AN12642

EdgeLock 2GO developer guide for Foundries.io users

Rev. 1.0 — 19 September 2022 Application note
614310

Document information

Information	Content
Keywords	EdgeLock 2GO, managed, API, secure objects, Foundries.io
Abstract	This document describes how to leverage on the Foundries.io API to create, update, delete and manage resources on the NXP EdgeLock 2GO platform. It also explains how to leverage the REST API to create device groups, register your devices and associate them to secure objects.

Revision history

Revision history

Revision number	Date	Description
1.0	20220919	First official released version

1 Introduction

EdgeLock 2GO is a fully managed cloud platform operated by NXP that provides secure provisioning services for easy deployment and maintenance of IoT devices that integrate EdgeLock SE05x secure elements.

EdgeLock 2GO provides access to several complementary services, including the Key Delivery Service (KDS) for the secure distribution of keys injected in secure element custom types, the applet update service for downloading IoT applet update scripts for EdgeLock SE051 and the EdgeLock 2GO Managed service.

EdgeLock 2GO Managed is a service that allows you to create and manage **secure objects**, such as symmetric roots of trusts, key-pairs and certificates, which are then securely provisioned by EdgeLock 2GO Managed into the secure elements of your IoT devices.

Secure objects and certificates created through EdgeLock 2GO Managed can be used for a wide variety of use cases, including for **secure cloud onboarding** of IoT devices to your preferred cloud service (e.g. AWS IoT Core or Azure IoT Hub), data encryption or decryption, access control, etc.

EdgeLock 2GO Managed abstracts the complexity of key and certificate management so you don't need to invest in a PKI infrastructure. Security of EdgeLock 2GO Managed relies on roots of trust that are injected into secure elements during their manufacturing and then uploaded to the platform. These roots of trust allow EdgeLock 2GO Managed to customize the secure elements and provision them with your device-specific certificates and keys.

EdgeLock 2GO Managed can be easily used and configured through the EdgeLock 2GO web portal or the EdgeLock 2GO REST API.

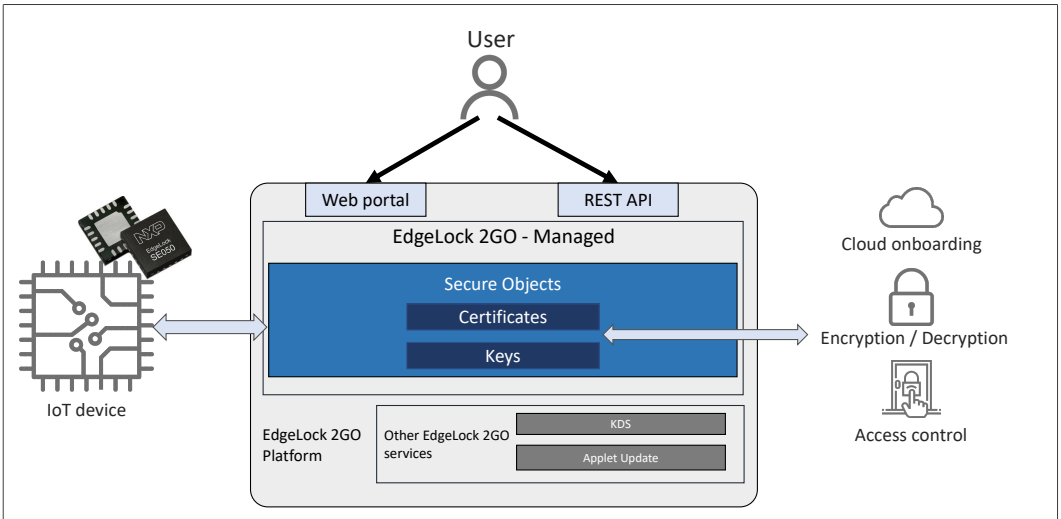

Figure 1. EdgeLock 2GO overview

2 API security model

The API is available at the following URL:

```
https://api.foundries.io/ota/factories/<factory>/el2g-proxy/
```

All the links provided in this document leave out the actual host and protocol parts of the URL to make this document more readable. If you want to obtain the full URL, you should append the endpoint URL signature to the base URL; e.g. if the endpoint signature is `/secure-objects`, the full API endpoint URL becomes `https://api.foundries.io/ota/factories/<factory>/el2g-proxy/secure-objects`.

API authentication and authorization can be performed with two different methods: **OAuth2 tokens** or **API tokens**. More information can be found in the API access page of the Foundries.io website. The examples provided in this document use API tokens for authorization. The API token must be sent for each request as an HTTP header field as shown below:

```
GET /services
OSF-TOKEN: <token>
```

Note: *Any implementation which is calling the EdgeLock 2Go REST API requires the following configuration:*

- *TLS 1.2*
- *Server Name Indication (SNI) according to RFC6066 (https://tools.ietf.org/html/rfc6066)*
- *Always use the official DNS Name (api.foundries.io) for the REST API call*

Typical REST API clients and their supported TLS libraries support TLS 1.2 and SNI as a default. Sometimes the required TLS configuration must be activated explicitly. Please refer to the documentation of the REST API library you are using.

