execute:

```
java SafeNetKmipGoogleDriveSample -downloadFile <filename> <destpath> <ENCKey>*
```

Where:

- `<filename>` – Name of the file to be downloaded.
- `<destpath>` – Path to store the downloaded file.
- `<ENCKey>` – Name of the key used for encryption.

Note: *`ENCKey` is needed to decrypt the file before downloading it from Google Drive.