GUIDE

SDS ENCRYPTION SERVICE FOR GOOGLE WORKSPACE

ADMINISTRATION GUIDE

Version 4.4.0

Document last updated: December 30, 2024

Reference: sds-en-sds-for-gw-administration_guide-v4.4

Table of contents

In the documentation, SDS encryption service for Google Workspace is referred to in its short form: SDS encryption service for Google Workspace.

This document is not exhaustive and minor changes may have been included in this version.

1. Getting started

The SDS encryption service for Google Workspace is a solution in which corporate data managed in the Google Workspace ecosystem can be protected, edited and consulted. Google Workspace is Google's cloud-based application suite for professionals. For more information, refer to the Google Workspace documentation.

The SDS encryption service for Google Workspace relies on Google Client Side Encryption (CSE), the end-to-end encryption method that Google offers for its Google Workspace applications. CSE is configured in the Google administration console. This technology is available only on Chrome or Microsoft Edge (Chromium) browsers. For more information on supported browsers, refer to the Google Client Side Encryption documentation, section *Browser requirements*.

Google generates DEKs (Data Encryption Keys) to encrypt files. Before such keys are stored on Google servers, the SDS encryption service for Google Workspace wraps them using KEKs (Key Encryption Keys).

The SDS encryption service for Google Workspace is installed in your on-premise or cloud-based infrastructure; KEKs are therefore stored with you and never sent to Google servers.

Before performing cryptographic operations, the SDS encryption service for Google Workspace first conducts a double verification:

- Authentication: checks the identity of the user requesting the operation,
- Authorization: checks the user's access privileges for the file to encrypt/decrypt.

The SDS encryption service for Google Workspace generates logs for all the operations that it performs.

NOTE
The use of the solution in any way other than as described in the documentation is not managed. Alternatively, get in touch with Stormshield Support for clarification.

2. Understanding the deployment procedure

2.1 Requirements

- The server on which the SDS encryption service for Google Workspace is installed must be healthy. There must be an information system security policy whose requirements are met on the servers. This policy shall verify the installed software is regularly updated and the system is protected against viruses and spyware or malware (firewall properly configured, antivirus updates, etc.). It is imperative to follow the operating system security recommendations issued by the ANSSI in their document *ANSSII-BP-028-EN*.
- Access to the administrative functions of the workstation system is restricted only to system administrators.
- The operating system must manage the logs generated by the product in accordance with the security policy of the company. It must for example restrict read access to these logs to only those explicitly permitted. For more information, see the section Logging requirements.
- You must set up a system upstream of the SDS encryption service for Google Workspace to protect against distributed denial-of-service (DDoS) and brute-force attacks. Please follow the ANSSI recommendations (French only).
- You must filter incoming requests upstream of the SDS encryption service for Google Workspace. Only requests meeting the following conditions should be accepted:
  - The request header size must be smaller than the NodeJS default value. See the NodeJS documentation.
  - The size of the request body must be less than 1 MB.
- The SDS encryption service for Google Workspace must be installed on a server whose system and OpenSource contributions are kept up to date.
- The server hosting the solution must be located in a secure physical environment with access control protocols and must be trusted.

2.2 Recommendations on administrators

- The SDS encryption service for Google Workspace's administrators are considered as trusted. They are responsible for defining the SDS encryption service for Google Workspace security policy by respecting the state of the art.
- The system administrator responsible is also considered as trusted. He/She is responsible for the installation and maintenance of the application and server. He/She applies the security policy defined by the SDS encryption service for Google Workspace administrators.

2.3 Recommendations on network rules

The content of the requests processed by the SDS encryption service for Google Workspace is in JSON format only. You can add the following rules to your web firewall (WAF) or load balancer to ensure optimum protection:

- All HTTP requests are blocked except:
  - The POST requests with a *Content-Type* header including "application/json".
  - The GET and OPTIONS requests without a *Content-Type* header or with a *Content-Type* header including "application/json".

2.4 Deploying the SDS encryption service for Google Workspace in your infrastructure

The table below lists the various steps involved in deploying SDS encryption service for Google Workspace.

Click on a link to open the corresponding procedure in the SDS encryption service for Google Workspace guide.

Steps	Description
1	Configuring the identity provider and Google Workspace
2	Setting up the infrastructure and install the SDS encryption service for Google Workspace
3	Configuring the config.json file
4	Configuring Key Encryption Keys (KEKs) • Configuring keys in KMS mode - or - • Configuring the keks.json file
5	Setting up a network configuration that can be reached via Google Workspace.
6	Running the SDS encryption service for Google Workspace
7	Checking system health
8	Configuring load balancing
9	(Optional) Using the SDS encryption service for Google Workspace in HTTPS and Configuring TLS ciphers
10	Setting up logging
11	(Optional) Customizing the authorization rules by applying an OPA policy

2.5 Adding the SDS encryption service for Google Workspace in Google Workspace

In the table below are the various steps involved in adding SDS encryption service for Google Workspace in Google Workspace. Click on a link to open the corresponding procedure in the SDS encryption service for Google Workspace guide or in Google help.

Steps	Description
1	Setting up your external key service
2	Connecting Google Workspace to the external key service
3	Connecting Google Workspace to the Identity provider
4	Indicating the well-known file in the config.json file
5	Enabling the service for users: Drive, Meet, Calendar, Gmail

3. Configuring the identity provider and Google Workspace

Before installing the SDS encryption service for Google Workspace, you must prepare the environment by configuring the identity provider and Google Workspace.

3.1 Configuring the identity provider

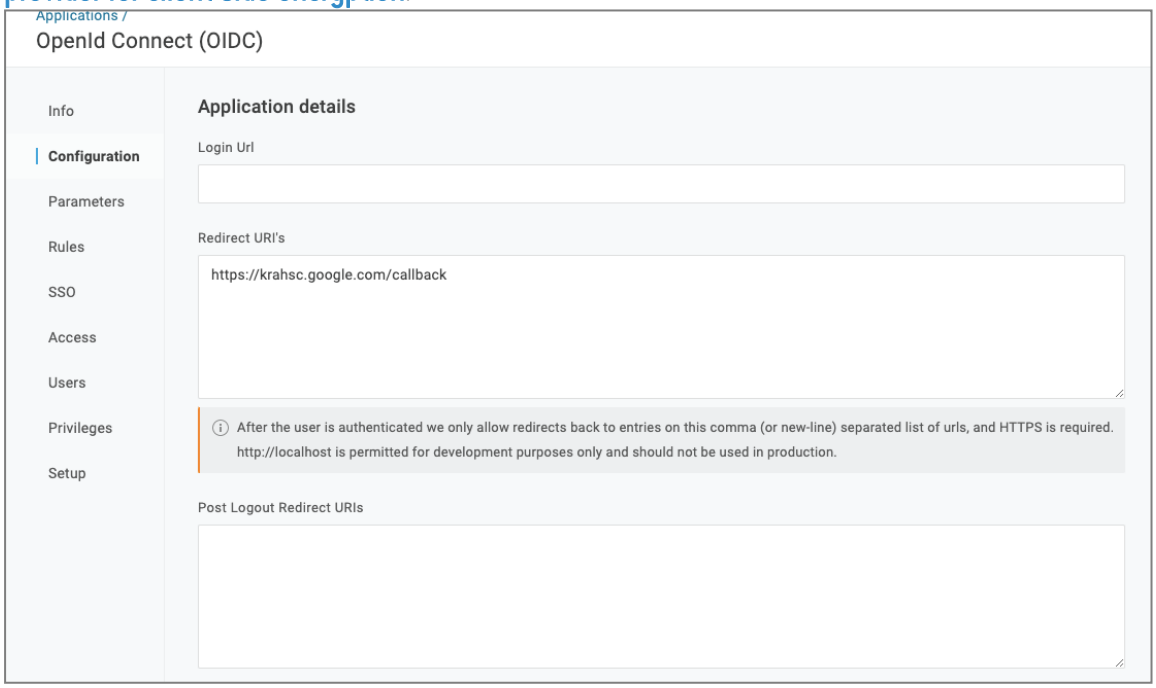
The SDS encryption service for Google Workspace uses an identity provider (IDP) to authenticate end users, manage their access permissions and their life cycles. Configure the provider of your choice and create an OpenID Connect application.

The SDS encryption service for Google Workspace is compatible with JWT tokens signed with the RS256 algorithm.

The procedure below describes the configuration with One Login. For more information, refer to the One Login documentation

3.1.1 Specifying the redirect URL

In OpenID Connect, select the **Configuration** menu, then specify the redirect URI in the **Redirect URI's** field. For more information, refer to the Google documentation Connect to your identity provider for client-side encryption.



3.1.2 Retrieving import values

In OpenID Connect, select the **SSO** menu and take note of the **ClientID** and **Issuer URL** import values:

These values will be used in the *config.json* file, in the **tenants > user_authentication > idps** section of the SDS encryption service for Google Workspace. Below are a few examples:

- **ClientID**: `3e14f1a0-5814-0550-cy6e-0bd6abe5ty43540000`
- **Issuer URL** (*Well-known* configuration): https://stormshield-example.onelogin.com/oidc/2/.well-known/openid-configuration

For more information on declaring the identity providers in the SDS encryption service for Google Workspace, refer to section Configure the SDS encryption service for Google Workspace, tenants parameter.

3.1.3 Managing authentication tokens

According to the specifications provided by Google, the authentication token contains a Json Web Token (i.e., JWT). For more information, see the RFC7516 document.

The mandatory and optional fields expected by the KACLS depending on the routes used are listed in the following table:

Routes	Mandatory fields	Optional fields
• wrap • unwrap • privilegedwrap, • privilegedunwrap • privatekeydecrypt, • privatekeysign, • privilegedprivatekeydecrypt • privilegedprivatekeysign	• iss • aud • exp • iat • email	• google_email
Authentication token to the KACLS: • privilegedunwrap They are used to authenticate a KACLS to another one in the context of a migration.	• iss • aud • exp • iat • kacls_url • resource_name	
delegate	• iss • aud • exp • iat • email • delegated_to • resource_name	google_email
wrapprivatekey	• iss • aud • exp • iat	

For more information, refer to the Google documentation.

Authentication tokens related to delegation

The authentication tokens used by the encryption and decryption operations (i.e., wrap and unwrap routes) in the context of a delegation operation are dynamically generated by the SDS encryption service for Google Workspace (delegate route): for security reasons, and as recommended by Google, these tokens have a lifetime of 15 minutes.

3.2 Configuring Google Workspace

You must indicate the URL of the external key service and the identity provider in the Google Workspace administration console.

For more information, refer to the Google documentation Use client-side encryption for users' data.

3.2.1 Specifying the External key service

External key service is the section in the Google Workspace administration console in which you specify information for the SDS encryption service for Google Workspace.

With the SDS encryption service for Google Workspace, several external key services can be used in your Google Workspace tenant's administration console. For example, if you want separate services for each distinct organizational unit (OU) in your organization for various Google applications (Meet, Drive, ...).

In standalone mode, you must enter a UUID for every tenant installed so that their associated KEKs will be available. For further information, refer to the section Adding KEKs to the file.

If you are using a Key Management System (KMS), the tenant's UUID is included in the attributes of the KEK. For more information, refer to the section Configuring KEKs in KMS mode.

With the SDS encryption service for Google Workspace, several tenants can be used on the same instance of the encryption service. For example, if your organization has several domains, you can manage each tenant independently for each domain.

An external key service must be specified for each tenant.

- The **Name** of the external key service can be shown in error messages that the end user will see.
- The **URL** of the external key service consists of the following:

Address of the SDS encryption service for Google Workspace instance that you are installing	E.g., https://cse.example.com/api/v1
Tenant UUID	E.g., a4670b0-4bc11-4290-a5bd-498c2e1fb0bf You must generate a v4 UUID to identify tenants, even when there is only one on your instance.

EXAMPLE
https://cse.example.com/api/v1/a4670b0-4bc11-4290-a5bd-498c2e1fb0b

Google applications will use this URL, so it must be a public address.

3.2.2 Specifying the identity provider (IDP)

For more information, refer to the Google documentation Connect to your IdP for CSE.

4. Installing the SDS encryption service for Google Workspace

There are two ways to deploy the SDS encryption service for Google Workspace: via an RPM for RedHat systems, or via a Docker image.

4.1 RPM versus Docker

4.1.1 RPM version

In the RPM version, the application runs as a system service managed by *systemd* via *systemctl*. It means that the SDS encryption service for Google Workspace:

- Automatically starts with the system,
- Can be started, stopped ans restarted with *systemctl* commands,
- Generates logs that you can access via *journalctl.*

The table below shows the usual commands for RPM installation:

Action	RPM command
Start	systemctl start cse
Stop	systemctl stop cse
Restart	systemctl restart cse
Display logs	journalctl -feu cse

4.1.2 Docker environment

In the Docker environment, Docker manages the container life cycle. It means that the SDS encryption service for Google Workspace:

- Starts when the container is launched,
- Is managed with Docker commands (*docker run*, *docker stop*, etc.),
- Writes the logs to the container standard output *(stdout/stderr*).

The table below shows the usual commands to install the solution via a Docker image:

Action	Docker command
Start	docker run <params> stormshield/kmaas:<version>
Stop	docker stop <containerID>
Restart	docker restart <containerID>
Display logs	docker logs <containerID>

4.2 Installing the SDS encryption service for Google Workspace via RPM

Before installing the SDS encryption service for Google Workspace, you must install the operating system and NodeJS.

4.2.1 Requirements

In a cluster of three servers for the SDS encryption service for Google Workspace, in order to manage an average of 45 requests per second and per Red Hat instance, each server must have at least the following resources:

- 4 processors and one thread per processor
- 4 GB of memory
- 20 GB of storage

If you want to improve performance, add the following resources in this order:

1. Threads for each processor,
2. Processors,
3. Instances in the cluster.

4.2.2 Compatibility

Each supported version of the operating system is compatible with specific versions of OpenSSL and NodeJS. Please check the compatibility in the table below:

Operating system	OpenSSL version	NodeJS version
RedHat Enterprise Linux 8.10	At least v3.2.2	v20 Tested with v20.16.0
RedHat Enterprise Linux 9.0	At least v3.0.1	v20

4.2.3 Installing the operating system

Install and activate a RedHat Enterprise Linux distribution version 8.10 or 9.0 based on the version of the RPM delivered by Stormshield.

For more information, refer to the RedHat 8 documentation or the RedHat 9 documentation.

It is imperative to follow the operating system security recommendations issued by the ANSSI in their document *ANSSII-BP-028-EN*.

You can install all dependencies offline on your operating system. To do so:

1. Get the RPM of the dependency.
2. Copy it on your machine.
3. Install it by running the command:
   `rpm -i`

4.2.4 Installing OpenSSL

- OpenSSL v1.1.1 is supplied by default with RedHat Enterprise Linux 8.10. You must manually install OpenSSL 3 using the commands below. Stormshield recommends using the EPEL repository.

```
# subscription-manager repos --enable codeready-builder-for-rhel-8-
$(arch)-rpms

# dnf install https://dl.fedoraproject.org/pub/epel/epel-release-
latest-8.noarch.rpm

# dnf install openssl3

# ln -b -s /usr/bin/openssl3 /usr/bin/openssl

# ln -b -s /usr/lib64/libssl.so.<openssl_version>
/usr/lib64/libssl.so

# ln -b -s /usr/lib64/libcrypto.so.<openssl_version>
/usr/lib64/libcrypto.so

# ln -b -s /usr/include/openssl3/openssl /usr/include/openssl
```

  where <openssl_version> must be replaced by the OpenSSL version installed, for example *3.0.7*.
- OpenSSL v3.2.7 is supplied by default with RedHat Enterprise Linux 9.0. Install OpenSSL using the following command:
  `# yum install openssl`

4.2.5 Installing NodeJS

1. Install the package using the following command:
   `# dnf module install nodejs:20`
2. Check that NodeJS has indeed been installed with the following command:
   `# node --version`

Ensure that the NodeJS autorun has been enabled.

This command will install the latest version of NodeJS 20. The version 4.4.0 of the SDS encryption service for Google Workspace has been tested with NodeJS 20.12.0.

4.2.6 Installing the SDS encryption service for Google Workspace

To install the SDS encryption service for Google Workspace, you must be a root user of the RedHat system.

Stormshield recommends that the server on which the SDS encryption service for Google Workspace is installed has a multi-core processor with a minimum of 4 cores.

1. Copy the `.rpm` file on the system.
2. Run the following command:
   `# rpm -i <package_name>.rpm`

If NodeJS was not installed beforehand, this error message will appear:

```
error: Failed dependencies : nodejs is needed by csexxx
```

The following folders and files will be installed:

Location	Resource
/usr/lib64/cse	Source file folder. On installation, the owner of the files is the user *stormshield-cse.* He has u=rx,g=,o= permissions. For security reasons, we recommend keeping these default settings.
/usr/bin/cse	Binary file folder
/etc/stormshield/cse	Configuration file folder: • *config.json.template* - template configuration file for the SDS encryption service for Google Workspace. • *keks.json.template* - template file for the list of key encryption keys (KEK). • *policy.wasm* - default security policy module. This module does not enable any security policies. • *policy.data.json* - data file used by the policy.wasm module.
/etc/systemd/system/cse.service	Configuration file to use the SDS encryption service for Google Workspace as a SystemD service
/usr/share/licenses/cse	License file folder
/usr/share/doc/stormshield/cse/copyright	Folder of the license files for the open-source libraries

4.3 Installing the SDS encryption service for Google Workspace via a Docker image

4.3.1 Requirements

- You must follow the ANSSI's recommendations from the ANSSI-FT-082 document, relating to the deployment of Docker containers.
- You must set up a container orchestration environment (e.g., Kubernetes , Docker Swarm) to automatically manage replication, high availability and container life cycle. For a resilient installation, Stormshield recommends a minimum of 3 instances of the SDS encryption service for Google Workspace.
- The configuration of the orchestrator depends on the technology used. Refer to your orchestrator's documentation for detailed installation steps and security best practices specific to your environment.

4.3.2 Knowing the contents of the Docker image archive

The archive of the SDS encryption service for Google Workspace contains the following files:

Location	Resource
stormshield-kmaas-{version}.tar	Docker image of the SDS encryption service for Google Workspace in *.tar* format.

Location	Resource
config.json.template	Template configuration file for the SDS encryption service for Google Workspace.
keks.json.template	Template file for the list of key encryption keys (KEK).
list-of-dependencies.html	List of the dependencies of the SDS encryption service for Google Workspace.
policy.wasm	Default security policy module. This module does not enable any security policies.
policy.data.json	Data file used by the *policy.wasm* module.