3 Using cURL to run the examples

All the API endpoints described in this document come with one or more call examples that can be executed using the cURL tool. cURL is a lightweight command-line tool for making HTTP requests without a web browser. cURL is particularly useful since it provides a language-agnostic way of testing API endpoints. Moreover, cURL has been ported to almost every operating system including Windows, Linux and MAC OS. If you are using Linux or MAC OS, cURL is already installed and ready to use from the terminal. If you have version 1803 or later of Windows 10, cURL is also installed by default and can be used from the Command Prompt. If you have an earlier version of Windows, you can follow the instructions provided here to install cURL.

Each cURL example will show the cURL command structure and necessary parameters, including the necessary HTTP headers. Parameters that must be filled by the users are enclosed in <>. To improve readability, the JSON body of the request will be listed as a separate entry and must be included in the cURL command as an escaped string. You can use this online tool to escape the JSON body.

For example, given the following request:

Request

cURL command:

```
curl -X POST "<base_URL>/products/<12nc>/device-groups/<device_group_id>/devices"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
     -H "Content-Type: application/json"
     -d "<JSON_body>"
```

JSON body:

```
{
  "deviceIds": [
    "348555491056936142262187597632590342062080"
  ]
}
```

After the substitution of the parameters, the complete cURL command would be similar to the one shown below:

```
curl -X POST "https://api.foundries.io/ota/factories/<factory>/el2g-
proxy/products/120405220204/device-groups/122/devices"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
     -H "Content-Type: application/json"
     -d "{\"deviceIds\":[\"348555492341936142262187597635711342062080\"]}"
```

4 EdgeLock 2GO API usage

This section provides an overview of the EdgeLock 2GO REST API endpoints. The description of each API endpoint will contain:

- The endpoint signature, including path parameters;
- A text description describing the endpoint function;
- An example request using the cURL tool. See Section 3 for instructions on how to use cURL.

For a more detailed description of the EdgeLock 2GO API parameters, including additional header and query parameters, please refer to the EdgeLock 2GO OpenAPI specification.

Section 4.1 describes the full API flow required to create secure objects and associate them to your devices.

Section 4.2 describes the full API flow required to upload to EdgeLock 2GO an intermediate certificate for the remote trust provisioning service signed by an external Certificate Authority (CA).

The remaining sections list and describe all the other API endpoints available:

- Additional API endpoints for device group management
- Additional API endpoints for claim codes management
- Additional API endpoints for secure object management
- Additional API endpoints for intermediate certificate management
- Additional API endpoints for activity management

4.1 API flow: Secure object provisioning

This section details the configuration steps required in EdgeLock 2GO to provision a secure object to devices using the EdgeLock 2GO REST API. The process consists of the following steps:

1. Create a device group
2. Add devices to a device group
3. Create a secure object
4. Assign a secure object to a device group

The full provisioning flow is schematized in Figure 2.

Note: Make sure that the API key you are using has all the permissions required to execute the API calls listed in this section.

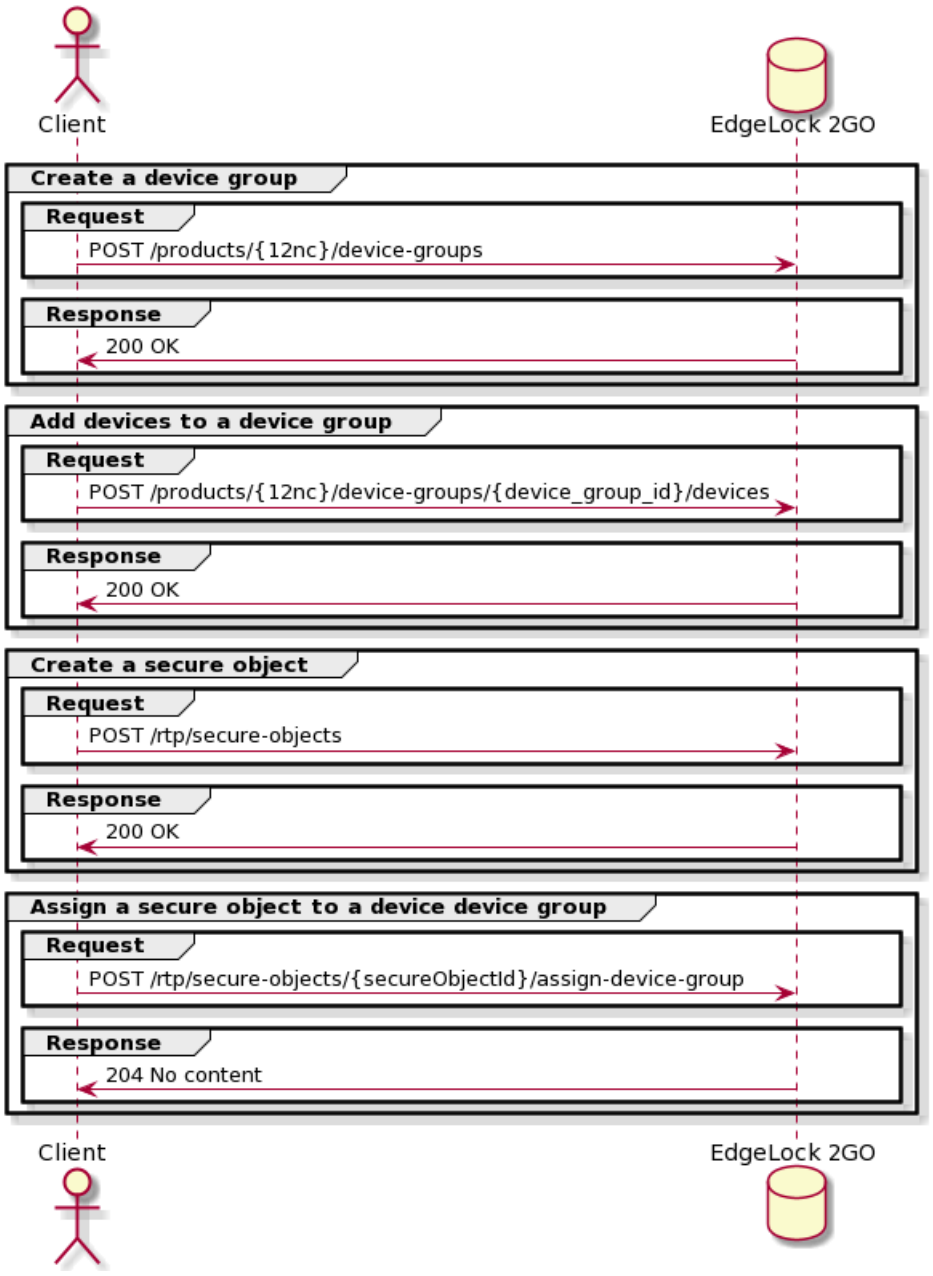

Figure 2. Provisioning a secure object in EdgeLock 2GO

4.1.1 Add a device group

```
POST /products/{12nc}/device-groups
```

With EdgeLock 2GO, you can organize your devices by group. A group typically includes devices with the same key and certificate configuration. Before adding a device, it is necessary to create a device group. This API endpoint allows you to add a new device group to EdgeLock 2GO. To create a new device group, you should specify the hardware 12NC code as a path parameter and the device group name as JSON body parameter. If the provided parameters are correct, the device group is created in EdgeLock 2GO and

a unique ID is assigned to it. The device group ID is returned in the JSON response and can be used to reference the device group in other API requests.

Note: The selection of the correct hardware 12NC is fundamental for the correct provisioning of the devices belonging to the device group. You can retrieve the 12NC using the API endpoint described in Section 4.3.1.

An example call is reported below:

Request

cURL Command:

```
curl -X POST "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/products/
<12nc>/device-groups"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
     -H "Content-Type: application/json"
     -d "[escaped JSON request body, see Section 3]"
```

JSON request body:

```
{
  "deviceGroupName": "TestGroup"
}
```

Response

```
{
  "id": 474,
  "deviceGroupName": "TestGroup",
  "deviceCount": 0,
  "productRelationship": "USER",
  "productDetails": {
    "nc12": "935389312472",
    "commercialName": "SE050C2HQ1/Z01V3",
    "hardwareType": "SE050",
    "hardwareVariant": "SE050C2",
    "hardwareFamilyType": "SE05",
    "softwareVersion": "0"
  },
  "resourceMetadata": {
    "createdBy": "apiKey: ApiKey",
    "createdTs": "2021-09-28T08:33:06.5Z",
    "lastModifiedBy": "apiKey: ApiKey",
    "lastModifiedTs": "2021-09-28T08:33:06.5Z",
    "version": null
  },
  "claimCodeOverviews": null
}
```

4.1.2 Add devices to a device group

```
POST /products/{12nc}/device-groups/{deviceGroupId}/devices
```

After creating a device group, we need to add a device or a batch of devices to the device group. This API endpoint allows you to add a device or a batch of devices to the EdgeLock 2GO. To add a device, you have to specify as path parameters the hardware 12NC code and the device group ID obtained in Section 4.1.1. The UIDs of the devices to add to the device group should be placed in an array in the JSON body of the request in big Int format or in hexadecimal format (UIDs in hexadecimal format must be prepended with 0x0, e.g., `0x040050011ce89d305cf379041f1d12946680`). To extract the UID of your device, refer to Section 8. To add a device using claim codes instead of UIDs, please refer to Section 4.4.1 and Section 10.