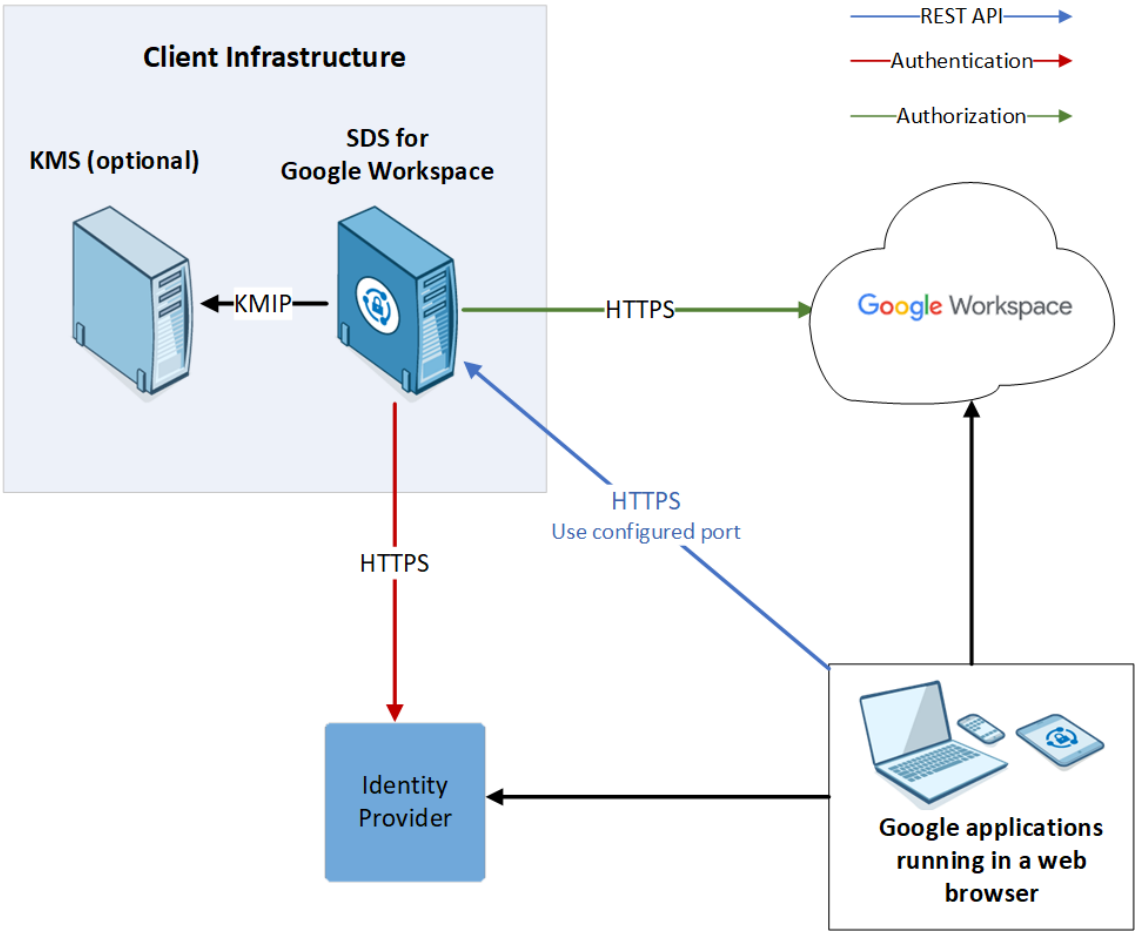
4.3.3 Installing Docker

1. Install Docker on each server where you want to run the SDS encryption service for Google Workspace. The minimum Docker version supported is 20.1.1.
   For more information, refer to the Install Docker Engine documentation.
2. Create a dedicated directory to host your configuration, in which you copy the template files provided with the Docker image. Rename the files as follows:
   - *config.json*: configuration file of the SDS encryption service for Google Workspace,
   - *keks.json*: file containing the list of key encryption keys (KEK).

   In step Configure the SDS encryption service for Google Workspace, you can edit these files directly in this directory.
3. Make sure that the directory containing the configuration files is available in the container through a volume or using your orchestrator's technology.
   For more information, please refer to your orchestrator documentation.
4. Ensure that the *keks.json* file and the private keys are made available in a secure way in the production environment.

5. Configuring the network

The diagram and the table below describe the various streams of incoming and outgoing traffic on the SDS encryption service for Google Workspace server. Configure your network to allow the following connections:



Description	Protocol	Source	Source port	Destination	Destination port
The SDS encryption service for Google Workspace API REST	HTTPS	Google application	*	SDS encryption service for Google Workspace	Depends on the administrator's configuration (*config.json*)
Getting the configuration of OpenID authentication	HTTPS	SDS encryption service for Google Workspace	*	OpenID endpoint	Specified by the configuration of the authentication service (usually 443)
Getting the configuration of the JWKS authorization	HTTPS	SDS encryption service for Google Workspace	*	JWKS endpoint	Specified by the configuration of the authorization service (usually 443)

Description	Protocol	Source	Source port	Destination	Destination port
Getting decryption keys	KMIP version 1.4	SDS for Google Workspace	*	Client's infrastructure	Depends on the administrator's configuration (usually 5696)

NOTE
In Docker deployment mode, you must expose the ports to the containers. For more information, please refer to your orchestrator documentation.

6. Configuring proxy access

If the SDS encryption service for Google Workspace is located behind a proxy in your infrastructure, the service must be configured to enable the use of this proxy. To do so, add the URL of the proxy and any exclusions to the configuration file.

6.1 In RPM mode

1. Run the following command:
   ```
   # systemctl edit cse.service
   ```
   The *override.conf* configuration file is created in the */etc/systemd/system/cse.service.d* directory if it was installed in the default directory.
2. Edit the file and copy the following text containing the environment variable for the proxy's URL:
   ```
   [Service]
   Environment="https_proxy=https://my-proxy.my-domain"
   ```
   Where *https://my-proxy.my-domain* is the URL of the proxy used.
3. If you need to exclude certain endpoints from the proxy, declare them in the same file via the no_proxy environment variable. The possible values for this variable are the following:
   - The * character means that all endpoints are excluded. This is equivalent to disabling the proxy.
   - A domain, for example *domain.com*,
   - A domain suffix, for example *.domain.com*,
   - A v4 or v6 IP address, for example *192.168.1.10* or *2001:67c:2e8:22::c100:68b*,
   - A v4 or v6 IP address in CIDR, for example *172.30.0.0/16* or *2001:67c:2e8:22::c100:68b/128*.

   The different values must be separated by commas.

   EXAMPLE
   Example of a file `cse.service` in which the proxy is configured and different endpoints are excluded from the proxy:
   ```
   [Service]
   Environment="https_proxy=https://my-proxy.my-domain"
   Environment="no_
   proxy=domain.com,192.168.1.10,2001:67c:2e8:22::c100:68b/128"
   ```
4. Reload the systemd service using the following command:
   ```
   # systemctl daemon-reload
   ```
5. Start the systemd service using the following command:
   ```
   # systemctl start cse
   ```
   A startup log indicates that the service is launched in proxy mode. For more information, refer to the section Logs relating to the status of the service (start, stop, failure).

6.2 In Docker mode

In Docker mode

- Declare the following environment variables to configure the proxy:
  - *https_proxy*: defines the proxy URL,
  - *no_proxy*: defines the endpoints excluded from the proxy.

EXAMPLE
Example of a Docker command declaring environment variables:

```
docker run -v /my-kmaas-config-folder:/etc/stormshield/cse -p 443:3000 -
e https_proxy="https://my-proxy.my-domain" -e
no_proxy="domain.com,192.168.1.10,2001:67c:2e8:22::c100:68b/128"
stormshield/kmaas:<version>
```

7. Configure the SDS encryption service for Google Workspace

The global configuration of the SDS encryption service for Google Workspace is managed through a JSON file, *config.json*, which is saved by default in */etc/stormshield/cse*. This file sets the specifications for authentication and authorization, as well as the port, service name and service mode.

7.1 Creating the global configuration file

A template file, *config.json.template*, is available to assist you.

- In RPM mode, this file is located in the directory */etc/stormshield/cse*.
  - Create your own global configuration file from the copy of the template using the following command:
    ```
    # cd /etc/stormshield/cse
    # cp --preserve config.json.template config.json
    ```
- In Docker mode, the file is located in the dedicated directory you have created during installation. For more information, see Installing the SDS encryption service for Google Workspace via a Docker image.

7.2 Assigning access privileges to the file

- Assign the read and write access privileges held by the current user to the file and read access to the current *config.json* group:
  ```
  # chmod u=rw,g=r,o= config.json
  ```

Do not assign any run privileges on these files, or any privileges to other users. If access privileges are too permissive, a warning log will be generated when the SDS encryption service for Google Workspace starts, but will not prevent it from launching.

During installation, the *stormshield-cse* user is the owner of the configuration files by default. Do not change the owner.

7.3 Editing the global configuration file

- Change the default values of the *config.json* file:

```
{
  "tenants": [
  {
    "tenant_id": "_TENANT_ID_",

    "user_authentication": {
         "enable_wellknown_cse_discovery": "_ENABLE_WELLKNOWN_CSE_DISCOVERY_",
       "idps": [
          {
               "discovery_uri": "_AUTHENTICATION_OPEN_ID_CONFIGURATION_URL_",
               "client_id": "_AUTHENTICATION_AUDIENCE_"
          },
          {
               "jwks_uri": "_IDPS_JWKS_URL_",
               "audience": "_IDPS_AUDIENCE_",
               "issuer": "_IDPS_ISSUER_"
          }
```

```
        ]
      },
      "admin_authentication": [
         {
               "discovery_uri": "_ADMIN_AUTHENTICATION_DISCOVERY_URI_",
               "client_id": "_ADMIN_AUTHENTICATION_ISSUER_"
         }
      ],
      "wrapprivatekey_authentication" : [
         {
               "discovery_uri": "_WRAPPRIVATEKEY_AUTHENTICATION_DISCOVERY_URI_",
               "client_id": "_WRAPPRIVATEKEY_AUTHENTICATION_ISSUER_"
         }
      ],
      "migration" : {
         "enabled": "_ENABLED_",
         "kaclstokacls_token": {
              "kid": "_KACLS_TO_KACLS_KEY_",
              "format": "_FORMAT_",
              "key": "_KEY_",
              "duration": "_DURATION_"
         },
         "acls": {
              "kacls_urls": [
                  "_ALLOWED_KACLS_URL_"
          ]
         }
      },
      "crypto_backends": [
      {
         "id": "_CRYPTO_BACKEND_ID_",
         "name": "_CRYPTO_BACKEND_NAME_",
         "type": "_CRYPTO_BACKEND_TYPE_",
         "configuration": {
               "host": "_HOST_",
               "model": "_MODEL_",
               "vendor": "_VENDOR_",
               "port": _PORT_,
               "credentials": {
                 "ca": "_CA_",
                 "cert": "_CERT_",
                 "key": "_KEY_"
               }
         }
      }
      ],
      "keys": {
        "users_private_keys": {
                "crypto_backend": {
                   "id": "_CRYPTO_BACKEND_ID_"
                }
        }
      },
      "kmaas": {
        "user_authentication": [
                {
                      "discovery_uri": "_KMAAS_AUTHENTICATION_DISCOVERY_URI_",
                      "client_id": "_KMAAS_AUTHENTICATION_CLIENTID_"
                }
        ]
      },
      "delegate": {
        "authentication": {
          "key": "_KEY_",
          "enable": "_ENABLE_"
        }
      }
},
```

```
],
"authorization": [
  {
        "issuer": "_AUTHORIZATION_ISSUER_",
        "url": "_AUTHORIZATION_JWKS_URL_",
        "audience": "_AUTHORIZATION_AUDIENCE_"
  }
],
"https": {
  "credentials": {
        "key": "_CREDENTIALS_KEY_",
        "cert": "_CREDENTIALS_CERT_",
        "ca": "_CREDENTIALS_CA_"
  }
},
"kacls_url": "_KACLS_URL_",
"port": "_PORT_NUMBER_",
"name": "_SERVER_NAME_",
"persistence_type": "_PERSISTENCE_TYPE_",
"kacls_kek_label": "_KACLS_KEK_LABEL_",
"keks": {

  "auto_refresh": {
        "scheduled": {
          "interval_seconds": "_SCHEDULED_AUTO_REFRESH_INTERVAL_SECONDS_"
        },
        "minimum_interval_seconds": "_AUTO_REFRESH_MINIMUM_INTERVAL_SECONDS_"
  }
},

"kmip_configuration": {
        "host": "_HOST_NAME_",
        "port": "_KMS_PORT_NUMBER_",
        "ca_certificate_path": "_KMS_CREDENTIALS_CA_",
        "client_certificate_path": "_KMS_CLIENT_CREDENTIALS_CERT_",
        "client_private_key_path": "_KMS_CLIENT_CREDENTIALS_KEY_"
  },
  "cache": {
        "enable": "_ENABLE_",
        "max_cache_capacity": "_MAX_CACHE_CAPACITY_",
        "max_object_size": "_MAX_OBJECT_SIZE_",
        "max_object_lifetime": "_MAX_OBJECT_LIFETIME_"
  },
  "external_request_timeout": "_EXTERNAL_REQUEST_TIMEOUT_",
  "logs": {
        "enable": "_ENABLE_",
        "kinds": ["_KIND_FILTER_LIST_"],
        "severities": ["_SEVERITY_FILTER_LIST_"]
  },
  "jwt_supported_signing_algorithms": ["_JWT_SUPPORTED_SIGNING_ALGORITHMS_
LIST_"]
}
```

The tables below describe the parameters in the *config.json* file. The first table lists simple parameters which do not contain any sub-objects. More complex parameters have their own table.

Unless otherwise specified, in the configuration files:

- All “String” fields are security-limited to 10,000 characters for security reasons,
- All ”Array” fields are limited to 500 items for security reasons.

7.3.1 Simple parameters

Parameter	Description	Type	Optional/ mandatory
kacls_url	The SDS encryption service for Google Workspace URL used to prevent "Man-in-the-middle" attacks. E.g., *https://<cse.example.com>*.	String	Mandatory
port	Port on which the SDS encryption service for Google Workspace listens. Port 3000 by default.	Integer	Optional
name	Name of the SDS encryption service for Google Workspace.	String	Optional
persistence_ type	Mode used for storing KEKs. The prescribed values are: • "json_file" if keys are stored in the *keks.json* file, • "kms"' if keys are stored in a key management system (KMS).	String	Mandatory
kacls_kek_ label	Label of the KEK if it is stored in a KMS.	String between 1 and 255 characters long.	Mandatory if "persistence_ type": "kms"
external_ request_ timeout	Timeout in milliseconds before canceling an external request (7000 ms by default). This timeout does not apply to the KMIP-related requests when using a KMS.	Integer	Optional
jwt_supported_ signing_ algorithms	List of the allowed signature algorithms for checking the validity of authorization and authentication tokens. Supported algorithms are: ["RS256", "RS384", "RS512", "ES256", "ES384", "ES512", "PS256", "PS384", "PS512"].	String	Mandatory (at least one algorithm)

7.3.2 tenants parameter

Contains configuration information specific to each tenant. They are grouped by "tenantid", which is the tenant unique identifier and is mandatory. The "tenantid" object includes the following components:

Parameter	Description	Type	Optional/ mandatory
user_authentication: Object containing the configurations that allow the client to authenticate.			
enable_wellknown_cse_ discovery	Activated by default (true). Enables the use of the .well-known remote configuration to validate user authentication.	Boolean	Optional

Parameter	Description	Type	Optional/ mandatory
idps	Object array containing the configuration to identity providers for client authentication. It must include either both elements "discovery_uri" and "client_id", or the three elements "jwks_uri", "audience" and "issuer": • discovery_uri: URL to the OpenID JSON configuration file, • client_id: recipient of the JWT authentication token (see RFC 7519), • jwks_uri : URL to the JSON Web Key Set file, • audience: recipient of the JWT authentication token (see RFC 7519), • issuer: issuer of the JWT authentication token (see RFC 7519). An entry must be added for each identity provider. To find out how to get the values of the elements, see Retrieving import values.	Array	Optional
admin_authentication: JSON object array describing how to validate the authentication of administration routes via a local configuration.			
discovery_uri	URL to the OpenID JSON configuration file.	String	Optional
client_id	Recipient of the JWT authentication token (see RFC 7519). An entry must be added for each identity provider.	String	Optional
wrapprivatekey_authentication : JSON object array describing how to validate the authentication of */wrapprivatekey* routes via a local configuration.			
discovery_uri	URL to the OpenID JSON configuration file.	String	Optional
client_id	Recipient of the JWT authentication token (see RFC 7519). An entry must be added for each identity provider.	String	Optional
migration: JSON object array containing information for the migration of a KACLS to another or the use of a backup KACLS. For mor information, see Migrating an external key service to another.			
enabled	Enables or disables the migration from one KACLS to another.	Boolean	Optional

Parameter	Description	Type	Optional/ mandatory
kaclstokacls_token	• kid: identifier used to generate a JWKS. • format: format of the key (PEM). • key: private key in PEM format. Used to form JWT authentication tokens and generate a JWKS making it possible to check these tokens. • duration: lifetime of the generated JWT authentication token.	Array	Mandatory if the enabled field is set to true
acls	• kacls_urls: list of allowed KACLS URLs. Must begin with "*https://*".	Array	Mandatory if the enabled field is set to true
crypto_backends: JSON object array containing the definition of the backend component performing the cryptographic operations.			
id	ID of the cryptographic backend in the form of a UUID v4 that you generate.	String in UUID format	
name	Name of your choice for the cryptographic backend.	String	
type	Cryptographic backend type. The possible values are: • kms to use the KMS API • node to use the SDS encryption service for Google Workspace	String	
configuration	JSON object array containing the cryptographic backend configuration in the "kms" mode. It includes the following fields: • host: URL of the KMS • port: KMS port • vendor: KMS vendor (Thales) • model: KMS model (ciphertrust) • credentials: - ca_certificate_path: path to the certification authority - client_certificate_path: path to the KMS client certificate - client_private_key_path: path to the KMS user key	Array	Mandatory in "kms" mode"
keys: Object containing the UUID of the cryptographic backend to be used for cryptographic operations.			

Parameter	Description	Type	Optional/ mandatory
users_private_keys	• crypto_backend: object defining the cryptographic backend to be used to get private keys. - id: UUID of the cryptographic backend defined in the "crypto_backend.id" object.	String	Mandatory if the crypto_ backend object is configured
kmaas:: JSON object array describing how to validate the authentication of the SDS CryptoAPI *encrypt* and *decrypt* routes. For more information, see SDS CryptoAPI			
user_authentication	Object containing the configurations that allow the client to authenticate. It includes the following fields: • discovery_uri: URL to the OpenID JSON configuration file. • client_id: Recipient of the JWT authentication token (see RFC 7519). An entry must be added for each identity provider.	Array	Mandatory
delegate: JSON object array containing the definition of the *delegate* software component performing the delegation operations. This feature is in alpha version. For more information, refer to Enabling Google Meet hardware use (alpha version).			
authentication	JSON object allowing to sign JWT tokens in RS256. It includes the following field: • key: private key signed in base64 and used to sign authentication token.	Object	Mandatory if the enable field is set to true
enable	Enable the delegation feature.	Boolean	Mandatory

Authentication

The values of the authentication parameters depends on the method used:

- OpenID configuration file (for OneLogin for instance):
  - `_AUTHENTICATION_OPEN_ID_CONFIGURATION_URL_` is the URL for the OpenID configuration file. For OneLogin authentication, it must resemble: https://<domain>.onelogin.com/oidc/2/.well-known/openid-configuration.
    For Google authentication, it is similar to: *https://accounts.google.com/.well-known/openid-configuration*.
  - `_AUTHENTICATION_AUDIENCE_`
    For OneLogin authentication, it corresponds to the *audience* setting.
    For Google authentication, it corresponds to the *OAuth Client ID* setting.

- JSON Web Key Set file (JWKS):
  - _IDPS_JWKS_URL_ corresponds to the URL for the JWKS file that contains the signature and/or encryption keys.
    For Google authentication, it is similar to:
    https://www.googleapis.com/service_accounts/v1/jwk/gsuitecse-tokenissuer-drive@system.gserviceaccount.com
  - _IDPS_AUDIENCE_
    For Google authentication, it corresponds to the *OAuth Client ID* setting.
  - _IDPS_ISSUER_ corresponds to the issuer of the authentication token.
    For Google authentication, it is gsuitecse-tokenissuer-drive@system.gserviceaccount.com

For more information, see section Retrieving import values

Configuration of the cryptographic backend

A cryptographic backend is a tool allowing to choose the type of encryption and signature that the SDS encryption service for Google Workspace will apply, including the Gmail private keys. Two backend modes are available: 'node' for the SDS encryption service for Google Workspace, and 'kms' for the KMS.

The supported algorithms for each mode are:

	Decryption algorithm	Signature algorithm
'node' mode	RSA/ECB/OAEPwithSHA-1andMGF1Padding, RSA/ECB/OAEPwithSHA-256andMGF1Padding, RSA/ECB/OAEPwithSHA-512andMGF1Padding	SHA1withRSA/PSS, SHA256withRSA/PSS, SHA512withRSA/PSS, SHA1withRSA SHA256withRSA SHA512withRSA
Node' mode with the CVE-2023-46809 enabled (See the limitations ci-dessous)	RSA/ECB/PKCS1Padding, RSA/ECB/OAEPwithSHA-1andMGF1Padding, RSA/ECB/OAEPwithSHA-256andMGF1Padding, RSA/ECB/OAEPwithSHA-512andMGF1Padding	SHA1withRSA/PSS, SHA256withRSA/PSS, SHA512withRSA/PSS, SHA1withRSA SHA256withRSA SHA512withRSA
'kms' mode	RSA/ECB/PKCS1Padding	SHA1withRSA/PSS, SHA256withRSA/PSS, SHA512withRSA/PSS, SHA1withRSA, SHA256withRSA SHA512withRSA

- With the 'kms' mode, be aware of the following limitations:
  - The SDS encryption service for Google Workspace is only compatible with the API of the Ciphertrust Manager from Thales.
  - Only one version of the private keys associated with a Ciphertrust keyname is supported. You should therefore not use the versioning feature of the Thales Ciphertrust Manager KMS for the encryption or signature keys used for the SDS encryption service for Google Workspace for Gmail.
  - Since PKCS 1.5 message signing is not compatible with Ciphertrust's REST API, the SDS encryption service for Google Workspace uses the KMIP protocol to perform signing operations. Configure the section kmip_configuration of the *config.json* file to sign messages in PKCS 1.5.

- In 'node' mode, data encryption using the PKCS 1.5 algorithm is vulnerable, particularly to the *Marvin Attack*. NodeJS version 20.11.1 has therefore removed the use of this algorithm via CVE-2023-46809. As Google only supports PKCS 1.5 for message signature and encryption, you must disable this CVE in NodeJS for the SDS encryption service for Google Workspace to use this feature.
  To do so, **in RPM mode**:
  1. Open the */etc/systemd/system/cse.service* file.
  2. Replace the `ExecStart=/usr/bin/env node cse`
     - by -
     `ExecStart=/usr/bin/env node --security-revert=CVE-2023-46809 cse`

     This bypass is not useful if you are in 'kms' mode.

In Docker mode, CVE-2023-4680 is applied by default and you do not have to modify *cse.service*. A warning log notifying that the CVE is disabled is issued when the SDS encryption service for Google Workspace starts, which is normal.

NOTE
If the HTTPS proxy is enabled, you must exclude the KMS domain from the proxy via the *no_proxy* environment variable. For more information, see the section Configuring proxy access.

Below is an example of a cryptographic backend configuration in "kms" mode with the Thales Ciphertrust Manager KMS:

```
"crypto_backends": [
  {
        "id": "3711cab6-83fc-4a97-9438-a1500edfd01a",
        "name": "My crypto tool",
        "type": "kms",
        "configuration": {
          "host": "https://web.ciphertrustmanager.local",
          "model": "ciphertrust",
          "vendor": "thales",
          "port": 443,
          "credentials": {
                 "ca": "/etc/stormshield/cse/ca_kms.pem",
                 "cert": "/etc/stormshield/cse/cert_kms.pem",
                 "key": "/etc/stormshield/cse/key_kms.pem"
          }
        }
  }
],
"keys": {
        "users_private_keys": {
          "crypto_backend": {
                 "id": "3711cab6-83fc-4a97-9438-a1500edfd01a"
          }
        }
}
```

7.3.3 authorization parameter

Array of JSON objects that describes how the authorization token generated by Google is verified. It includes the following components.

Add one entry per Google service (e.g., Meet, Drive, Calendar, Gmail). The values of these settings are provided in the Google documentation. For more information, see Example of the authorization parameter for Google services

Parameter	Description	Type	Optional/ mandatory
issuer	Issuer of the JWT authorization token (see RFC 7519).	String	Optional
url	URL to the JWKS JSON file.	String	Mandatory
audience	Recipient of the JWT authorization token (see RFC 7519).	String	Optional

Example of the authorization parameter for Google services

This extract of the *config.json* file is an example of how the `authorization` token can be configured for the Drive, Meet, Calendar and Gmail Google services, Gmail and the migration of one KACLS to another. You can customize the rules that allow or deny a request to the SDS encryption service for Google Workspace, using Open Policy Agent (OPA) policies. For more information, see the section Customizing the authorization rules.

```
"authorization": [
    {
    "url": "https://www.googleapis.com/service_accounts/v1/jwk/gsuitecse-tokenissuer-drive@system.gserviceaccount.com",
    "issuer": "gsuitecse-tokenissuer-drive@system.gserviceaccount.com",
    "audience": "cse-authorization"
    },
    {
    "url": "https://www.googleapis.com/service_accounts/v1/jwk/gsuitecse-tokenissuer-meet@system.gserviceaccount.com",
    "issuer": "gsuitecse-tokenissuer-meet@system.gserviceaccount.com",
    "audience": "cse-authorization"
    }
    {
    "url": "https://www.googleapis.com/service_accounts/v1/jwk/gsuitecse-tokenissuer-calendar@system.gserviceaccount.com",
    "issuer": "gsuitecse-tokenissuer-calendar@system.gserviceaccount.com",
    "audience": "cse-authorization"
    }
    {
    "url": "https://www.googleapis.com/service_accounts/v1/jwk/gsuitecse-tokenissuer-gmail@system.gserviceaccount.com",
    "issuer": "gsuitecse-tokenissuer-gmail@system.gserviceaccount.com",
    "audience": "cse-authorization"
    }
    {
    "url": "https://www.googleapis.com/service_accounts/v1/jwk/apps-security-cse-kaclscommunication@system.gserviceaccount.com",
```

"issuer": "apps-security-cse-kaclscommunication@system.gserviceaccount.com"",
"audience": "cse-authorization"
}
]

7.3.4 https parameter

JSON object that describes the HTTPS certificate. To be used when you want to run the server in secure mode. It includes the following components:

Parameter	Description	Type	Optional/ mandatory
credentials	• key: path to the private HTTPS key in PEM format. • cert: path to the HTTPS certificate in PEM format. • ca (optional): path to the HTTPS certification authority in PEM format.	Object	Optional

7.3.5 keks parameter

JSON object describing the refresh frequency of the KEK while the SDS encryption service for Google Workspace is running. It includes the following components

Parameter	Description	Type	Optional/ mandatory
auto_refresh	• scheduled.interval_seconds: frequency at which the KEKs are scheduled to be automatically refreshed by the SDS encryption service for Google Workspace (by default 86400 seconds, minimum value 1800 seconds). Make sure you specify a value matching your needs and use. Any value lower than 1800 seconds (30 minutes) will be considered invalid and will prevent the server from starting. • minimum_interval_seconds : minimum interval between two refresh operations, whether periodic or one-off (3600 seconds by default). Limits queries to the KMS. KEK refresh is only applicable if the parameter persistence_type: "kms"	Integer in seconds	Optional

7.3.6 kmip_configuration parameter

JSON object that describes the KMS configuration. It is mandatory if "persistence_type": "kms". It includes the following objects:

Parameter	Description	Type	Optional/ mandatory
host	KMS server.	String	Mandatory
port	Port that the KMS listens on (optional, 5696 by default).	Integer	Optional

Parameter	Description	Type	Optional/ mandatory
client_private_key_path	Path to the private key of the KMS client in PEM format.	String	Mandatory
client_certificate_path	Path to the certificate of the KMS client in PEM format	String	Mandatory
ca_certificate_path	Path to the certification authority of the KMS client in PEM format.	String	Optional

NOTE:
The maximum number of simultaneous connections to the KMS server depends on your infrastructure and the KMS configuration.

7.3.7 cache parameter

JSON object that describes the configuration of the cache. Set the values according to your configuration, e.g., available memory. It includes the following objects:

Parameter	Description	Type	Optional/ mandatory
enable	Activation status of the cache (true by default).	Boolean	Optional
max_cache_capacity	Maximum capacity of the cache (100 MB by default).	Integer in MB	Optional
max_object_size	Maximum size of a cached object in KB (100 KB by default). OpenId, jwks and remote configurations (*cse-config*) are cached. Set a value high enough to store these objects.	Integer in MB	Optional
max_object_lifetime	Lifetime of a cached object (1440 min by default). We advise against exceeding the lifetime of tokens provided by the identity providers.	Integer in minutes	Optional

7.3.8 logs parameter

JSON object that allows configuring how to display the logs in the new format. If the object is missing in the *config.json* file, only the logs with the old format are displayed. It includes the following objects.

For more information, see the section Understanding the new log format.

Parameter	Description	Type	Optional/ mandatory
enable	Enable log display.	Boolean	Mandatory

Parameter	Description	Type	Optional/ mandatory
kinds	List of the log families to be displayed in logs. The possible values are “http”, "domain" and "system".	String	Mandatory
severities	List of the severity levels to be displayed in the logs. The possible values are "emerg", "alert", "crit","err", "warning", "notice", "info", and "debug".	String	Mandatory

7.4 Using remote authentication

You can create a remote configuration file, *cse-configuration*, to share your authentication credentials with external collaborators. This file must be in a directory */.well-known/*, located at the root of the domain (*https://cse.${domain}/.well-known/*). It makes it possible to verify the signature of the user's token and indicate which identity providers to use.

The remote file will be looked up if the *user_authentication* section in the *config.json* file is not filled in. It is retrieved during authentication via the URL:
https://cse.${domain_from_email_from_token}/.well-known/cse-configuration

This is a fixed URL. Ensure that it can be reached by using the SDS encryption service for Google Workspace.

NOTE
For security reasons, the routes *privilegedwrap*, *privilegedunwrap*, *privilegedprivatekeydecrypt*, and *wrapprivatekey* are not allowed for remote authentication.

For more information, refer to the Google documentation Connect to identity provider for client-side encryption website.

To create the remote authentication file:

- Create a file named *cse-configuration*. Its contents are as follows:

```
{
"name": "_IDP_NAME_",
"client_id": "_AUTHENTICATION_AUDIENCE_",
"discovery_uri": "_AUTHENTICATION_OPEN_ID_CONFIGURATION_URL_",
"grant_type": "_GRANT_TYPE_"
}
```

Parameter	Description	Type	Authorized values	Optional/ mandatory
name	Name of the identity provider.	String		Optional
client_id	OIDC (OpenID Connect) client ID that the client application uses to get a JWT.	String		Mandatory
discovery_uri	OIDC discovery URL, as defined in the OpenID specification.	String		Mandatory

Parameter	Description	Type	Authorized values	Optional/ mandatory
grant_type	OAuth traffic used for OIDC	String	implicit | authorization_code	Optional

If you use the Google identity provider, the values of the authentication settings are as follows:

```
{
"name": https://accounts.google.com
"client_id": "37*********",
"discovery_uri": "https://accounts.google.com/.well-known/openid-configuration"
}
```

7.5 Using the SDS encryption service for Google Workspace in secure mode (HTTPS, KMS)

To secure your SDS encryption service for Google Workspace, you can:

- Secure connections between the SDS encryption service for Google Workspace components with HTTPS.
- Set up a key management system (KMS) to store your KEKs. In this case, you will use secure protocol KMIP.

In HTTPS and KMIP, you need a private key and certificate in PEM format, and as an option, a certification authority for each protocol.

Private keys and certificates are highly sensitive items in terms of security. You must follow the ANSSI recommendations (in French only) concerning their life cycle.

1. Assign the read and write access privileges held by the current user to the files and read access to the current group:
   ```
   # chmod u=rw,g=r,o= <ca-https.file> <cert-https.file> <key-https.file>
   # chmod u=rw,g=r,o= <ca-kms.file> <cert-kms.file> <key-kms.file>
   ```
   Do not assign any run privileges on these files, or any privileges to other users. If access privileges are too permissive, a warning log will be generated when the SDS encryption service for Google Workspace starts, but will not prevent it from launching.
   During installation, the *stormshield-cse* user is the owner of the configuration files by default. Do not change the owner.
2. In the *config.json* file, specify the path of these files:
   - For HTTPS in the *https.credentials* section,
   - For KMIP in the *kmip_configuration* section.

7.6 Using the SDS encryption service for Google Workspace with Google applications

The SDS encryption service for Google Workspace allows users to encrypt data for the following Google applications:

Application	Encryption perimeter	Availability	Use
Google Drive	Encrypting Google Drive confidential documents.	• Windows and macOS desktops (Google Drive for Desktop) • iOS and Android mobile devices	• *well-known* remote file • Local configuration
Google Meet	Encryption of video conferences and calls created with Google Meet	• Google Workspace web client • iOS and Android mobile devices	• *well-known* remote file • Local configuration
Google Calendar	Encrypting a meeting created with Google Calendar: related description, attachments, and Meet conference.	• Google Workspace web client • iOS and Android mobile devices	• *well-known* remote file • Local configuration

7.6.1 Enabling the use of a Google application via a remote file

- Configure your identity provider as described in the Google documentation Connect to your identity provider for client-side encryption.

7.6.2 Enabling the use of a Google application in the local configuration

- In the *config.json* local configuration file, declare the *client_id* of your applications in the *user_authentication - idps* section.

For example, if you use the Google Identity provider, this section of the *config.json* file should be as follows for drivefs, drive-android, and drive-ios:

```
"user_authentication": {
  "idps": [
  {
        "discovery_uri": "https://accounts.google.com/.well-known/openid-
configuration",
        "client_id": "947318989803-
k88lapdik9bledfml8rr69ic6d3rdv57.apps.googleusercontent.com"
  },
  {
        "discovery_uri": "https://accounts.google.com/.well-known/openid-
configuration",
        "client_id": "378076965553-
g44pde5vvf113hdd8j84a32kl4e7hqa0.apps.googleusercontent.com"
  },
  {
        "discovery_uri": "https://accounts.google.com/.well-known/openid-
configuration",
        "client_id": "640853332981-
r48oo8ht2kl9v029vsgtatkh4gtue0pn.apps.googleusercontent.com"
```

```
    }
    ]
},
```

For more information, refer to the Configure the SDS encryption service for Google Workspace section.

Currently, sharing encrypted content with external users is only available on web applications. This feature will be available on mobile applications in a future release.

7.6.3 Enabling external user access for Google Drive and Google Meet

You can share encrypted content with external users in Google Drive and invite external participants to encrypted Google Meet conferences.

This feature will be available on mobile applications in a future release of the SDS encryption service for Google Workspace.

To enable the Guest Access feature:

1. Create a dedicated identity provider for Google Drive and declare all desired external users so that they can authenticate. Only declared users will be able to access shared encrypted content. For more information, see Configuring the identity provider.
2. In the same way, create a dedicated identity provider for Google Meet.
3. In the Google Workspace administration interface, add the identity providers specific to external users. For more information, refer to the Google documentation on configuring a guest IdP for all external users.

Once the Guest Access feature has been enabled:

- When a document is shared via Google Drive to external users, they receive emails enabling them to connect to the dedicated identity provider and view the document.
  If the external users do not have a Google account, they must also validate their email addresses with Google every 7 days.
- When you invite external users to an encrypted Google Meet conference, they receive emails containing a link to join the encrypted conference directly. To do this, they must authenticate to the dedicated identity provider.

If using the Guest Access feature, it is not yet possible to share data between participants using a well-known file.

7.6.4 Enabling Google Meet hardware use (alpha version)

To attend an encrypted Google Meet conference from a room equipped with Meet hardware, the user must delegate authentication.

To enable delegation:

- In the local configuration file *config.json*, declare your authentication information in the section *tenantid - delegate*.

For more information, refer to the section tenants parameter.

8. Configuring Gmail usage

The SDS encryption service for Google Workspace can be used with Gmail. This feature is available with the web version of Gmail and with the mobile application on Android and iOS.

Two mode are available:

- Gmail standard mode with encrypted keys stored at Google,
- Gmail advanced mode based on a Key management system (KMS) with keys stored in the KMS.

Depending on the mode you choose, you need to know the limitations about the supported algorithms. For more information, refer to the table in the section Configuration of the cryptographic backend.

Stormshield recommends the use of different key pairs for encryption and signature. In this case, repeat the step Encrypting users' private keys with the SDS encryption service for Google Workspace for each private key.

To help you using of Gmail with the client-side encryption service, you can implement the SDS Orchestrator solution. It provides and manages encryption and signature keys for your Google Workspace accounts. For more information, please contact your Stormshield sales representative.

8.1 Using Gmail in standard mode

8.1.1 Configuring the SDS encryption service for Google Workspace

Modify the *config.json* file as described in the steps below: For more information, refer to the Configure the SDS encryption service for Google Workspace section.

1. Fill in the `crypto_backend` section, and assign the "node" value to the `type` field. Do not fill in the `configuration` block. See crypto_backend parameters and Configuration of the cryptographic backend.
2. In the `id` field of the `keys` section, enter the UUID of the cryptographic backend set in step 1. See keys parameters and Configuration of the cryptographic backend.
3. Fill in the `authorization` section with Gmail information as shown in Authorization settings.
4. Fill in the `wrapprivatekey_authentication` section as shown in wrapprivatekey_authentication parameters.
5. Optional. Fill in the `admin_authentication` section as shown in admin_authentication parameters to perform privileged operations.

8.1.2 Encrypting users' private keys with the SDS encryption service for Google Workspace

Ensure that the SDS encryption service for Google Workspace is fully configured and operational before following the steps below.