Note: The selection of the correct hardware 12NC is fundamental for the correct provisioning of the devices belonging to the device group. You can retrieve the 12NC using the API endpoint described in Section 4.3.1.

Note: The device UIDs are always returned in big int format in the JSON response, even if they were provided in HEX format.

An example call is reported below:

Request

cURL Command:

```
curl -X POST "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/products/
<12nc>/device-groups/<device_group_id>/devices"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
     -H "Content-Type: application/json"
     -d "[escaped JSON request body, see Section 3]"
```

JSON request body:

```
{
  "deviceIds": [
    "34855549105693614226218759763259034206208O"
  ]
}
```

Response

```
{
  "current": "https://api.foundries.io/ota/factories/<factory>/el2g-
proxy/products/120405220204/device-groups/474/devices?page=0&size=20",
  "next": null,
  "prev": null,
  "totalElements": 1,
  "pageType": "DeviceServiceProvisioningPage",
  "content": [
    {
      "device": {
        "id": "34855549105693614226218759763259034206208O",
        "deviceGroupId": 474,
        "deviceGroupName": "TestGroup",
        "productDetail": null,
        "lastConnection": null,
        "resourceMetadata": {
          "createdBy": "SYSTEM_USER",
          "createdTs": "2020-07-08T13:15:57.396Z",
          "lastModifiedBy": "userId: 161",
          "lastModifiedTs": "2021-03-17T10:20:28.894Z",
          "version": 6
        }
      },
      "provisionings": []
    }
  ]
}
```

4.1.3 Create a secure object

```
POST /rtp/secure-objects
```

This API endpoint allows you to create a new secure object in EdgeLock 2GO. In the request JSON body, you should specify the type of secure object that you want to create (*MASTER_KEY, HMAC_KEY, STATIC_PUBLIC_KEY, KEYPAIR, CERTIFICATE, BINARY_FILE*), the secure object name and OID (1 up to 8 hexadecimal characters, unique per device), the policies to apply to the secure object and other parameters that depend on the type of secure object that is being created. For instance, when creating a key pair secure object, it is necessary to specify the algorithm used to generate the key

pair. If the parameters of the request have been correctly specified, a new secure object is created in the EdgeLock 2GO platform and a unique ID is assigned to it. You can get the secure object ID from the JSON response (`id` field) and use it in successive API calls when needed.

Note: the OID of a secure object must be unique per device. Secure objects with the same OID cannot be assigned to the same device group.

Note: please refer to Section 4.5.17 to retrieve the OID validation rules enforced by EdgeLock 2GO.

Six example calls, one for each type of secure object, are reported below. For simplicity, the default usage policy is applied to all objects. Please refer to Section 9 to learn more about policies and how to use the API to configure them.

4.1.3.1 Create keypair secure object

An example call for the creation of a key pair secure object is shown below. You must specify the algorithm used for the generation of the key pair among the ones supported by the platform (NIST_P192, NIST_P224, NIST_P256, NIST_P384, NIST_P521, RSA_1024, RSA_2048, RSA_3072, RSA_4096, ECC_ED_25519, ECC_MONT_DH_25519, ECC_MONT_DH_448, BRAINPOOLP160R1, BRAINPOOLP192R1, BRAINPOOLP224R1, BRAINPOOLP256R1, BRAINPOOLP320R1, BRAINPOOLP384R1, BRAINPOOLP512R1, SECP160K1, SECP192K1, SECP224K1, SECP256K1).

Request

```
cURL Command:
curl -X POST "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/secure-
objects"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
     -H "Content-Type: application/json"
     -d "[escaped JSON request body, see Section 3]"
```

```
JSON request body:
{
  "name": "MyNewKeyPair",
  "algorithm": "NIST_P384",
  "secureObjectType": "KEYPAIR",
  "objectId": "83000007",
  "generateOnDeviceConnection": false,
  "policySourceType": "DEFAULT",
  "policies": null
}
```

Response

```
{
  "secureObjectType": "KEYPAIR",
  "id": 5424,
  "name": "MyNewKeyPair",
  "objectId": "83000007",
  "policySourceType": "DEFAULT",
  "generateOnDeviceConnection": false,
  "deviceGroupCount": null,
  "provisioningCount": null,
  "downloadedCount": null,
  "resourceMetadata": {
    "createdBy": "apiKey: RTP",
    "createdTs": "2021-03-16T09:56:31.665Z",
    "lastModifiedBy": "apiKey: RTP",
    "lastModifiedTs": "2021-03-16T09:56:31.665Z",
    "version": 0
  },
  "algorithm": "NIST_P384"
}
```