For every private key to be encrypted, call up the */wrapprivatekey* API route in POST with the following headers and payload:

- **URL**: in the format "{protocol: http | https}://{kacls url}/api/v1/{tenantId}/wrapprivatekey", where:
  - {kacls url} is the URL of the key service that you have declared in Specifying the External key service
  - {tenantId} is your tenant's UUID.
- **Mandatory headers**:
  - Content-Type: 'application/json',
  - Connection: 'keep-alive',
- **Payload**:

Field	Description
authentication	Valid authentication token
private_key	The user's private key encrypted in pem format, and base64-encoded
perimeter_id	Optional string
supported_algorithms	List of supported algorithms:['RSA/ECB/PKCS1Padding', 'RSA/ECB/OAEPwithSHA-1andMGF1Padding', 'RSA/ECB/OAEPwithSHA-256andMGF1Padding', 'RSA/ECB/OAEPwithSHA-512andMGF1Padding' 'SHA1withRSA', 'SHA256withRSA', 'SHA512withRSA', 'SHA1withRSA/PSS', 'SHA256withRSA/PSS', 'SHA512withRSA/PSS'];

EXAMPLE:
Request enabling the encryption of a private key, sent in POST over the */wrapprivatekey* route:

```
{
  "authentication": "eyJhbGciOiJSUzI1NiIsImtpZCI6ImFjZGEz...",
  "private_key": "LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVk...",
  "supported_algorithms": [
       "RSA/ECB/PKCS1Padding",
       "RSA/ECB/OAEPwithSHA-1andMGF1Padding",
       "RSA/ECB/OAEPwithSHA-256andMGF1Padding",
       "RSA/ECB/OAEPwithSHA-512andMGF1Padding,"
       "SHA1withRSA",
       "SHA256withRSA",
       "SHA512withRSA"
       "SHA1withRSA/PSS",
       "SHA256withRSA/PSS",
       "SHA512withRSA/PSS"
  ]
}
```

The response to this request is a JSON object named *wrapped_private_key* which contains a string representing the encrypted private key.

8.1.3 Providing private keys to Google

- Enable Gmail and provide your users' encrypted private keys and certification chains. For more information, refer to the Google documentation Gmail only: Set up your organization for client-side encryption.
  Certification chains must meet the following Google specifications:
  - S/MIME certificate profiles,
  - Set up rules to require S/MIME.

8.1.4 Using Gmail

- To use Gmail to send encrypted messages to internal or external users, refer to Google documentation Learn about Gmail Client-side encryption.

8.2 Using Gmail in advanced mode based on a KMS

8.2.1 Configuring the SDS encryption service for Google Workspace

Modify the *config.json* file as described in the steps below: For more information, refer to the Configure the SDS encryption service for Google Workspace section.

1. Fill in the whole `crypto_backend` section, and assign the "kms" value to the `type` field. See crypto_backend parameters and Configuration of the cryptographic backend.
2. In the `id` field of the `keys` section, enter the UUID of the cryptographic backend set in step 1. See keys parameters and Configuration of the cryptographic backend.
3. Fill in the `authorization` section with Gmail information as shown in Authorization settings.
4. Fill in the `wrapprivatekey_authentication` section as shown in wrapprivatekey_ authentication parameters.
5. Optional. Fill in the `admin_authentication` section as shown in admin_authentication parameters to perform privileged operations.

8.2.2 Encrypting users' private keys with the SDS encryption service for Google Workspace

Ensure that the SDS encryption service for Google Workspace is fully configured and operational before following the steps below.

- For every private key to be encrypted, call up the */wrapprivatekey* API route in POST with the following headers and payload:
- **URL**: in the format "{protocol: http | https}://{kacls url}/api/v1/{tenantId}/wrapprivatekey", where:
  - {kacls url} is the URL of the key service that you have declared in Specifying the External key service
  - {tenantId} is your tenant's UUID.

- **Mandatory headers**:
  - Content-Type: 'application/json',
  - Connection: 'keep-alive',
- **Payload**:

Field	Description
authentication	Valid administrator authentication token
private_key	ID of the user's private key stored in the KMS, and base64-encoded.
perimeter_id	Optional string
supported_algorithms	List of supported algorithms: ['RSA/ECB/PKCS1Padding', 'SHA1withRSA', 'SHA256withRSA', 'SHA512withRSA', 'SHA1withRSA/PSS', 'SHA256withRSA/PSS', 'SHA512withRSA/PSS'];
public_key	Public key of the user in PEM format, and base64-encoded.

EXAMPLE:
Request enabling the encryption of a private key, sent in POST over the */wrapprivatekey* route:

```
{
  "authentication": "eyJhbGciOiJSUzI1NiIsImtpZCI6ImFjZGEz...",
  "private_key": "LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVk...",
  "supported_algorithms": [
        "RSA/ECB/PKCS1Padding",
        "SHA1withRSA/PSS",
        "SHA256withRSA/PSS",
        "SHA512withRSA",
        "SHA512withRSA/PSS"
  ],
  "public_key" : "e32tLS1CRUdJTiBSU0FgUFJJVkFURSBLRck..."
}
```

8.2.3 Providing the encrypted ID of the private keys to Google

- Enable Gmail and provide your user's private key encrypted IDs and certification chains. For more information, refer to the Google documentation Gmail only: Set up your organization for client-side encryption.
  Certification chains must meet the following Google specifications:
  - S/MIME certificate profiles,
  - Set up rules to require S/MIME signature and encryption

8.3 Using Gmail

- To use Gmail to send encrypted messages to internal or external users, refer to Google documentation Learn about Gmail Client-side encryption.

9. Configuring KEKs

The SDS encryption service for Google Workspace uses key encryption keys, i.e., KEKs, to wrap and unwrap Data Encryption Keys (DEKs). Google provides DEKs to encrypt/decrypt data.

There are two ways to store keys:

- Standalone mode: KEKs are stored in plaintext in the */etc/stormshield/cse/keks.json* file on the server.
- KMS mode: KEKs are stored in a key management system (KMS). They are selected in the KMS by using the value of the *kacls_kek_label* parameter in the *config.json* file. They are refreshed regularly, based on the value of the *keks* parameter in the *config.json* file. See keks parameter.

KEKs are highly sensitive items in terms of security. You must follow the ANSSI recommendations concerning their life cycle.

9.1 Configuring KEKs in standalone mode

9.1.1 Generating KEKs and MKEKs

KEKs are 256-byte AES-256 keys listed in the *keks.json* file in the form of base64-encoded character strings.

The SDS encryption service for Google Workspace does not generate KEKs, so you must create them beforehand. You can use OpenSSL to do so, for example:

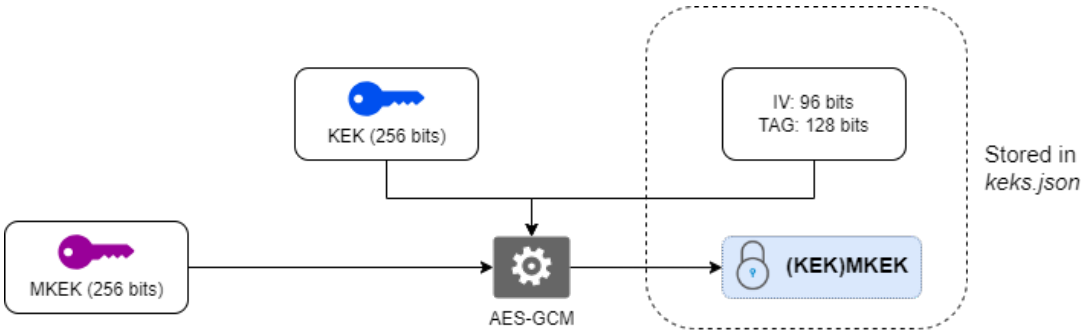
- On a Red Hat system, install OpenSSL by using the following command:
  `yum install openssl`.
- On a Linux system, install OpenSSL using the package manager corresponding to your distribution.
- With the `rand` command, data directly encoded in base64 can be generated:
  `openssl rand -base64 32`

To improve the security of the SDS encryption service for Google Workspace, you can also use OpenSSL to generate Master encryption keys (i.e., MKEK). An MKEK is used to encrypt your KEKs so that they are not exposed in clear text in the configuration file.

The MKEKs must be as follows:

- They must be 256 bits in size,
- They must be base64-encoded,
- They must encrypt the KEKs with a 12 byte initialization vector and a 16 byte authentication tag.

The diagram below illustrates how the MKEK encrypts KEKs.



For more information on aes-256-gcm encryption to generate the MKEK, the encrypted KEK, the initialization vector and the tag, see the ANSSI document ANSSI-PG-083.

If you are using at least one KEK encrypted by an MKEK, you must declare the MKEK_VALUE environment variable containing the value of the MKEK encryption key before starting the SDS encryption service for Google Workspace.

9.1.2 Preparing the key encryption key file

The SDS encryption service for Google Workspace manages key encryption keys, i.e., KEKs, and stores them in the *keks.json* file. This file must be saved in the directory */etc/stormshield/cse*.

Creating the KEK file

In RPM mode, a file template can be found in */etc/stormshield/cse* to assist you.

- Create your own KEK file from the copy of the template using the following command:
```
# cd /etc/stormshield/cse
# cp --preserve keks.json.template keks.json
```

In Docker mode, the file is located in the dedicated directory you have created during installation. For more information, see Installing the SDS encryption service for Google Workspace via a Docker image.

Assigning access privileges to the file

1. Assign the read and write access privileges held by the current user to the *config.json* file and read access to the current group:
```
# chmod u=rw,g=r,o= keks.json
```
2. Set *stormshield-cse* as the file owner.
```
# chown stormshield-cse keks.json
```

If access privileges are too permissive, a warning log will be generated when the SDS encryption service for Google Workspace starts, but will not prevent it from launching.

9.1.3 Adding KEKs to the file

After you have generated your KEKs, encrypted or not, add them manually to the *keks.json* file.

The same file can include both types of KEKs: encrypted and not encrypted.

Parameter	Description	Optional/ mandatory
tenant_id	UUID v4 of your tenant, the same that you have specified for the External key service.	Mandatory
active_kek_id	ID of the active KEK that will be used to encrypt keys.	Mandatory
keks: JSON object array containing the definition of non encrypted KEKs.		
id	Unique ID generated in UUID v4 format.	Mandatory
kek_b64	Value of the KEK.	
encrypted_keks: JSON object array containing the definition of the MKEK-encrypted KEKs.		
id	Unique ID generated in UUID v4 format.	Mandatory

Parameter	Description	Optional/ mandatory
encrypted_kek_b64	Value of the encrypted KEK. It must imperatively be generated using the *aes-256-gcm* encryption algorithm.	
crypto_material	• crypto_context: object containing the following fields, generated while encrypting the KEK with the MKEK: - iv: initialization vector - tag: authentication tag	
	• encryption_algorithm: must be *aes-256-gcm*	
	• m_kek_location: information about the location of the MKEK used to encrypt the KEK - key_name: name of the environment variable containing the MKEK value - type: must be *env*	

Generate v4 UUIDs with any tools of your choice (e.g., UUID Generator).

EXAMPLE
The file contents below are given simply as an example and must not be used as such in your SDS encryption service for Google Workspace configuration.

```
{
 "tenants": [
   {
    "tenant_id": "6666fde8-9957-4846-91fc-a59158b4febc",
    "active_kek_id": "3a55f631-c27a-4ccf-94af-1e36e5b7b72b",
    "keks": [
        {
          "kek_b64": "jiMvs6yEMvI244PCPy5B7qlVsBcbkl6lZtOyQKIeP08=",
          "id": "3a55f631-c27a-4ccf-94af-1e36e5b7b72b"
        },
        {
          "kek_b64": "+F/2qYyIiAMSfUoMjzXq6W6yvGeppo21R0pVpspJ5UA=",
          "id": "e134c09e-1398-4b7a-bc61-c79c46a7878e"
        },
        {
          "kek_b64": "oZgvT+CDhLNYZjFpIXBhBZtvRHComBomNCuwZKM0Oto=",
          "id": "3a66f631-c58a-4oif-94af-1e36e5b7l71t"
        }
       ]
  }
 ]
}
```

9.1.4 Renewing a KEK

To renew a KEK, ensure that the previous keys are still accessible for unwrapping, and that the new KEK cannot be used before it is deployed on all CSE servers.

1. Add a new KEK to the *keks.json* file:
   a. Retrieve the *keks.json* file on one of the CSE servers.
   b. Generate a new AES 256 KEK.
   c. Add this KEK to the file and assign a new unique ID to it.
2. Publish the *keks.json* file successively on each CSE server:
   a. Replace the *keks.json* file with the one modified in step 1. Ensure that you keep the same access privileges as for the existing file.
   b. Run the restart command on the server.
   The server will restart and reload its list of KEKs from the *keks.json* file. The active KEK does not change for the moment.
3. Set the new active KEK in the *keks.json* file:
   a. Edit the *keks.json* file again.
   b. Change the value associated with `active_kek_id` so that it points to the ID of the KEK generated in step 1.
4. Publish the *keks.json* file again on all the CSE servers.
   On each successive CSE server:
   a. Replace the *keks.json* file with the one modified in step 3. Ensure that you keep the same access privileges as for the existing file.
   b. Run the restart command on the server.
   The server will restart and reload its list of KEKs from the *keks.json* file. The active KEK is changed and the server is ready to wrap keys with the new KEK.

   In step 4, if a wrapping request is submitted on a server that uses the new KEK, all the other servers can respond to an unwrapping request regardless of their status, since they all know the new key.

9.2 Configuring KEKs in KMS mode

9.2.1 Requirements

To use the SDS encryption service for Google Workspace with a key management system (KMS), you must meet the following requirements:

- The version of the protocol used for connecting to the KMS must be KMIP 1.4,
- The algorithm used for wrapping KEKs must be AES-GCM.

9.2.2 Generating KEKs in the KMS

In the interface of the KMS, create a new key with the following values:

name	<name_of_your_kek>
algorithm	AES-256
exportable	true
usage	not necessary

custom attribute	x-sds-kacls-kek-label:<my_kacls_keks_label> Label that identifies all your KEKs. It must match the value of the *kacls_kek_label* field in the *config.json* file.
	x-sds-kacls-tenant-id:<UUIDv4> Identifier of your tenant in UUID v4 format. This ID must match the one specified for the External key service.

The KMIP client must be allowed to use this key.

When the SDS encryption service for Google Workspace is being initialized, all KEKs that match the *x-sds-kacls-kek-label* label will be retrieved, regardless of their status in the KMS.

While all retrieved KEKs can be used for unwrap operations, only the most recent key of each tenant will be used for wrap operations. This particular KEK is identified by the *is_active_kek:true* field in logs.

9.2.3 Renewing KEKs in the KMS

For greater security, you can regularly renew the active KEK. To do so, generate a new KEK in the KMS. The SDS encryption service for Google Workspace will automatically import this KEK as the active key when the keys are refreshed. Older keys will be kept for unwrap operations.

The Thales KMS does not allow more than 200 KEKs to be managed. Please do not exceed this limit.

If the KEK list refresh operation fails, the list of current keys will be kept and service will not be disrupted. The SDS encryption service for Google Workspace will refresh the key list again when a periodic or one-off refresh operation is triggered.

10. Customizing the authorization rules

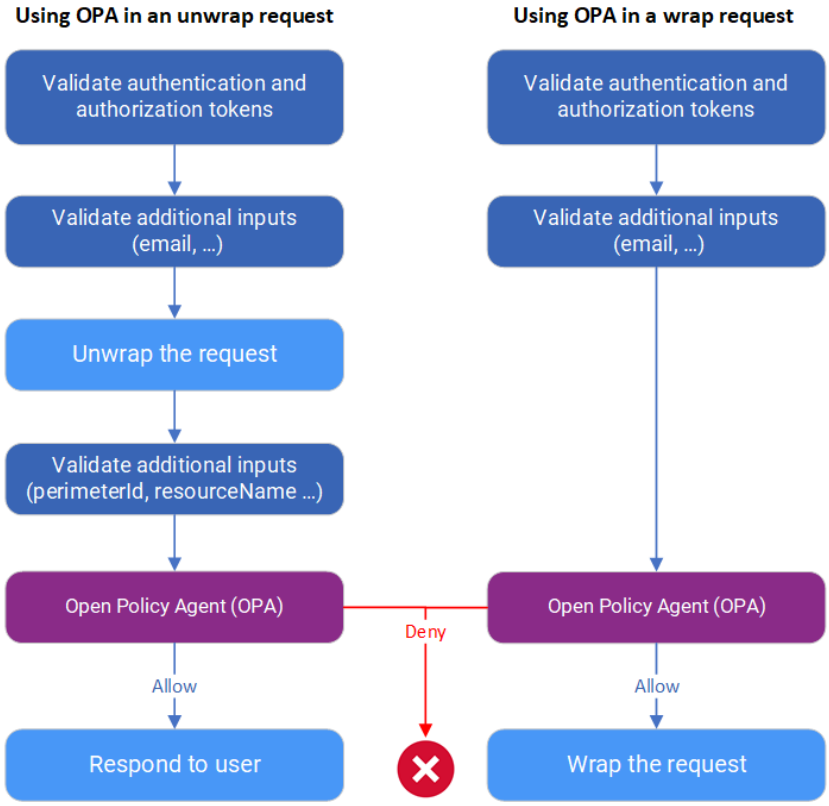
You can customize the rules that allow or deny a request to the SDS encryption service for Google Workspace, using Open Policy Agent (OPA). The policy evaluates the request inputs. If the request is forbidden, the access is denied and the "403 Forbidden" error is returned.

EXAMPLE
You can define a policy allowing access to the SDS encryption service for Google Workspace only to users from the *stormshield.eu* domain.

You can add an OPA policy to the following API routes. If you specify rules for other routes, they will be ignored.

- wrap, unwrap, privilegedwrap, privilegedunwrap, rewrap, certs, digest, wrapprivatekey, privatekeydecrypt, privilegedprivatekeydecrypt, and privatekeysign.

The diagram below indicates at which stage of the requests the OPA policy is applied for the "wrap" and "unwrap" requests.



10.1 Defining an OPA policy

If using SDS encryption service for Google Workspace on several instances, you must apply the following procedure on all instances.

1. Edit the two files needed to define a policy:

File name	Location	Description
policy.wasm	/etc/stormshield/cse/	• This file defines the policy rules, based on a request inputs • It is generated from a rego file, • Use the OPA command line tool to generate it, • Use The rego Playground to develop and test a policy.
policy.data.json	/etc/stormshield/cse/	• This file contains data that can be referenced in the rego file. • It makes it possible to add variables to the *policy.wasm* file so that you do not to have to recompile the file each time you modify it.

In RPM mode, both these files already exist in */etc/stormshield/cse*. They define a default policy which allows all requests. If you want to create your own policy, customize these files using the inputs and the examples provided in the following sections.
To disable the customized policy, remove one of the *policy.wasm* or *policy.data.json* files. The service will then start without any policy being applied. A log will be issued to indicate that the policy is disabled.
In Docker mode, these default policy files are provided along with the Docker image. You can place them in the folder containing your configuration files, or define your own. If they are not present, the SDS encryption service for Google Workspace starts normally and issues a log to indicate that the policy has been disabled.

2. Run the restart command to take into account the modified *policy.wasm* and *policy.data.json* files. If one of the files is not valid, the service does not start and a log is issued. For more information, refer to the section Logs relating to the application of an OPA policy.

The *policy.wasm* and *policy.data.json* files are optional. If one of the files is not present, the service starts and no policies are applied. A log is issued to indicate that the policy is disabled.

10.1.1 Defining an OPA policy for SDS CryptoAPI

The files required to use an OPA policy for CryptoAPI do not exist by default and are as follows:

- */etc/stormshield/cse/policy-kmaas.wasm*
- */etc/stormshield/cse/policy-kmaas.data.json*

Their format and configuration are the same as the standard policy files.

10.2 Inputs relating to all API routes

The following inputs can be used by the Google Workspace administrator to create customized rules to access the SDS encryption service for Google Workspace. Use them to filter the application of policies.

You can use these inputs in the custom policy.

Input	Description	Source of the input
endpoint	API routes called: "wrap", "unwrap", "privilegedwrap", "privilegedunwrap", "rewrap", "certs", "wrapprivatekey", "privilegedprivatekeydecrypt", "digest", "privatekeydecrypt", "privatekeysign".	URL of the request
tenantId	unique identifier of a tenant in UUID format. Example: 2363615f-5b08-4119-afdb-fad3f5f3f420.	URL of the request
perimeterId	Optional. Value relating to the key used to wrap a DEK.	Data encapsulated in the DEK within the request body
resourceName	Unique identifier of the object encrypted by the DEK.	Data encapsulated in the DEK within the request body
reason	String containing a JSON object. Not currently used.	Request body

10.3 Inputs specific to the wrap and unwrap API routes

Input	Description	Source of the input
authentication.email	User's email address. In UTF8 format, in lower case.	JWT authentication token provided by the IDP.
authentication.googleEmail	Optional If the email field is not a Google Workspace address belonging to the domain, googleEmail is the user's email address. It takes priority over the email field. In UTF8 format, in lower case.	JWT authentication token provided by the IDP.
authentication.iss	Entity which has created and signed the token.	JWT authentication token provided by the IDP.
authentication.aud	Corresponds to the audience for which the token was issued. Example: ['cse-authorization', 'cse-authorization1']	JWT authentication token provided by the IDP.
authentication.iat	Date when the token was issued. In timestamp format (integer) Example: 1677679386	JWT authentication token provided by the IDP.

Input	Description	Source of the input
authentication.exp	Date when the token expires. In timestamp format (integer) Example: 1677679386	JWT authentication token provided by the IDP.
authentication.customClaims	Optional. Custom claims provided par the IDP. SeeExample of policy implementation.	JWT authentication token provided by the IDP.
authorization.iss	Entity which has created and signed the token. Can be used to differentiate the various Google Workspace applications, or to migrate from a KACLS to another. Example: “gsuitecse-tokenissuer-drive@system.gserviceaccount.com” See Authorization settings.	JWT authorization token provided by Google
authorization.role	Role of the authorized user. • "writer" for wrap and unwrap, • "reader" for unwrap	JWT authorization token provided by Google
authorization.aud	Corresponds to the audience for which the token was issued. Example: ['cse-authorization', 'cse-authorization1']	JWT authorization token provided by Google
authorization.exp	Date when the token expires. In timestamp format (integer) Example: 1677679386	JWT authorization token provided by Google
authorization.iat	IssuedAt, date when the token was issued. In timestamp format (integer) Example: 1677679386	JWT authorization token provided by Google
authorization.emailType	Identifies the origin of the e-mail address in the token. Prescribed values: • "google" for Google accounts (default value), • "google-visitor" for Google-verified accounts, • "customer-idp" for IDP accounts.	JWT authorization token provided by Google
contentType	Cryptographic content type. Prescribed value: "dek"	Cryptographic component used.

10.4 Inputs specific to the privilegedwrap and privilegedunwrap API routes

Input	Description	Source of the input
authentication.email	User's email address. In UTF8 format, in lower case.	JWT authentication token provided by the IDP.

Input	Description	Source of the input
authentication.googleEmail	Optional If the email field is not a Google Workspace address belonging to the domain, googleEmail is the user's email address. It takes priority over the email field. In UTF8 format, in lower case.	JWT authentication token provided by the IDP.
authentication.iss	Entity which has created and signed the token.	JWT authentication token provided by the IDP.
authentication.aud	Corresponds to the audience for which the token was issued. Example: ['cse-authorization', 'cse-authorization1']	JWT authentication token provided by the IDP.
authentication.iat	Date when the token was issued. In timestamp format (integer) Example: 1677679386	JWT authentication token provided by the IDP.
authentication.exp	Date when the token expires. In timestamp format (integer) Example: 1677679386	JWT authentication token provided by the IDP.
authentication.customClaims	Optional. Custom claims provided par the IDP. SeeExample of policy implementation.	JWT authentication token provided by the IDP.
contentType	Cryptographic content type. Prescribed value: "dek"	Cryptographic component used.

10.5 Inputs specific to the rewrap API route

Input	Description	Source of the input
originalKacIsUrl	URL of the initial KACLS for a migration.	Request body
authorization.email	User's email address. In UTF8 format, in lower case.	JWT authorization token provided by Google
authorization.role	Role of the authorized user: "migrator"	JWT authorization token provided by Google
authorization.resourceName	Same as resourceName, but originates from the authorization token.	JWT authorization token provided by Google
authorization.kacIsUrl	URL of the KACLS.	JWT authorization token provided by Google

Input	Description	Source of the input
authorization.iss	Entity which has created and signed the token. Can be used to differentiate the various Google Workspace applications, or to migrate from a KACLS to another. Example: “gsuitecse-tokenissuer-drive@system.gserviceaccount.com” See Authorization settings.	JWT authorization token provided by Google
authorization.aud	Corresponds to the audience for which the token was issued. Example: ['cse-authorization', 'cse-authorization1']	JWT authorization token provided by Google
authorization.exp	Date when the token expires. In timestamp format (integer) Example: 1677679386	JWT authorization token provided by Google
authorization.iat	IssuedAt, date when the token was issued. In timestamp format (integer) Example: 1677679386	JWT authorization token provided by Google
contentType	Cryptographic content type. Prescribed value: "dek"	Cryptographic component used.

10.6 Inputs specific to the certs API route

Only *endpoint* and *tenantId* inputs are available for the *certs* API route. For more information, refer to the section Inputs relating to all API routes.

10.7 Inputs specific to the digest API route

Input	Description	Source of the input
authorization.email	User’s email address. In UTF8 format, in lower case.	JWT authorization token provided by Google
authorization.role	Role of the authorized user: "check"	JWT authorization token provided by Google
authorization.resourceName	Same as resourceName, but originates from the authorization token.	JWT authorization token provided by Google
authorization.kaclsUrl	URL of the KACLS.	JWT authorization token provided by Google

Input	Description	Source of the input
authorization.iss	Entity which has created and signed the token. Can be used to differentiate the various Google Workspace applications, or to migrate from a KACLS to another. Example: “gsuitecse-tokenissuer-drive@system.gserviceaccount.com” See Authorization settings.	JWT authorization token provided by Google
authorization.aud	Corresponds to the audience for which the token was issued. Example: ['cse-authorization', 'cse-authorization1']	JWT authorization token provided by Google
authorization.exp	Date when the token expires. In timestamp format (integer) Example: 1677679386	JWT authorization token provided by Google
authorization.iat	IssuedAt, date when the token was issued. In timestamp format (integer) Example: 1677679386	JWT authorization token provided by Google
contentType	Cryptographic content type. Prescribed value: "dek"	Cryptographic component used.

10.8 Inputs specific to the privatekeydecrypt and privatekeysign API routes

Input	Description	Source of the input
algorithm	Algorithm used to encrypt the private key.	Data within the request body.
authentication.email	User's email address. In UTF8 format, in lower case.	JWT authentication token provided by the IDP.
authentication.googleEmail	Optional If the email field is not a Google Workspace address belonging to the domain, googleEmail is the user's email address. It takes priority over the email field. In UTF8 format, in lower case.	JWT authentication token provided by the IDP.
authentication.iss	Entity which has created and signed the token.	JWT authentication token provided by the IDP.
authentication.aud	Corresponds to the audience for which the token was issued. Example: ['cse-authorization', 'cse-authorization1']	JWT authentication token provided by the IDP.

Input	Description	Source of the input
authentication.iat	Date when the token was issued. In timestamp format (integer) Example: 1677679386	JWT authentication token provided by the IDP.
authentication.exp	Date when the token expires. In timestamp format (integer) Example: 1677679386	JWT authentication token provided by the IDP.
authentication.customClaims	Optional. Custom claims provided par the IDP. SeeExample of policy implementation.	JWT authentication token provided by the IDP.
authorization.email	User's email address. In UTF8 format, in lower case.	JWT authorization token provided by Google
authorization.role	Role of the authorized user. • "decrypter" for privatekeydecrypt, • "signer" for privatekeysign	JWT authorization token provided by Google
authorization.resourceName	Same as resourceName, but originates from the authorization token.	JWT authorization token provided by Google
authorization.perimeterId	Same as perimeterId, but originates from the authorization token.	JWT authorization token provided by Google
authorization.kaclsUrl	URL of the KACLS.	JWT authorization token provided by Google
authorization.iss	Entity which has created and signed the token. Can be used to differentiate the various Google Workspace applications, or to migrate from a KACLS to another. Example: "gsuitecse-tokenissuer-drive@system.gserviceaccount.com" See Authorization settings.	JWT authorization token provided by Google
authorization.aud	Corresponds to the audience for which the token was issued. Example: ['cse-authorization', 'cse-authorization1']	JWT authorization token provided by Google
authorization.exp	Date when the token expires. In timestamp format (integer) Example: 1677679386	JWT authorization token provided by Google
authorization.iat	IssuedAt, date when the token was issued. In timestamp format (integer) Example: 1677679386	JWT authorization token provided by Google

Input	Description	Source of the input
authorization.spkiHashBase64	SPKI hash in base 64 to validate authorization.	JWT authorization token provided by Google
authorization.spkiHashAlgorithm	Encryption algorithm used to produce the SPKI hash.	JWT authorization token provided by Google
authorization.messageId	Optional. Value relating to the encryption key used during a wrapprivatekey operation.	JWT authorization token provided by Google
contentType	Cryptographic content type. Prescribed values: "private-key-pem" or "private-key-name"	Cryptographic component configured.

10.9 Inputs specific to the wrapprivatekey and privilegedprivatekeydecrypt API routes

Input	Description	Source of the input
authentication.email	User's email address. In UTF8 format, in lower case.	JWT authentication token provided by the IDP.
authentication.googleEmail	Optional If the email field is not a Google Workspace address belonging to the domain, googleEmail is the user's email address. It takes priority over the email field. In UTF8 format, in lower case.	JWT authentication token provided by the IDP.
authentication.iss	Entity which has created and signed the token.	JWT authentication token provided by the IDP.
authentication.aud	Corresponds to the audience for which the token was issued. Example: ['cse-authorization', 'cse-authorization1']	JWT authentication token provided by the IDP.
authentication.iat	Date when the token was issued. In timestamp format (integer) Example: 1677679386	JWT authentication token provided by the IDP.
authentication.exp	Date when the token expires. In timestamp format (integer) Example: 1677679386	JWT authentication token provided by the IDP.
authentication.customClaims	Optional. Custom claims provided par the IDP. SeeExample of policy implementation.	JWT authentication token provided by the IDP.
contentType	Cryptographic content type. Prescribed values: "private-key-pem" or "private-key-name"	Cryptographic component configured.

10.10 Inputs specific to the SDS CryptoAPI encrypt and decrypt routes

As a Google Workspace administrator, you can use these inputs to create custom rules to access the CryptoAPI SDS encryption service, for example to filter the application of policies. Use some of these inputs in the created policy.

Input	Description	Source of the input
endpoint	API routes called: "encrypt", "decrypt"	URL of the request
tenantID	unique identifier of a tenant in UUID format. Example: 2363615f-5b08-4119-a5bd-fad3f5f3f420	URL of the request
authentication.iss	Entity which has created and signed the token.	JWT authentication token provided by the IDP.
authentication.aud	Corresponds to the audience for which the token was issued. Example: ['cse-authorization', 'cse-authorization1']	JWT authentication token provided by the IDP.
authentication.iat	Date when the token was issued. In timestamp format (integer) Example: 1677679386	JWT authentication token provided by the IDP.
authentication.exp	Date when the token expires. In timestamp format (integer) Example: 1677679386	JWT authentication token provided by the IDP.
authentication.customClaims	Optional. Custom claims provided par the IDP. SeeExample of policy implementation.	JWT authentication token provided by the IDP.
contentType	Cryptographic content type. Prescribed values: "private-key-pem" or "private-key-name"	Cryptographic component configured.

10.11 Example of policy implementation

In this example, a rule is added which allows only the users present in the *authorizedUsers* list to perform a request for the 'unwrap' route used for Google Drive.

In the *policy.rego* file:

- *input* refers to the data provided by the SDS encryption service for Google Workspace to the policy.
- *data* refers to the data in the *policy.data.json* file.

Once the *policy.rego* file has been compiled into *policy .wasm* using via the OPA tool, you can add or remove authorized users by updating the content of the *authorizedUsers* field in the *policy.data.json* file.

In this example, a user with an email address in the authentication token 'user1@test.com' or 'user2@test.com' will be allowed to use the 'unwrap' route to decrypt a Google Drive document, whereas other users will not be allowed to do so. Other users will be allowed to use the "unwrap" route if it does not imply Google Drive.

10.11.1 policy.rego file

```
package cse

import future.keywords.if

import future.keywords.in

# ------------------------------------------------------

# Deny by default
default allow := false

# Allow all other endpoints
allow if {
input.endpoint != "unwrap"
}

# Allow access to unwrap endpoint if not concerning drive application
allow if {
input.endpoint == "unwrap"
input.authorization.iss != "gsuitecse-tokenissuer-
drive@system.gserviceaccount.com"
}

# Allow access to unwrap concerning drive, only for authorizedUsers
allow if {
input.endpoint == "unwrap"
input.authorization.iss == "gsuitecse-tokenissuer-
drive@system.gserviceaccount.com"
input.authentication.email in data.authorizedUsers
}
```

10.11.2 policy.data.json file

```
{
"authorizedUsers": ["user1@test.com", "user2@test.com"]
}
```

10.12 Using custom claims

Authentication and authorization tokens may contain user data (claims) that are not required by the SDS encryption service for Google Workspace.

Such data is placed in a “customClaims” object and can be used in an OPA policy file.

For example, the SDS encryption service for Google Workspace may get an authentication token of the following form:

```
{
      iss: 'issuer-authentication',
      aud: 'cse-authentication' ,
      exp: 1731599885,
      iat: 1728917885,
        email: 'user@domain.com',
        user_age: 23
}
```

The "user_age" property is not required by the SDS encryption service for Google Workspace. However, it will be transmitted to OPA in a “customClaims” object, as follows:

```
{
      iss: 'issuer-authentication',
      aud: 'cse-authentication' ,
      exp: 1731599885,
      iat: 1728917885,
        email: 'user@domain.com',
        customClaims: {
              user_age: 23
        }
}
```

You can use the "user_age" property in the *policy.rego* file, using the *customClaims.{key}* syntax. In the example below, access is restricted to users over the age of 18:

```
package cse
import future.keywords.if

# Deny by default
default allow := false

# Allow access if age is more than 18
allow if {
  input.authentication.customClaims.user_age >= 18
}
```

11. Running the SDS encryption service for Google Workspace

11.1 In RPM mode

1. Run the SDS encryption service for Google Workspace as a systemd service with the root user using the following command:
   ```
   # systemctl start cse
   ```
2. Check the status of the service:
   ```
   # systemctl status cse
   ```
3. Enable the autorun of the service when the machine starts:
   ```
   # systemctl enable cse
   ```

The SDS encryption service for Google Workspace uses all available CPU cores. This parameter cannot be configured.

11.2 In Docker mode

Follow all the steps below to run the SDS encryption service for Google Workspace in Docker mode.

11.2.1 Loading the Docker image

Load the image of the SDS encryption service for Google Workspace in Docker using the following command:

```
docker load --input stormshield-kmaas-<version>.tar
```

11.2.2 Requirements

Execution UID/GUID

The SDS encryption service for Google Workspace runs with the node user (UID/GID 1000) in the container. To ensure that your application runs correctly and securely, be sure to specify the execution UID/GID correctly. Below is an example of a command:

```
docker run -u 1000:1000 stormshield/kmaas:<version>
```

File access

- You must allow containers to access your configuration files (i.e., *keks.json*, *config.json*, OPA files, certificate files and private keys).
- For *config.json* and *keks.json* files, Stormshield makes the following recommendations:
  - If you have several instances of the SDS encryption service for Google Workspace, expose a single file to the various containers of the application, as they must be identical on all instances,
  - Mount them read-only, as they will never be modified by the SDS encryption service for Google Workspace.
    Below is an example of a command with a read-only folder containing configuration files:
    ```
    docker run -v /my-kmaas-config-folder:/etc/stormshield/cse:ro
    stormshield/kmaas:<version>
    ```

- Sensitive files (i.e., *keks.json*, private keys) must be managed by secure mechanisms provided by your orchestrator. Refer to the documentation of your orchestrator.

Network traffic redirection

The service listens on the port defined in the *config.json* file (3000 by default) in the container. Below is an example of a command that forwards host port 443 to port 3000:

```
docker run -p 443:3000 my-image
```

Refer to your orchestrator's documentation to set up port forwarding in a production environment.

Access to environment variables

Containers must have access to the environment variables mentioned in this administration guide. Below is the command for declaring an environment variable:

```
docker run -e MY_VARIABLE=my-variable-value
stormshield/kmaas:<version>
```

Refer to your orchestrator's documentation to set the environment variables in a production environment.

11.2.3 Starting a container

Example of a Docker command to start a SDS encryption service for Google Workspace container:

```
docker run -v /my-kmaas-config-folder:/etc/stormshield/cse
-p 443:3000 -u 1000:1000 stormshield/kmaas:4.4.0.2427
```

12. Configuring TLS ciphers

When the SDS encryption service for Google Workspace is configured in HTTPS, it uses NodeJS which depends on OpenSSL for cryptographic operations.

In RPM mode, by default, the service starts with the following cipher list:

TLS 1.3

- TLS_AES_256_GCM_SHA384
- TLS_AES_128_GCM_SHA256
- TLS_AES_128_CCM_SHA256
- TLS_CHACHA20_POLY1305_SHA256

TLS 1.2

- ECDHE-ECDSA-AES256-GCM-SHA384
- ECDHE-RSA-AES128-GCM-SHA256
- ECDHE-RSA-AES256-GCM-SHA384

In Docker mode, the default cipher list is the same as NodeJS.

To communicate more securely when using the SDS encryption service for Google Workspace, you can restrict the list of ciphers allowed during TLS operations.

The ciphers used are important security elements. Refer to ANSSI documentation SDE-NT-35 on TLS ciphers.

Note that if you use SELinux to secure the machines, the list of TLS algorithms allowed on the machine may change. This may prevent the SDS encryption service for Google Workspace from starting or cause incompatibility with external resource retrieval. In this case, you must adjust the list of algorithms allowed by the SDS encryption service for Google Workspace by following the procedure below.