4.1.3.2 Create X.509 certificate secure object

An example call for the creation of an X.509 certificate secure object is shown below. To create an X.509 certificate secure object, you must first create a key pair to associate to the certificate as shown in Section 4.1.3.1. You must also create an intermediate CA for signing the certificate as described in Section 4.6.1 or in Section 4.2. The key pair ID and the intermediate CA ID must be specified in the JSON body of the request. Finally, at least the common name and validity of the certificate must be specified. Please refer to the OpenAPI specification file for the complete set of parameters that can be set.

***Note**: using an intermediate certificate with a key strength that is lower that the key pair secure object is not recommended. Please refer to Section 4.6.8 to obtain the list of supported and recommended intermediate CA and key pair algorithm combinations.*

Request

cURL Command:

```
curl -X POST "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/secure-
objects"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
     -H "Content-Type: application/json"
     -d "[escaped JSON request body, see Section 3]"
```

JSON request body:

```
{
  "name": "MyNewCertificate",
  "secureObjectType": "CERTIFICATE",
  "keyPairId":5424,
  "intermediateCaId":474,
  "commonNamePrefix":"myCommonName",
  "objectId": "83000008",
  "certificateValidity":{
      "validityType":"MONTH",
      "spanInMonths":120
   },
  "allowSigning":true,
  "generateOnDeviceConnection": false,
  "policySourceType": "DEFAULT"
}
```

Response


```
{
  "secureObjectType": "CERTIFICATE",
  "id": 5426,
  "name": "MyNewCertificate",
  "objectId": "83000008",
  "generateOnDeviceConnection": false,
  "policySourceType": "DEFAULT",
  "deviceGroupCount": null,
  "provisioningCount": null,
  "downloadedCount": null,
  "resourceMetadata": {
    "createdBy": "SYSTEM_USER",
    "createdTs": "2021-03-16T10:19:13.279Z",
    "lastModifiedBy": "SYSTEM_USER",
    "lastModifiedTs": "2021-03-16T10:19:13.279Z",
    "version": 0
  },
  "keyPairName": "MyNewKeyPair",
  "intermediateCaName": "MySigningCertificate",
  "commonNamePrefix": "myCommonName",
  "certificateValidity": {
    "validityType": "MONTH",
    "spanInMonths": 120
  },
  "country": null,
  "state": null,
  "locality": null,
  "organization": null,
  "organizationUnit": null,
  "email": null,
  "allowSigning": true,
  "keyUsageExt": null,
  "extendedKeyUsageExt": null
}
```


4.1.3.3 Create AES master key secure object

An example call for the creation of an AES master key secure object is shown below. To create an AES master key secure object, you must first encrypt the AES key that you want to import using the EdgeLock 2GO public key and sign the encrypted key using your own public key. The PGP encrypted and signed AES key and your public key must be included in the JSON body of the request. Please refer to Section 12 for detailed instructions to sign and encrypt the AES master key using Kleopatra.

***Note**: The PGP public key (`customerPgpPublicKey`) and the encrypted AES key (`encryptedKey`) must be formatted as described in Section 11 before adding them to the request body.*

Request

cURL Command:

```
curl -X POST "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/secure-
objects"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
     -H "Content-Type: application/json"
     -d "[escaped JSON request body, see Section 3]"
```

JSON request body:

```
{
  "name": "MyNewAESMasterKey",
  "secureObjectType": "MASTER_KEY",
  "objectId": "83000009",
  "keySize":"AES256",
  "customerPgpPublicKey": "<your_PGP_public_key>",
  "encryptedKey": "<PGP_encrypted_AES_key>",
  "generateOnDeviceConnection": false,
  "policySourceType": "DEFAULT"
}
```

Response

```
{
  "secureObjectType": "MASTER_KEY",
  "id": 5324,
  "name": "MyNewAESMasterKey",
  "objectId": "83000009",
  "generateOnDeviceConnection": false,
  "policySourceType": "DEFAULT",
  "deviceGroupCount": null,
  "provisioningCount": null,
  "downloadedCount": null,
  "resourceMetadata": {
    "createdBy": "apiKey: RTP",
    "createdTs": "2021-03-16T09:56:31.665Z",
    "lastModifiedBy": "apiKey: RTP",
    "lastModifiedTs": "2021-03-16T09:56:31.665Z",
    "version": 0
  },
  "algorithm": "AES",
  "length": 256,
  "imported": true
}
```

4.1.3.4 Create HMAC master key secure object

An example call for the creation of an HMAC master key secure object is shown below. To create an HMAC master key secure object, you must first encrypt the HMAC key that you want to import using the EdgeLock 2GO public key and sign the encrypted key using your own public key. The PGP encrypted and signed HMAC key and your public key must be included in the JSON body of the request. Please refer to Section 12 for detailed instructions to sign and encrypt the HMAC master key using Kleopatra.