12.1 Modifying the TLS cipher list in RPM mode

1. Add the *cipher_list.conf* file in the */etc/systemd/system/cse.service.d* directory.
2. Add the following lines in this file:
   `[Service]`
   `Environment=NODE_OPTIONS=--tls-cipher-list=#CUSTOM_CIPHER_LIST#`
   - where -
   `#CUSTOM_CIPHER_LIST#` represents the list of the desired ciphers, separated by ":".

   EXAMPLE
   [Service]
   Environment=NODE_OPTIONS=--tls-cipher-list=TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:

   The cipher format must match the following rules:

   - The SDS encryption service for Google Workspace does not support the following cryptographic suites, nor the "!", "+", and "-" operators.
   - You must use the OpenSSL cipher format, and not the standard format.
     To list ciphers with both their standard names and OpenSSL names, run the command `openssl ciphers -stdname`.
     To convert a standard cipher name to the OpenSSL format, run the command `openssl ciphers -convert STANDARD_CIPHER_NAME_TO_CONVERT`.
3. If the list contains valid but not recommended ciphers, a warning log is issued. If a cipher is unknown, an error is issued and the service does not start.

12.2 Modifying the list of TLS ciphers in Docker mode

- Declare the NODE_OPTIONS environment variable:
  `NODE_OPTIONS=--tls-cipher-list=<liste-of-tls-algorithms>`

  EXAMPLE
  `docker run -e "NODE_OPTIONS=--tls-cipher-list=TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384" stormshield/kmaas:<version>`

13. Checking system health

After you have installed and run the SDS encryption service for Google Workspace, check that it is running correctly.

13.1 Checking via the *status* API

1. Use the *status* API route:

```
curl -H "Origin: <origin_url>" <my-cse-full-url>/status
```

where:

Parameter	Value
<origin_url>	The SDS encryption service for Google Workspace enforces across-origin resource sharing (CORS) rule to guarantee that requests genuinely originate from the Google API. This rule makes it possible to verify the *origin* HTTP header. The origins that are allowed are all URLs of https://*.google.com type, for example: • https://client-side-encryption.google.com • https://admin.google.com • Requests containing an incorrect *origin* header are rejected. For more information, refer to the Google documentation Connect to your identity provider for client-side encryption.
<my-cse-full-url>	Full URL specified for th external key service. For more information, refer to Specifying the External key service.

EXAMPLE

```
curl -H "Origin: https://client-side-encryption.google.com"
https://cse.example.com/api/v1/a4670b0-4bc11-4290-a5bd-
498c2e1fb0b/status
```

2. If the system is running correctly, the *status* API return must be in the following form:

```
{"server_type":"KACLS","vendor_id":"SDS for Google
Workspace","version":"4.3.0.2354","name":"name of my
CSE","operations_supported":
["wrap","unwrap","digest","rewrap","privilegedwrap","privilegedunw
rap","wrapprivatekey","privatekeysign","privatekeydecrypt","privil
egedprivatekeydecrypt"]}
```

13.2 Checking via the *health* API

This API returns only a limited amount of information about the running of the SDS encryption service for Google Workspace and does not require any CORS in the HTTP header.

1. Use the *health* API route:

```
curl https://<my-cse-url>/health
```

2. If the system is running correctly, the return must be in the following form:

```
{}
```

14. Backing up and restoring the SDS encryption service for Google Workspace files

In RPM mode, Stormshield recommends that you deploy SDS encryption service for Google Workspace in a cluster to improve performance and limit the impact if a failure occurs. However, some of the files used by the service and the configuration of the server must be backed up if you have deployed the solution via an RPM.

In Docker mode, you must configure your orchestrator to guarantee data persistence. Implement a regular file backup strategy, as well as secure file storage on a separate infrastructure.

14.1 Backing up SDS encryption service for Google Workspace files

- Back up the files below **every time changes are made**:

Name	Description	Level	Consequences if file is lost
config.json	File that describes the configuration of the SDS encryption service for Google Workspace server	Moderate	Configuration must be rebuilt
keks.json	**File containing all the KEKs**	**Very high**	**Users' encrypted data cannot be decrypted**
HTTPS certificates	Files used for the HTTPS connection	Moderate	Configuration must be rebuilt
KMS certificates	Files used for the KMS connection	Moderate	Configuration must be rebuilt

14.2 Restoring the files in RPM mode

If any node of the cluster fails, follow the procedure below to restore files on each node:

1. Disconnect the node from the cluster.
2. Reconfigure the Red Hat instance if necessary.
3. Reinstall the service if necessary.
4. Deploy the backed up files:
   - config.json
   - keks.json
   - Certificates for HTTPS
   - Certificates for KMS
5. Restart the service and check whether it runs.
6. Reconnect the node(s) of the cluster.

14.3 Restoring the files in Docker mode

In Docker mode, the various configuration files are made available to containers via a volume. If one of the containers fails, recreate the SDS encryption service for Google Workspace container

so that the configuration files are automatically taken into account.

For more information, please refer to your orchestrator documentation.

15. Decrypting files and emails

To decrypt files and emails encrypted by the SDS encryption service for Google Workspace, you can use Google's decryption utility *decrypter.exe* (Beta version):

1. As the super-administrator of the Google Workspace domain, export encrypted data using the export tool or Google Vault. For more information, refer to the Google documentation. The export operation is only possible once every 30 days.
2. Decrypt encrypted data using Google's *decrypter.exe* utility (Beta version). For more information, refer to the Google documentation.

16. Migrating an external key service to another

If you have an external third-party key service (also known as a KACLS) and you want to replace it with the SDS encryption service for Google Workspace, follow the Google migration procedure. During this procedure, you will be able to retrieve all your old encrypted data and re-encrypt it to the SDS encryption service for Google Workspace.

Before launching the migration, you must choose a backup key service to which old data will also be encrypted. Google will launch two parallel migrations: encrypted data will be migrated to the SDS encryption service for Google Workspace and to the backup key service.

The SDS encryption service for Google Workspace must be configured before migration is enabled in the Google Admin interface.

16.1 Configuring migration in the SDS encryption service for Google Workspace

1. Generate a pair of RSA keys without password in PEM format with the tool of your choice. For example with OpenSSL and the following commands:
```
openssl genpkey -algorithm RSA -out private_key.pem -pkeyopt rsa_keygen_bits:4096
openssl rsa -pubout -in private_key.pem -out public_key.pem
```
2. Encode the generated private key in base64 with the following command:
```
openssl base64 -A -in private_key.pem -out private_key_base64.txt
```
3. Fill in the `"migration"` section in the *config.json* file:
   - In the `"format"` field, specify the pem key format,
   - In the `"key"` field, enter the private key generated in base64,
   - In the `"kacls_urls"` field, add all the key services with which the SDS encryption service for Google Workspace must communicate, including the backup key service.

4.
```
"migration": {
        "enabled": true,
        "kaclstokacls_token": {
                "kid": "key_identifier",
                "format": "pem",
                "key": "a_private_key",
                "duration": 3600
        },
        "acls": {
                "kacls_urls": ["https://kacls_1", "https://kacls_2"]
        }
    }
```

   For more information, refer to Configure the SDS encryption service for Google Workspace, in particular the section on Migration.

5. In the *config.json* file, fill in the `authorization` section with information on the migration, as shown in Authorization parameters.

16.2 Adding the SDS encryption service for Google Workspace in Google

1. In the Google Workspace administration console, add the SDS encryption service for Google Workspace as a new key service
2. Select an operational backup key service as well, to which old data will also be encrypted. This backup key service can be the one that you wish to replace.

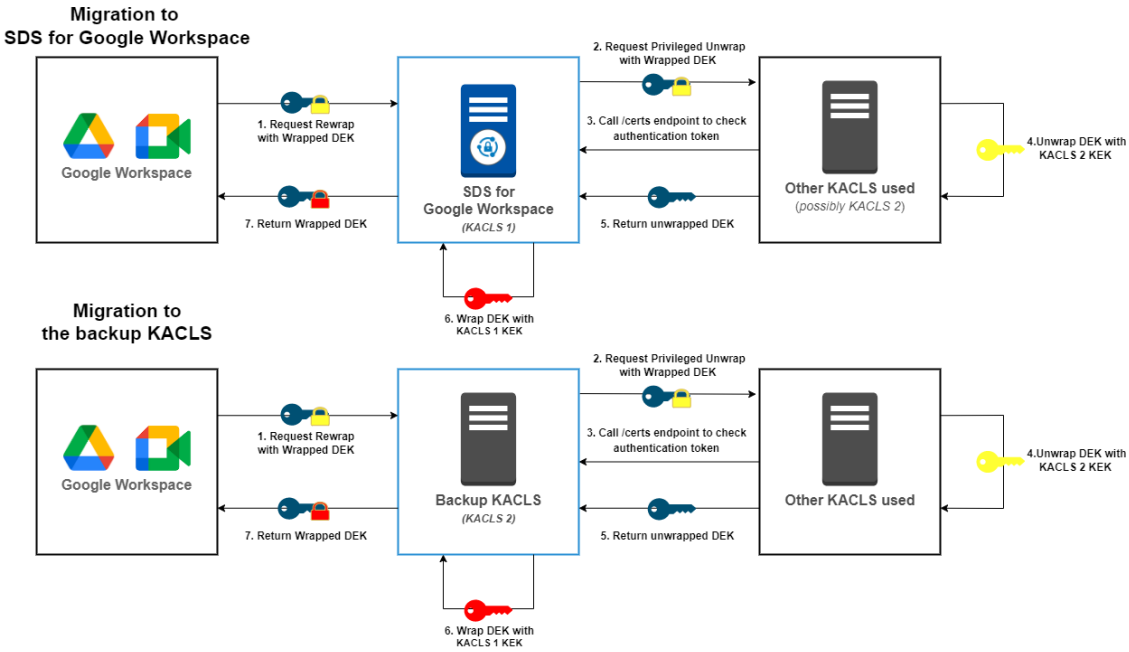
For more information, refer to the Google documentation.

16.3 Enabling key service migration in Google

1. In the Google Workspace administration console, select the SDS for Google Workspace key service.
2. Enable key service migration.
   Google will launch the migration. Refer to the corresponding logs in the SDS encryption service for Google Workspace. See Logs relating to the rewrap API route and Logs relating to the privilegedunwrap API route .

For more information, refer to the Google documentation.

The diagram below shows the various stages of the migration to the SDS encryption service for Google Workspace and to the backup key service. In this diagram, the term 'KACLS' refers to a key service.



NOTE
The tokens generated by the SDS encryption service for Google Workspace and provided to another KACLS in the Privileged Unwrap request are signed using the RS256 algorithm.

16.4 Using the backup key service other than for migration service

The backup key service guarantees access to content if an issue arises with the main key service. It is mandatory in a migration, but you can also use it to test a new key service, for example. In this case, encrypting data to the backup key service is still considered a migration, and you must configure the SDS encryption service for Google Workspace accordingly:

- Fill in the `"migration"` section in the *config.json* file. In the `"kacls_urls"` field, add the backup key service.

```
"migration": {
        "enabled": true,
        "kaclstokacls_token": {
                "kid": "key_identifier",
                "format": "pem",
                "key": "a_private_key",
                "duration": 3600
        },
        "acls": {
                "kacls_urls": ["https://backup_kacls"]
        }
    }
```

17. SDS CryptoAPI

The SDS CryptoAPI feature provides two API routes, *crypto/encrypt* and *crypto/decrypt*, which encrypts and decrypts data without Google being involved.

To enable SDS CryptoAPI, fill in the section *tenant - kmaas* in the *config.json* configuration file:

Here is an example of how to configure the SDS Crypto API:

```
kmaas {
       "user_authentication": [

         {
           "discovery_uri": "https://myIDP/.well-known/openid-configuration",
           "client_id": "kmaas_client_id"
         }
    ]
  }
}
```

For more information, refer to Installing SDS encryption service for Google Workspace.

Routes use this tenant's keys in the *keks.json* file for data encryption.

The mandatory fields in the payload of the JWT token used to authenticate routes are “iss”, “aud”, “exp” and “iat”.

You can define custom policies for these routes. For more information, see Customizing the authorization rules.

WARNING
SDS CryptoAPI is currently in its alpha version. Stormshield does not guarantee that data encrypted with this version can be decrypted with a later version of the feature.

A Swagger API documentation is available for these routes. Please contact Stormshield to consult it.

18. Managing logs

The SDS encryption service for Google Workspace generates logs for every operation, making it possible to trace all operations performed and potential issues. Logs are generated in JSON format. In RPM mode, the logs are managed by the *systemd* service.

There are two different log formats:

- Refer to the section Understanding the contents of logs for more information on logs in the old format.
- Refer to the section Understanding the new log format and to the SDS encryption service for Google Workspace Log Guide for more information on logs in the new format.

18.1 Logging requirements

Logging is a technical activity essential to the security of information systems. To meet the logging requirements for the SDS encryption service for Google Workspace, you must :

- Follow the security recommendations for logging systems issued by ANSSI in their document *ANSSI-PA-012*,
- Set up a partition dedicated to logs, with restricted access rights,
- Formalize and implement a log rotation policy for all logging system equipment.
- In Docker mode, follow the recommendations issued by ANSSI in their document ANSSI-FT-082 relating to Docker container deployment.

18.2 Accessing logs

18.2.1 In RPM mode

Logs are stored in the *systemd* standard folder. The following commands can be used to show and export logs.

Command	Description
cat /var/log/messages	Shows all logs of all services.
journalctl -u cse	Shows all logs relating to the SDS encryption service for Google Workspace.
journalctl -u cse -b	Shows all logs relating to the SDS encryption service for Google Workspace since the last time the machine was started.
journalctl -u cse > cse.log	Exports all logs relating to the SDS encryption service for Google Workspace into a *cse.log* file.

18.2.2 In Docker mode

Access the logs of the SDS encryption service for Google Workspace by running the Docker standard command:

```
docker logs <containerID> -f
```

18.3 Understanding the contents of logs

The SDS encryption service for Google Workspace generates two types of logs:

- Configuration logs, which provide information about the status and running of the SDS encryption service for Google Workspace. Such logs belong to either the *server_status*, *kmip_status*, or *kmip_decrypt* categories.
- API logs, which provide information on API calls to the SDS encryption service for Google Workspace. Such logs belong to the *api_route* category.

The following tables describe the fields for each log type.

In these tables:

- **JWT** means JSON web token,
- **KACLS** is the name that Google gives to the SDS encryption service for Google Workspace (i.e., Key Access Control List Service).

18.3.1 Generic log fields

The fields described in the table below appear in all SDS encryption service for Google Workspace logs.

Field	Type Description	Type	Examples
severity	Log severity: • *info*: standard severity used for most logs, • *warning*: indicates that the operation was successful but generated a warning, • *critical*: indicates that the operation ended in an error.	String	Prescribed values: "info", "warning", "critical"
kind	Log group to which the log belongs, for example: • domain: logs concerning SDS encryption service for Google Workspace business operations This field is only present in logs in the new format.	Character string	Prescribed values: "domain", "system"
category	Log category, for instance: • *server_status*: configuration log that provides information about the startup of the SDS encryption service for Google Workspace and its status, • *api_route*: log of the API that provides information about API routes (e.g., "/wrap").	String	Prescribed values: "server_status", "api_route", "kmip_status", "kmip_extract", "kek_reading", "policy_status", "jwks", "open_id", "keks", "kek"
action	Event that occurred, for instance for the "keks" category: • *setup*: KEK persistence and refresh mode has been configured. • *load*: A KEK retrieval has been triggered. This field is only present in logs in the new format.	Character string	
version	Current version of the service.	String	"4.3.0.2354"

Field	Type Description	Type	Examples
timestamp	Date and time at which the log was created. In ISO-8601 format.	String	"2021-03-05T16:53:28Z"
hostname	Host name.	String	"cseserver13"
process	Node.js worker PID.	Integer	4031
tenant_id	Tenant identifier.	String (uuid v4)	"025f02fe-bee2-444b-bf76-b5ead30327c0"
errors	Optional	Array	

18.3.2 Logs relating to the status of the service (start, stop, failure)

The fields described below belong to logs that contain errors which appear when the SDS encryption service for Google Workspace starts.

Field	Description	Type	Examples
name	Service name.	String	"Stormshield SDS CSE #13"
status	Server status.	String	Prescribed values: "started", "stopped", "failure"
host	Server address.	String	"172.16.16.240", "cse13.stormshield.eu"
port	Port on which the SDS encryption service for Google Workspace listens.	Integer	4333
persistence_type	Persistence mode of KEK data.	String	Prescribed values: "json_file", "kms"
https	Server startup mode: HTTP or HTTPS.	Boolean	Prescribed values: "true" (HTTPS), "false" (HTTP)
cache	Activation status of the cache.	Boolean	Prescribed values: "true", "false"
proxy	Information relating to the activation of the proxy It contains the information below:	Object	Mandatory
	proxy.enabled: Activation status of the proxy	Boolean	Prescribed values: "true", "false"
	proxy.https_url: URL used for the HTTPS proxy	String	"http://myhttpsproxyurl"

18.3.3 Logs relating to configurations retrieved from the identity provider

The log fields described below provide information on the retrieval status of configurations. These logs are issued when the service is started.

These logs can be generated when data is retrieved from the application cache or when a request is made.

If you use *discoveryUri* to retrieve the identity provider configuration, the logs displayed will be as follows:

Field	Type Description	Type	Examples
category	Log category.	String	Prescribed value: "open_id"
status	Status of the request.	String	Prescribed values: "fetching", "unreachable", "fetch success"
discovery_uri	URI used for retrieving the configuration.	String	"https://localhost:3001/static/one-login/.well-known/openid-configuration"
token_type	Function of the token to retrieve.	String	Prescribed values: "authorization", "authentication"
role	Role associated with the token to retrieve.	String	Prescribed values: "user", "admin", "wrapprivatekey"
source	Source of the configuration used.	String	Prescribed values: "local_configuration", "remote_well_known_cse_configuration"

If you use *discoveryUri* to retrieve the identity provider configuration, the logs displayed will be as follows:

Field	Type Description	Type	Examples
category	Log category.	String	Prescribed value: "jwks"
status	Status of the request.	String	Prescribed values: "checking reachability", "unreachable", "reachable"
jwks_uri	JWKS used for retrieving the configuration.	String	"https://localhost:3001/static/one-login/.well-known/jwks.json"
token_type	Function of the token to retrieve.	String	Prescribed values: "authorization", "authentication"
role	Role associated with the token to retrieve.	String	Prescribed values: "user", "admin", "wrapprivatekey

18.3.4 Logs relating to KEK management

The log fields described below relate to KEK management.

keks category

connect action

The connect action means that a connection to the KMS in REST mode has been tested. It generates an "info" severity log. The fields in this log are as follows:

Field	Description	Examples
persistence_type	Mode used for storing KEKs.	"kms", "json_file"

auto_refresh	Information on automatic KEK refresh: enabled/disabled, and refresh frequency in seconds.	"auto_refresh": {
		"enabled": true,
		"interval_seconds":86400,
		}

resource	IP address of the KMS, or URI of the *keks.json* file where the KEKs are located.	"file:///etc/stormshield/cse/keks.json"

***load* action**

The load action means that KEK retrieval has been triggered. It generates an "info" severity log. The fields in this log are as follows:

Field	Description	Examples
persistence_type	Mode used for storing KEKs.	"kms", "json_file"
resource	IP address of the KMS, or URI of the *keks.json* file where the KEKs are located.	"file:///etc/stormshield/cse/keks.json"
reason	Reason for triggering KEK retrieval. The possible values are: • "initialization" if the KEKs are loaded at application startup • "scheduled" if the KEKs are loaded following a scheduled refresh	

setup action

The setup action means that KEK refresh and KEK storing mode have been configured. It generates an "info" severity log. The fields in this log are as follows:

Field	Description	Examples
persistence_type	Mode used for storing KEKs.	"kms", "json_file"
resource	IP address of the KMS, or URI of the *keks.json* file where the KEKs are located.	"file:///etc/stormshield/cse/keks.json"

Field	Description	Examples

auto_refresh	Information about:
	• Periodic KEK refresh: enabled/disabled, and refresh frequency in seconds.
	• Minimum interval between two refresh operations, whether periodic or one-off.
		"auto_refresh": {
		"scheduled": {
		"enabled": true,
		"interval_seconds":86400
		}
		"minimum_interval_seconds": 1800
		}

kek category

***load* action**

The load action means that a KEK has been loaded in the SDS encryption service for Google Workspace memory. It generates an "info" severity log. The fields in this log are as follows:

Field	Description	Examples
kek_id	Unique identifier of the KEK used.	"bbc9cd0b-803c-4d9c-93f9-b43bdfc0bf96"
is_active_kek	Indicates if the KEK is the active encryption key for the tenant. The possible values are "true" and "false".	
persistence_type	Mode used for storing KEKs.	"kms", "json_file"
resource	IP address of the KMS, or URI of the *keks.json* file where the KEKs are located.	"file:///etc/stormshield/cse/keks.json"

18.3.5 Logs relating to the retrieval of key encryption keys (KEKs)

The log fields described below provide information on the retrieval status of KEKs. These logs are shown when the service starts and when the KEKs described in Renewing a KEK are refreshed.