Note: *the HMAC key size (`keySize` parameter in the request) must be provided in bytes (1 up to 256 bytes).*

Note*: The PGP public key (`customerPgpPublicKey`) and the encrypted HMAC key (`encryptedKey`) must be formatted as described in Section 11 before adding them to the request body.*

Request

cURL Command:

```
curl -X POST "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/secure-objects"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
     -H "Content-Type: application/json"
     -d "[escaped JSON request body, see Section 3]"
```

JSON request body:

```
{
  "name": "MyNewHMACMasterKey",
  "secureObjectType": "HMAC_KEY",
  "objectId": "8300000A",
  "keySize":"256",
  "customerPgpPublicKey": "<your_PGP_public_key>",
  "encryptedKey": "<PGP_encrypted_HMAC_key>",
  "generateOnDeviceConnection": false,
  "policySourceType": "DEFAULT"
}
```

Response

```
{
  "secureObjectType": "HMAC_KEY",
  "id": 5325,
  "name": "MyNewHMACMasterKey",
  "objectId": "8300000A",
  "generateOnDeviceConnection": false,
  "policySourceType": "DEFAULT",
  "deviceGroupCount": null,
  "provisioningCount": null,
  "downloadedCount": null,
  "resourceMetadata": {
    "createdBy": "apiKey: RTP",
    "createdTs": "2021-03-16T09:56:31.665Z",
    "lastModifiedBy": "apiKey: RTP",
    "lastModifiedTs": "2021-03-16T09:56:31.665Z",
    "version": 0
  },
  "length": 256
}
```

4.1.3.5 Create a non-confidential binary file secure object

An example call for the creation of a non-confidential binary file secure object is shown below. To create a non-confidential binary file secure object, you must provide the payload of the binary file in base-64 format. Optionally, you can provide the SHA3-512 checksum of the binary file payload.

Request

cURL Command:

```
curl -X POST "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/secure-
objects"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
     -H "Content-Type: application/json"
     -d "[escaped JSON request body, see Section 3]"
```

JSON request body:

```
{
  "name": "MyNewNonConfidentialBinaryFile",
  "secureObjectType":"BINARY_FILE",
  "objectId":"8300000B",
  "payload":{
      "binaryFileType":"NON_CONFIDENTIAL",
      "binaryFile":"<binary_file_payload>",
      "checkSum":"<optional_SHA3-512_checksum_payload>"
   },
  "generateOnDeviceConnection": false,
  "policySourceType": "DEFAULT"
}
```

Response

```
{
  "secureObjectType": "BINARY_FILE",
  "id": 5427,
  "name": "MyNewNonConfidentialBinaryFile",
  "objectId": "8300000B",
  "generateOnDeviceConnection": false,
  "policySourceType": "DEFAULT",
  "deviceGroupCount": null,
  "provisioningCount": null,
  "downloadedCount": null,
  "resourceMetadata": {
    "createdBy": "apiKey: RTP",
    "createdTs": "2021-03-16T11:07:03.869Z",
    "lastModifiedBy": "apiKey: RTP",
    "lastModifiedTs": "2021-03-16T11:07:03.869Z",
    "version": 0
  },
  "binaryFileType": "NON_CONFIDENTIAL"
}
```

4.1.3.6 Create a confidential binary file secure object

An example call for the creation of a confidential binary file secure object is shown below. To create a confidential binary file secure object, you must first encrypt the binary file that you want to import using the EdgeLock 2GO public key and sign the file using your own public key. The PGP encrypted and signed binary file and your public key must be included in the JSON body of the request. Please refer to Section 12 for detailed instructions to sign and encrypt the confidential binary file using Kleopatra.

***Note**: The PGP public key (`customerPgpPublicKey`) and the encrypted binary file (`encryptedBinaryFile`) must be formatted as described in Section 11 before adding them to the request body.*

Request

```
cURL Command:
curl -X POST "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/secure-
objects"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
     -H "Content-Type: application/json"
     -d "[escaped JSON request body, see Section 3]"
```

```
JSON request body:
{
  "name": "MyNewConfidentialBinaryFile",
  "secureObjectType":"BINARY_FILE",
  "objectId":"8300000C",
  "payload":{
      "binaryFileType":"CONFIDENTIAL",
      "customerPgpPublicKey":"<your_PGP_public_key>",
      "encryptedBinaryFile":"<PGP_encrypted_binary_file_payload>"
   },
  "generateOnDeviceConnection": false,
  "policySourceType": "DEFAULT"
}
```

Response


```
{
  "secureObjectType": "BINARY_FILE",
  "id": 5428,
  "name": "MyNewConfidentialBinaryFile",
  "objectId": "8300000C",
  "generateOnDeviceConnection": false,
  "policySourceType": "DEFAULT",
  "deviceGroupCount": null,
  "provisioningCount": null,
  "downloadedCount": null,
  "resourceMetadata": {
    "createdBy": "apiKey: RTP",
    "createdTs": "2021-03-16T11:07:03.869Z",
    "lastModifiedBy": "apiKey: RTP",
    "lastModifiedTs": "2021-03-16T11:07:03.869Z",
    "version": 0
  },
  "binaryFileType": "CONFIDENTIAL"
}
```


4.1.3.7 Create a static public key secure object

An example call for the creation of a static public key secure object is shown below. To create a static public key secure object, you must provide the key algorithm, the public key that you want to import in PEM format and optionally the SHA3-512 checksum of the public key. Supported algorithms are the same listed in Section 4.1.3.1.

***Note**: The static public key (`publicKey`) must be formatted as described in Section 11 before adding it to the request body.*

Request

```
cURL Command:
curl -X POST "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/secure-
objects"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
     -H "Content-Type: application/json"
     -d "[escaped JSON request body, see Section 3]"
```

```
JSON request body:
{
  "name": "MyNewStaticPublicKey",
  "secureObjectType":"STATIC_PUBLIC_KEY",
  "objectId":"8300000D",
  "publicKey": "<public_key_PEM_format>",
  "checksum": "<optional_SHA3-512_checksum>",
  "algorithm":"RSA_2048",
  "generateOnDeviceConnection": false,
  "policySourceType": "DEFAULT"
}
```

Response

```
{
   "secureObjectType":"STATIC_PUBLIC_KEY",
   "id":5428,
   "name":"MyNewStaticPublicKey",
   "objectId":"8300000D",
   "generateOnDeviceConnection": false,
   "policySourceType": "DEFAULT",
   "deviceGroupCount":null,
   "provisioningCount":null,
   "downloadedCount":null,
   "resourceMetadata":{
      "createdBy":"userId: 161",
      "createdTs":"2021-03-16T11:41:27.097Z",
      "lastModifiedBy":"userId: 161",
      "lastModifiedTs":"2021-03-16T11:41:27.097Z",
      "version":0
   },
   "algorithm":"RSA_2048",
   "publicKey":"<public_key_PEM_format>"
}
```

4.1.4 Assign a secure object to a device group

```
POST /rtp/secure-objects/{secureObjectId}/assign-device-group
```

In EdgeLock 2GO, you can assign secure objects to your device groups so you can easily provision and manage multiple devices with the same configuration. This API endpoint allows you to associate one or more device groups to a secure object. To associate a secure object to one or more device groups, you should specify as a path parameter the ID of the secure object obtained in Section 4.1.3. The JSON body of the request must contain the device group ID obtained in Section 4.1.1. If the request is successful, your devices should now be ready to be provisioned with the secure object through the EdgeLock 2GO, which will occur as soon as the devices are turned on and connect to EdgeLock 2GO.

***Note**: before assigning the secure object to the device group, check that the secure object has a usage policy that matches the hardware type of the device group. See Section 9 for more information.*

An example call is reported below:

Request

cURL Command:

```
curl -X POST "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/secure-
objects/<secureObjectId>/assign-device-group"
     -H "accept: */*"
     -H "OSF-TOKEN: <token>"
     -H "Content-Type: application/json"
     -d "[escaped JSON request body, see Section 3]"
```

JSON request body:

```
{
  "deviceGroupId": "123"
}
```

Response

```
204 No content
```

4.2 API flow: Uploading an external intermediate certificate

This section details the configuration steps required to upload to EdgeLock 2GO an intermediate certificate signed by an external Certificate Authority (CA) using the EdgeLock 2GO REST API. This intermediate CA can then be used to sign X.509 certificate secure objects. To generate an intermediate certificate using NXP root CA, please refer to Section 4.6.1.

The process consists of the following steps:

1. Create a Certificate Signing Request (CSR) for the intermediate certificate
2. Retrieve the CSR value
3. Upload intermediate certificate signed by your external CA

The full API flow is schematized in Figure 3.

Note: Make sure that the API key you are using has all the permissions required to execute the API calls listed in this section.