Field	Description	Type	Examples
kek_id	Identifier of the KEK used.	String	"bbc9cd0b-803c-4d9c-93f9-b43bdfc0bf96"
is_active_kek	Active KEK used for encryption.	String	Prescribed values: "true", "false"
source	Source of the KEK.	String	Prescribed values: "kms", "local"

18.3.6 Logs relating to the connection to the KMS

The log fields described below provide information on the connection to the key management system (KMS) and the status of key encryption key (KEK) extraction.

These logs vary according to the type of cryptographic backend used: KMS REST API, or the SDS encryption service for Google Workspace.

KMS REST API

kms category

connect action

The connect action means that a connection to the KMS in REST mode has been tested. It generates an "info" severity log if the KMS is reachable, or a "warning" log if the KMS is unreachable. The fields in this log are as follows:

Field	Description	Examples
context.ca	Path to the certification authority	/etc/stormshield/cse/ca_kms.pem
context.cert	Path to the KMS client certificate	/etc/stormshield/cse/cert_kms.pem
context.host	URL of the KMS	https://web.ciphertrustmanager
context.port	KMS port	443
context.current_version	KMS API version	2.2.1
context.minimum_version	Minimum version supported by the SDS encryption service for Google Workspace	2.2
context.status	Connection status	Prescribed values: "true", "false"

SDS encryption service for Google Workspace

These logs are shown when the service starts and when the KEKs described in Renewing a KEK are refreshed.

Field	Description	Type	Examples
host	KMS address	String	"https://10.1.1.24"
port	KMS port	Integer	"5696"
supported_version	List of KMIP protocol versions supported by the KMS on which the user is connected.	String	Prescribed values: ["1.4","1.3","1.2","1.1","1.0"]
kmip_version	Version of the KMIP protocol used	String	"1.4"
status	Status of KEK extraction	String	Prescribed values: "started","starting", "stopped", "failure"

18.3.7 Logs relating to the *health* API route

The fields described below belong to logs about the *health* API route. This route is used to check that the SDS encryption service for Google Workspace is operating correctly.

Field	Description	Type	Examples
api_route	Name of the API route. Here "/health".	String	Prescribed values: "/health", "/status", "/wrap", "/unwrap", "/privilegedunwrap", "/digest", "/rewrap"
user_agent	User agent used in the request.	String	"Chrome/27.0.1453.110"
source_address	Network traffic source (client requesting the connection).	String	"172.16.16.212"
http_status	HTTP response code that indicates the status of the request to the proxy.	Integer	"200", "404", "413", "500"
response_payload	Data relating to the HTTP response.	Object	
name	Server name.	String	"SDS CSE"

18.3.8 Logs relating to the *status* API route

The fields described below belong to logs about the *status* API route. This route makes it possible to get information on the SDS encryption service for Google Workspace installed.

Field	Description	Type	Examples
api_route	URL slug of the API route.	String	"/api/v1/{tenantId}/status"
user_agent	User agent used in the request.	String	"Chrome/27.0.1453.110"
source_address	Network traffic source (client requesting the connection).	String	"172.16.16.212"
http_status	HTTP response code that indicates the status of the request to the proxy.	Integer	"200", "405", "413", "500"
response_payload	Data relating to the HTTP response.	Object	
name	Server name.	String	"Preprod SDS CSE"
server_type	Server type.	String	"KACLS"
vendor_id	Vendor ID of the server.	String	"SDS_CSE"
operations_supported	Operations supported by the service.	Array	['wrap', 'unwrap', 'privilegedwrap', 'privilegedunwrap', 'digest', 'rewrap', 'privatekeysign', 'privatekeydecrypt', 'privilegedprivatekeydecrypt']

18.3.9 Logs relating to the *wrap* API route

The fields described below belong to logs about the *wrap* API route. This route enables key wrapping.

Field	Description	Type	Examples
api_route	URL slug of the API route.	String	"/api/v1/{tenantId}/wrap"
user_agent	User agent used in the request.	String	"Chrome/27.0.1453.110"
source_address	Network traffic source (client requesting the connection).	String	"172.16.16.212"
http_status	HTTP response code that indicates the status of the request to the proxy.	Integer	Prescribed values: "200", "400", "401", "405", "413", "415", "500"
request_payload	Data relating to the HTTP request.	Object	
reason	JSON string providing additional context about the operation.	String	"{client:'drive' op:'update'}"
authorization_token	JWT guaranteeing that the user is allowed to wrap a key for 'resource_name'. It contains the information below:	Object	
	email: e-mail address of the user that the authorization token concerns.	String	"alice@domain.eu"
	email_type: Origin of the user's email address. • google: Google accounts (default value), • google-visitor: account verified by Google, • customer-idp: IDP account.	String	
	role: role requested in the authorization token. Prescribed value: "writer"	String	
	resource_name: resource identifier.	String	"6Bhds6BhdRkqt3Rkqt36Bhd"
	perimeter_id: identifier to conduct verifications of authentication and authorization requests.	String	"s6Bhds6BhdRkqt3Rkqt3"
	kacls_url: URL of the KACLS.	String	"https://someserver.eu"
	iss: identifies the service that generates the JWT (issuer).	String	"www.google.com"
	aud: identifies the recipient of the JWT (audience).	String array	"s6BhdRkqt3"

Field	Description	Type	Examples
	exp: identifies the expiry time after which the JWT must no longer be accepted.	Integer (timestamp in seconds)	"1694617320"
	iat: identifies the date on which the JWT was created (issued at).	Integer (timestamp in seconds)	"1694617320"
authentication_ token	JWT generated by a third-party tool that guarantees the user's identity. It contains the information below:	Object	
	claims: standard user data provided by the IDP. • email: e-mail address of the user that the authentication token concerns. • google_email: email address of the Google account of the user that the authentication token concerns. • iss: identifies the service that generates the JWT (issuer). • aud: identifies the recipient of the JWT (audience). • exp: identifies the expiry time after which the JWT must no longer be accepted. • iat: identifies the date on which the JWT was created (issued at).	String String String array Integer (timestamp in seconds) Integer (timestamp in seconds)	"username@domain.eu" "www.onelogin.com" "s6BhdRkqt3" "1694617320" "1694617320"
	number_of_custom_claims: number of custom claims included in the authentication token.	Integer	2
additional_data	Additional data. It contains the information below:	Object	
	wrap_properties: data relating to the wrap operation.	Object	
	kek_id: identifier of the KEK used.	String	"bbc9cd0b-803c-4d9c-93f9-b43bdfc0bf96"
	version: Version of the wrap operation.	Integer	1
	mode: Mode used for the wrap operation. • persistence: Persistence mode used	Object String	"json_file"
	cse_version: version of the service during the wrap operation.	String	"1.0.23458"

Field	Description	Type	Examples
	authentication_mode: authentication mode used for the wrap operation.	String	"local-configuration", "admin-configuration", "cse-configuration"
	authentication_domain: domain used for authentication.	String	"domain.com"

18.3.10 Logs relating to the certs API route

The fields described below belong to logs concerning the certs API route. This route is called by another KACLS, as part of a migration or backup operation: it returns the KACLS public certificate.

Field	Description	Type	Mandatory/Optional
keys	KACLS public certificate in JSON Web Key Set format as defined in RFC 7517. Example provided by Google.	JSON Web Key Set object	Mandatory

Other public certificate example:

```
"keys":   [
          {
                "kty": "RSA",
                "n": "o_mYVlR9dFTVilwx-aFhLNX-kdO-ClsYf8qP5fMVG-9-
wycen6oBmAmoQOumZP8zS3Sj6fxIC3PYB9wwW-2qAQuB7kEDT6V03-
8SIUz9S1lw",
                "e": "AQAB",
                "kid": "kacls-to-kacls-migration-key",
                "use": "sig",
                "alg": "RS256"
          }
    ]
```

18.3.11 Logs relating to the *unwrap* API route

The fields described below belong to logs about the *unwrap* API route. This route enables key unwrapping.

Field	Description	Type	Examples
api_route	URL slug of the API route.	String	"/api/v1/{tenantId}/unwrap"
user_agent	User agent used in the request.	String	"Chrome/27.0.1453.110"
source_address	Network traffic source (client requesting the connection).	String	"172.16.16.212"
http_status	HTTP response code that indicates the status of the request to the proxy.	Integer	Prescribed values: "200", "400","401", "405", "413", "415", "500"
request_payload	Data relating to the HTTP request.	Object	

Field	Description	Type	Examples
reason	JSON string providing additional context about the operation.	String	"{client:'drive' op:'update'}"
authorization_token	JWT attesting that the user is allowed to wrap a key for 'resource_name'. It contains the information below:	Object	
	email: e-mail address of the user that the authorization token concerns.	String	"alice@domain.eu"
	email_type: Origin of the user's email address. • google: Google accounts (default value), • google-visitor: account verified by Google, • customer-idp: IDP account.	String	
	role: role requested in the authorization token.	String	Prescribed values: "reader", "writer"
	resource_name: resource identifier.	String	"6Bhds6BhdRkqt3Rkqt36Bhd"
	perimeter_id: identifier to conduct verifications of authentication and authorization requests.	String	"s6Bhds6BhdRkqt3Rkqt3"
	kacls_url: URL of the KACLS.	String	"https://someserver.eu"
	iss: identifies the service that generates the JWT (issuer).	String	"www.google.com"
	aud: identifies the recipient of the JWT (audience).	String array	"s6BhdRkqt3"
	exp: identifies the expiry time after which the JWT must no longer be accepted.	Integer (timestamp in seconds)	"1694617320"
	iat: identifies the date on which the JWT was created (issued at).	Integer (timestamp in seconds)	"1694617320"
authentication_token	JWT generated by a third-party tool that guarantees the user's identity.	Object	

Field	Description	Type	Examples
	claims: standard user data provided by the IDP. • email: e-mail address of the user that the authentication token concerns. • google_email: email address of the Google account of the user that the authentication token concerns. • iss: identifies the service that generates the JWT (issuer). • aud: identifies the recipient of the JWT (audience). • exp: identifies the expiry time after which the JWT must no longer be accepted. • iat: identifies the date on which the JWT was created (issued at).	String String String array Integer (timestamp in seconds) Integer (timestamp in seconds)	"username@domain.eu" "www.onelogin.com" "s6BhdRkqt3" "1694617320" "1694617320"
	number_of_custom_claims: number of custom claims included in the authentication token.	Integer	2
additional_data	Additional data. It contains the information below:	Object	
	wrap_properties: data relating to the wrap operation.	Object	
	kek_id: identifier of the KEK used.	String	"bbc9cd0b-803c-4d9c-93f9-b43bdfc0bf96"
	version: version of the wrap operation.	Integer	1
	mode: mode used for the wrap operation. • persistence: persistence mode used	Object String	Prescribed values: "json_file"
	cse_version: version of the service during the wrap operation.	String	"1.0.23458"
	authentication_mode: authentication mode used for the wrap operation.	String	"local-configuration", "admin-configuration", "cse-configuration"
	authentication_domain: domain used for authentication.	String	"domain.com"

18.3.12 Logs relating to the *digest* API route

The fields described below belong to logs about the *digest* API route. Such routes make it possible to check whether the migration to another KACLS was successful.

Field	Description	Type	Examples
api_route	URL slug of the API route.	String	"/api/v1/{tenantId}/digest"
user_agent	User agent used in the request.	String	"Chrome/27.0.1453.110"
source_address	Network traffic source (client requesting the connection).	String	"172.16.16.212"
http_status	HTTP response code that indicates the status of the request to the proxy.	Integer	Prescribed values: "200", "400","401", "405", "413", "415", "500"
request_payload	Data relating to the HTTP request.	Object	
reason	JSON string providing additional context about the operation.	String	"{client:'drive' op:'update'}"
authorization_token	JWT attesting that the user is allowed to wrap a key for 'resource_name'. It contains the information below:	Object	
	email: e-mail address of the user that the authorization token concerns.	String	"alice@domain.eu"
	role: role requested in the authorization token.	String	Prescribed value: "check"
	resource_name: resource identifier.	String	"6Bhds6BhdRkqt3Rkqt36Bhd"
	kacls_url: URL of the KACLS.	String	"https://someserver.eu"
	iss: identifies the service that generates the JWT (issuer).	String	"www.google.com"
	aud: identifies the recipient of the JWT (audience).	String array	"s6BhdRkqt3"
	exp: identifies the expiry time after which the JWT must no longer be accepted.	Integer (timestamp in seconds)	"1694617320"
	iat: identifies the date on which the JWT was created (issued at).	Integer (timestamp in seconds)	"1694617320"
additional_data	Additional data. It contains the information below:	Object	
	wrap_properties: data relating to the wrap operation.	Object	
	kek_id: identifier of the KEK used.	String	"bbc9cd0b-803c-4d9c-93f9-b43bdfc0bf96"

Field	Description	Type	Examples
	version: version of the wrap operation.	Integer	1
	mode: mode used for the wrap operation. • persistence: persistence mode used	Object String	Prescribed values: "json_file"
	cse_version: version of the service during the wrap operation.	String	"1.0.23458"
	authentication_mode: authentication mode used for the wrap operation.	String	"local-configuration", "admin-configuration", "cse-configuration"
	authentication_domain: domain used for authentication.	String	"domain.com"

18.3.13 Logs relating to the *rewrap* API route

The fields described below belong to logs about the *rewrap* API route. Such routes make it possible to migrate from one KACLS to another.

Field	Description	Type	Examples
api_route	URL slug of the API route.	String	"/api/v1/{tenantId}/rewrap"
user_agent	User agent used in the request.	String	"Chrome/27.0.1453.110"
source_address	Network traffic source (client requesting the connection).	String	"172.16.16.212"
http_status	HTTP response code that indicates the status of the request to the proxy.	Integer	Prescribed values: "200", "400", "401", "405", "413", "415", "500"
request_payload	Data relating to the HTTP request.	Object	
reason	JSON string providing additional context about the operation.	String	"{client:'drive' op:'update'}"
authorization_token	JWT guaranteeing that the user is allowed to wrap a key for 'resource_name'. It contains the information below:	Object	
	email: e-mail address of the user that the authorization token concerns.	String	"alice@domain.eu"
	role: role requested in the authorization token.	String	Prescribed value: "migrator"
	resource_name: resource identifier.	String	"6Bhds6BhdRkqt3Rkqt36Bhd"
	kacls_url: URL of the KACLS.	String	"https://someserver.eu"

Field	Description	Type	Examples
	iss: identifies the service that generates the JWT (issuer).	String	"www.google.com"
	aud: identifies the recipient of the JWT (audience).	String array	"s6BhdRkqt3"
	exp: identifies the expiry time after which the JWT must no longer be accepted.	Integer (timestamp in seconds)	"1694617320"
	iat: identifies the date on which the JWT was created (issued at).	Integer (timestamp in seconds)	"1694617320"
authentication_token	JWT generated by a third-party tool that guarantees the user's identity. It contains the information below:	Object	
	claims: standard user data provided by the IDP. • email: e-mail address of the user that the authentication token concerns. • google_email: email address of the Google account of the user that the authentication token concerns. • iss: identifies the service that generates the JWT (issuer). • aud: identifies the recipient of the JWT (audience). • exp: identifies the expiry time after which the JWT must no longer be accepted. • iat: identifies the date on which the JWT was created (issued at).	String String String array Integer (timestamp in seconds) Integer (timestamp in seconds)	"username@domain.eu" "www.onelogin.com" "s6BhdRkqt3" "1694617320" "1694617320"
	number_of_custom_claims: number of custom claims included in the authentication token.	Integer	2
additional_data	Additional data. It contains the information below:	Object	
	wrap_properties: data relating to the wrap operation.	Object	
	kek_id: identifier of the KEK used.	String	"bbc9cd0b-803c-4d9c-93f9-b43bdfc0bf96"
	version: Version of the wrap operation.	Integer	1

Field	Description	Type	Examples
	mode: Mode used for the wrap operation. • persistence: Persistence mode used	Object String	"json_file"
	cse_version: version of the service during the wrap operation.	String	"1.0.23458"
	authentication_mode: authentication mode used for the wrap operation.	String	"local-configuration", "admin-configuration", "cse-configuration"
	authentication_domain: domain used for authentication.	String	"domain.com"

18.3.14 Logs relating to the *privilegedwrap* API route

The fields described below belong to logs about the *privilegedwrap* API route. This route allows the administrator to perform bulk file import.

Field	Description	Type	Example
api_route	URL slug of the API route.	String	"/api/v1/{tenantId}/privilegedwrap"
user_agent	User agent used in the request.	String	"Chrome/27.0.1453.110"
source_address	Network traffic source (client requesting the connection).	String	"172.16.16.212"
http_status	HTTP response code that indicates the status of the request to the proxy.	Integer	Prescribed values: "200", "400", "401", "405", "413", "415", "500"
request_payload	Data relating to the HTTP request.	Object	
reason	JSON string providing additional context about the operation.	String	"{client:'drive' op:'update'}"
authentication_token	JWT generated by a third-party tool that guarantees the user's identity. It contains the information below:	Object	
	claims: standard user data provided by the IDP. • email: e-mail address of the user that the authentication token concerns. • iss: identifies the service that generates the JWT (issuer). • aud: identifies the recipient of the JWT (audience). • exp: identifies the expiry time after which the JWT must no longer be accepted. • iat: identifies the date on which the JWT was created (issued at).	String String String array Integer (timestamp in seconds) Integer (timestamp in seconds)	"username@domain.eu" "www.onelogin.com" "s6BhdRkqt3" "1694617320" "1694617320"
	number_of_custom_claims: number of custom claims included in the authentication token.	Integer	2
additional_data	Additional data. It contains the information below:	Object	
	wrap_properties: data relating to the wrap operation.	Object	
	kek_id: identifier of the KEK used.	String	"bbc9cd0b-803c-4d9c-93f9-b43bdfc0bf96"

Field	Description	Type	Example
	version: version of the wrap operation.	Integer	1
	mode: mode used for the wrap operation. • persistence: persistence mode used	Object String	"json_file"
	cse_version: version of the service during the wrap operation.	String	"1.0.23458"
	authentication_mode: authentication mode used for the wrap operation.	String	"local-configuration", "admin-configuration", "cse-configuration"
	authentication_domain: domain used for authentication.	String	"domain.com"

18.3.15 Logs relating to the *privilegedunwrap* API route

The fields described below belong to logs about the *privilegedunwrap* API route. This route allows:

- An administrator to decrypt exported user data with the *decrypter.exe* Google utility,
- An encryption service to migrate data from another KACLS to itself.

Field	Description	Type	Examples
api_route	URL slug of the API route.	String	"/api/v1/{tenantId}/privilegedunwrap"
user_agent	User agent used in the request.	String	"Chrome/27.0.1453.110"
source_address	Network traffic source (client requesting the connection).	String	"172.16.16.212"
http_status	HTTP response code that indicates the status of the request to the proxy.	Integer	Prescribed values: "200", "400", "401", "405", "413", "415","500"
request_payload	Data relating to the HTTP request.	Object	
reason	JSON string providing additional context about the operation.	String	"{client:'drive' op:'update'}" In a migration, the value is set: "KACLS migration"
resource_name	Resource identifier.	String	"6Bhds6BhdRkqt3Rkqt36Bhd"
authentication_token	JWT generated by a third-party tool that guarantees the user's identity. It contains the information below:	Object	

Field	Description	Type	Examples
	Only for user use cases email: e-mail address of the user that the authentication token concerns.	String	"alice@domain.eu"
	Only for migration use cases kaclsUrl: URL of the KACLS that initiates the privileged unwrap request.	String	"https://cse.mysds.io/api/v1/0995624d-13f5-40a9-9c59-fee6fe3ef3f4"
	iss: identifies the service that generates the JWT (issuer).	String	URL of the issuing KACLS
	aud: identifies the recipient of the JWT (audience).	String array	In a migration, the value is set: "kacls-migration"
	exp: identifies the expiry time after which the JWT must no longer be accepted.	Integer (timestamp in seconds)	"1694617320"
	iat: identifies the date on which the JWT was created (issued at).	Integer (timestamp in seconds)	"1694617320"
additional_data	Additional data. It contains the information below:	Object	
	wrap_properties: data relating to the wrap operation.	Object	
	kek_id: identifier of the KEK used.	String	"bbc9cd0b-803c-4d9c-93f9-b43bdfc0bf96"
	version: version of the wrap operation.	Integer	1
	mode: mode used for the wrap operation. • persistence: persistence mode used (*none*)	String	Prescribed values: • *kms* or *json file* for persistence
	cse_version: version of the service during the wrap operation.	String	"1.0.23458"
	authentication_mode: authentication mode used for the wrap operation.	String	"local-configuration", "admin-configuration", "cse-configuration" In a migration, the value is set: "migration"
	authentication_domain: domain used for authentication.	String	"domain.com"

18.3.16 Logs relating to the *wrapprivatekey* API route

The fields described below belong to logs about the *wrapprivatekey* API route. This internal route makes it possible for Stormshield to encrypt user keys for Gmail. It is never called by Google.

Field	Description	Type	Example
api_route	URL slug of the API route.	String	"/api/v1/{tenantId}/wrapprivatekey"
user_agent	User agent used in the request.	String	"Chrome/27.0.1453.110"
source_address	Network traffic source (client requesting the connection).	String	"172.16.16.212"
http_status	HTTP response code that indicates the status of the request to the proxy.	Integer	Prescribed values: "200", "400", "401", "405", "413", "415", "500"
request_payload	Data relating to the HTTP request.	Object	
authentication_token	JWT generated by a third-party tool that guarantees the user's identity. It contains the information below:	Object	
	claims: standard user data provided by the IDP. • email: e-mail address of the user that the authentication token concerns. • iss: identifies the service that generates the JWT (issuer). • aud: identifies the recipient of the JWT (audience). • exp: identifies the expiry time after which the JWT must no longer be accepted. • iat: identifies the date on which the JWT was created (issued at).	String String String array Integer (timestamp in seconds) Integer (timestamp in seconds)	"username@domain.eu" "www.onelogin.com" "s6BhdRkqt3" "1694617320" "1694617320"
	number_of_custom_claims: number of custom claims included in the authentication token.	Integer	2
additional_data	Additional data. It contains the information below:	Object	
	wrap_properties: data relating to the wrap operation.	Object	
	kek_id: identifier of the KEK used.	String	"80c65a46-33db-4f26-bfe3-cefbbb4f16d8"

Field	Description	Type	Example
	version: version of the wrap operation.	Integer	1
	mode: Mode used for the wrap operation. • persistence: Persistence mode used	Object String	"json_file", "kms""
	cse_version: version of the service during the wrap operation.	String	"4.1.1637-0.2.beta"
	supported_algorithms: encryption and signature algorithms used with this key.	String array	["RSA/ECB/PKCS1Padding", "RSA/ECB/OAEPwithSHA-1andMGF1Padding","RSA/ECB/OAEPwithSHA-256andMGF1Padding","RSA/ECB/OAEPwithSHA-512andMGF1Padding","SHA1withRSA","SHA256withRSA","SHA512withRSA","SHA1withRSA/PSS","SHA256withRSA/PSS","SHA512withRSA/PSS"]

18.3.17 Logs relating to the *privatekeysign* API route

The fields described below belong to logs concerning the *privatekeysign* API route. Google calls up this route when it encrypts and sends encrypted emails.

Field	Description	Type	Example
api_route	URL slug of the API route.	String	"/api/v1/{tenantId}/privatekeysign"
user_agent	User agent used in the request.	String	"Chrome/27.0.1453.110"
source_address	Network traffic source (client requesting the connection).	String	"172.16.16.212"
http_status	HTTP response code that indicates the status of the request to the proxy.	Integer	Prescribed values: "200", "400", "401", "405", "413", "415", "500"
request_payload	Data relating to the HTTP request.	Object	
reason	JSON string providing additional context about the operation.	String	"sign email"
authorization_token	JWT guaranteeing that the user is allowed to wrap a key for 'resource_name'. It contains the information below:	Object	
	email: e-mail address of the user that the authorization token concerns.	String	"username@domain.eu"
	role: role requested in the authorization token.	String	Prescribed value: "signer"
	resource_name: resource identifier.	String	"6Bhds6BhdRkqt3Rkqt36Bhd"
	perimeter_id: identifier to conduct verifications of authentication and authorization requests.	String	"s6Bhds6BhdRkqt3Rkqt3"
	kacls_url: URL of the KACLS.	String	"https://someserver.eu"
	iss: identifies the service that generates the JWT (issuer).	String	"gsuitecse-tokenissuer-gmail@system.gserviceaccount.com"
	aud: identifies the recipient of the JWT (audience).	String array	"cse-authorization"
	exp: identifies the expiry time after which the JWT must no longer be accepted.	Integer (timestamp in seconds)	"1694617320"
	iat: identifies the date on which the JWT was created (issued at).	Integer (timestamp in seconds)	"1694617320"

Field	Description	Type	Example
	spki_hash: digest of the private key in Base64.	String	"saEz24lohDOHIGjddhscQsdjCQfFBqNHs1crLUE+Kt4="
	spki_hash_algorithm: encryption algorithm used	String	"SHA-256"
	message_id: optional ID of the message to which the signature applies.	String	
authentication_token	JWT generated by a third-party tool that guarantees the user's identity. It contains the information below:	Object	
	claims: standard user data provided by the IDP. • email: e-mail address of the user that the authentication token concerns. • google_email: email address of the Google account of the user that the authentication token concerns. • iss: identifies the service that generates the JWT (issuer). • aud: identifies the recipient of the JWT (audience). • exp: identifies the expiry time after which the JWT must no longer be accepted. • iat: identifies the date on which the JWT was created (issued at).	String String String array Integer (timestamp in seconds) Integer (timestamp in seconds)	"username@domain.eu" "www.onelogin.com" "s6BhdRkqt3" "1694617320" "1694617320"
	number_of_custom_claims: number of custom claims included in the authentication token.	Integer	2
additional_data	Additional data. It contains the information below:	Object	
	wrap_properties: data relating to the wrap operation.	Object	
	kek_id: identifier of the KEK used.	String	"80c65a46-33db-4f26-bfe3-cefbbb4f16k8"
	version: version of the wrap operation.	Integer	"2"

Field	Description	Type	Example
	mode: Mode used for the wrap operation. • persistence: Persistence mode used	Object String	"json_file", "kms"
	cse_version: version of the service during the wrap operation.	String	"4.1.1637-0.2.beta"
	key_name : name of the private key used by the KMS to sign in 'kms' mode.	String	38cf0196-1fbf-11ee-be56-0242ac120002
	crypto_mode: type of cryptographic backend used to decrypt session keys.	String	"kms" or "node"
	supported_algorithms: encryption and signature algorithms used with this key.	String array	["RSA/ECB/PKCS1Padding","RSA/ECB/OAEPwithSHA-1andMGF1Padding","RSA/ECB/OAEPwithSHA-256andMGF1Padding","RSA/ECB/OAEPwithSHA-512andMGF1Padding","SHA1withRSA","SHA256withRSA","SHA512withRSA","SHA1withRSA/PSS","SHA256withRSA/PSS","SHA512withRSA/PSS"]

18.3.18 Logs relating to the *privatekeydecrypt* API route

The fields described below belong to logs about the *privatekeydecrypt* API route. Google calls up this route when it decrypts an encrypted email.

	Description	Type	
api_route	URL slug of the API route.	String	"/api/v1/{tenantId}/privatekeydecrypt"
user_agent	User agent used in the request.	String	"Chrome/27.0.1453.110"
source_address	Network traffic source (client requesting the connection).	String	"172.16.16.212"
http_status	HTTP response code that indicates the status of the request to the proxy.	Integer	Prescribed values: "200", "400", "401", "405", "413", "415", "500"
request_payload	Data relating to the HTTP request.	Object	
reason	JSON string providing additional context about the operation.	String	"unpack request"
authorization_token	JWT guaranteeing that the user is allowed to wrap a key for 'resource_name'. It contains the information below:	Object	
	email: e-mail address of the user that the authorization token concerns.	String	"username@domain.eu"
	role: role requested in the authorization token.	String	Prescribed value: "decrypter"
	resource_name: resource identifier.	String	"6Bhds6BhdRkqt3Rkqt36Bhd"
	perimeter_id: identifier to conduct verifications of authentication and authorization requests.	String	"s6Bhds6BhdRkqt3Rkqt3"
	kacls_url: URL of the KACLS.	String	"https://someserver.eu"
	iss: identifies the service that generates the JWT (issuer).	String	"gsuitecse-tokenissuer-gmail@system.gserviceaccount.com"
	aud: identifies the recipient of the JWT (audience).	String array	"cse-authorization"
	exp: identifies the expiry time after which the JWT must no longer be accepted.	Integer (timestamp in seconds)	"1694617320"
	iat: identifies the date on which the JWT was created (issued at).	Integer (timestamp in seconds)	"1694617320"

	Description	Type	
	spki_hash: hash of the private key in Base64.	String	"saEz24lohDOHIGjddhscQsdjCQfFBqNHs1crLUE+Kt4="
	spki_hash_algorithm: algorithm used to produce the spki_hash hash.	String	"SHA-256"
	message_id: optional ID of the message to which the signature applies.	String	
authentication_token	JWT generated by a third-party tool that guarantees the user's identity. It contains the information below:	Object	
	claims: standard user data provided by the IDP. • email: email address of the user that the authentication token concerns. • google_email: email address of the Google account of the user that the authentication token concerns. • iss: identifies the service that generates the JWT (issuer). • aud: identifies the recipient of the JWT (audience). • exp: identifies the expiry time after which the JWT must no longer be accepted. • iat: identifies the date on which the JWT was created (issued at).	String String String array Integer (timestamp in seconds) Integer (timestamp in seconds)	"username@domain.eu" "www.onelogin.com" "s6BhdRkqt3" "1694617320" "1694617320"
	number_of_custom_claims: number of custom claims included in the authentication token.	Integer	2
additional_data	Additional data. It contains the information below:	Object	
	wrap_properties: data relating to the wrap operation.	Object	
	kek_id: identifier of the KEK used.	String	"80c65a46-33db-4f26-bfe3-cefbbb4f16d8"
	version: version of the wrap operation.	Integer	1

	Description	Type	
	mode: mode used for the wrap operation. • persistence: persistence mode used	Object String	 "json_file", "kms""
	cse_version: version of the service during the wrap operation.	String	"4.1.1637-0.2.beta"
	key_name : name of the public key used by the KMS to sign in 'kms' mode.	String	38cf0196-1fbf-11ee-be56-0242ac120002div>
	crypto_mode: type of cryptographic backend used to decrypt session keys.	String	"kms" or "node"
	supported_algorithms: encryption and signature algorithms used with this key.	String array	["RSA/ECB/PKCS1Padding","RSA/ECB/OAEPwithSHA-1andMGF1Padding","RSA/ECB/OAEPwithSHA-256andMGF1Padding","RSA/ECB/OAEPwithSHA-512andMGF1Padding","SHA1withRSA","SHA256withRSA","SHA512withRSA","SHA1withRSA/PSS","SHA256withRSA/PSS","SHA512withRSA/PSS"]

18.3.19 Logs relating to the *privilegedprivatekeydecrypt* API route

The fields described below belong to logs concerning the *privilegedprivatekeydecrypt* API route. A Google administrator calls up this privileged route to decrypt an encrypted email, for example via the Google *decrypter.exe* utility.

Field	Description	Type	Example
api_route	URL slug of the API route.	String	"/api/v1 {tenantId}/privilegedprivatekeydecrypt"
user_agent	User agent used in the request.	String	"Chrome/27.0.1453.110"
source_address	Network traffic source (client requesting the connection).	String	"172.16.16.212"
http_status	HTTP response code that indicates the status of the request to the proxy.	Integer	Prescribed values: "200", "400", "401", "405", "413", "415", "500"
request_payload	Data relating to the HTTP request.	Object	
reason	JSON string providing additional context about the operation.	String	"Command-line decrypter"
authentication_token	JWT generated by a third-party tool that guarantees the user's identity. It contains the information below:	Object	
	claims: standard user data provided by the IDP. • email: e-mail address of the user that the authentication token concerns. • google_email: email address of the Google account of the user that the authentication token concerns. • iss: identifies the service that generates the JWT (issuer). • aud: identifies the recipient of the JWT (audience). • exp: identifies the expiry time after which the JWT must no longer be accepted. • iat: identifies the date on which the JWT was created (issued at).	String String String array Integer (timestamp in seconds) Integer (timestamp in seconds)	"username@domain.eu" "www.onelogin.com" "s6BhdRkqt3" "1694617320" "1694617320"
	number_of_custom_claims: number of custom claims included in the authentication token.	Integer	2
additional_data	Additional data. It contains the information below:	Object	

Field	Description	Type	Example
	wrap_properties: data relating to the wrap operation.	Object	
	kek_id: identifier of the KEK used.	String	"80c65a46-33db-4f26-bfe3-cefbbb4f16d8"
	version: version of the wrap operation.	Integer	1
	mode: mode used for the wrap operation. • persistence: persistence mode used	Object String	"json_file", "kms""
	cse_version: version of the service during the wrap operation.	String	"4.1.1637-0.2.beta"
	key_name : name of the private key used by the KMS to sign in 'kms' mode.	String	38cf0196-1fbf-11ee-be56-0242ac120002div>
	crypto_mode: type of cryptographic backend used to decrypt session keys.	String	"kms" or "node"
	supported_algorithms: encryption and signature algorithms used with this key.	String array	["RSA/ECB/PKCS1Padding", "RSA/ECB/OAEPwithSHA-1andMGF1Padding","RSA/ECB/OAEPwithSHA-256andMGF1Padding","RSA/ECB/OAEPwithSHA-512andMGF1Padding","SHA1withRSA","SHA256withRSA","SHA1withRSA/PSS","SHA256withRSA/PSS","SHA512withRSA/PSS"]

18.3.20 Logs relating to the application of an OPA policy

The log fields described below relate to the application of an OPA policy. For more information, see the section Customizing the authorization rules.

The *policy.wasm* and *policy.data.json* files are optional. If one of the files is not present, the service starts and no policies are applied. A log is issued to indicate that the policy is disabled.

Field	Description	Type	Examples
status	Status of the policy.	String	Prescribed values: "enabled", "disabled", "loading"
loadingFile	File is being loaded.	String	Prescribed values: "policy", "data"
type	Type of the applied policy.	String	Prescribed value: "opa"

18.4 Understanding the new log format

Version 4.3 of the SDS encryption service for Google Workspace introduces a new log format. It provides greater granularity and optimizes log tracking.

Eventually, all logs will be displayed in this format, but currently only the logs and fields described in the SDS encryption service for Google Workspace *Log Guide* are available. Refer to this guide for a full description of logs in the new format.

18.4.1 Configuring the display of logs in the new format

Displaying logs in the new format is optional. If you want to enable it, edit the *config.json* file in the *logs* section.

You can also filter logs to display only certain log families and severity levels.

For more information, refer to the section logs parameter.

18.4.2 Correlation identifier

A unique identifier in UUIDV4 format is automatically generated for each request. This is the correlation ID linking all logs related to the same request or event.

You can customize this ID by defining in the requests an *x-request-id* header containing a string of up to 360 characters.

If the header is missing or invalid, the correlation ID is generated automatically.

19. Uninstalling the SDS encryption service for Google Workspace

19.1 In RPM mode

1. Run the following command as a user with administration privileges:
   ```
   # rpm -e cse
   ```
   The files added when installing the RPM are deleted, except:
   - The files that you have modified in the meantime. They are saved with the *.rpmsave* extension.
   - The file that you have added yourself. They are kept.
2. Manually delete the configuration files that you have created or modified: *config.json*, *keks.json, policy.wasm and policy.data.json.*

NOTE
KEKs are highly sensitive items in terms of security. It is imperative to follow the ANSSI recommendations concerning their life cycle.

19.2 In Docker mode

1. Clean up the containers:
   ```
   docker stop <container_name>
   docker rm <container_name>
   ```
2. Delete the local image:
   ```
   docker rmi stormshield/kmaas:<version>
   ```
3. Delete the data volumes:
   ```
   docker volume rm <volume_name>
   ```

20. Further reading

Additional information and answers to questions you may have about SDS encryption service for Google Workspace are available on the Documentation website and in the Stormshield knowledge base (authentication required).

documentation@stormshield.eu