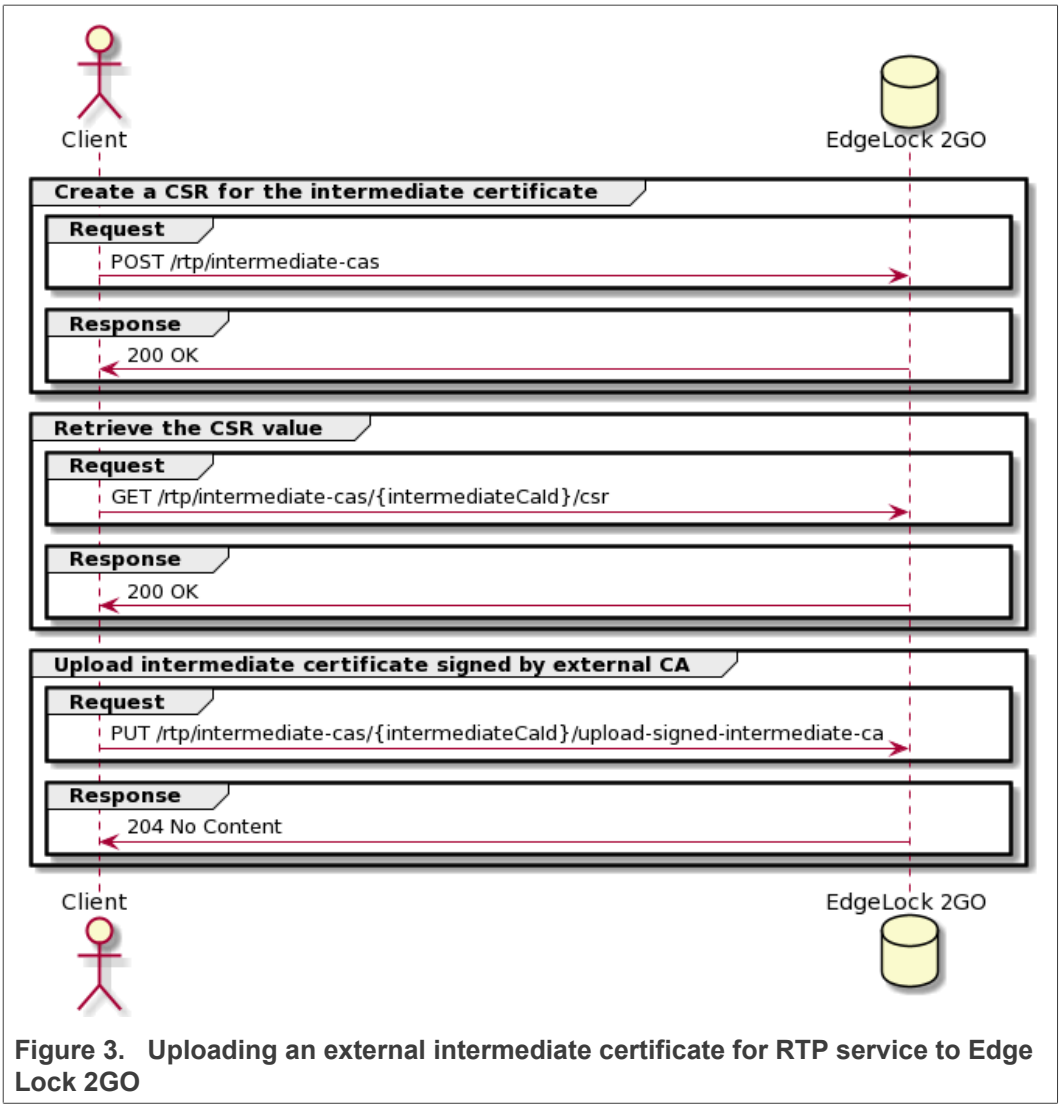

Figure 3. Uploading an external intermediate certificate for RTP service to Edge Lock 2GO

4.2.1 Create a CSR for the intermediate certificate

```
POST /rtp/intermediate-cas
```

EdgeLock 2GO allows you to associate to an X.509 certificate secure object an intermediate certificate signed by an external CA instead of the default NXP root CA. The first step of this process consists of the generation of a CSR. The CSR is a message that you can present to a CA to apply for a digital certificate and that contains the public key of the applicant and other relevant data used for identification. This API endpoint allows you to generate a CSR in EdgeLock 2GO. The key pair used to generate the CSR is kept secure inside EdgeLock 2GO so you don't have to worry about storing it securely in your own premises. If the request is successful, you will obtain the unique ID of the CSR in EdgeLock 2GO so you can use it in the successive API calls.

Note: the generated CSR is only available in PEM format.

Note: the certificate can be generated using one of the following algorithms: NIST_P256, NIST_P384, NIST_P521, RSA_2048, RSA_4096

An example call is reported below:

Request

cURL Command:

```
curl -X POST "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/
intermediate-cas"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
     -H "Content-Type: application/json"
     -d "[escaped JSON request body, see Section 3]"
```

JSON request body:

```
{
  "algorithm": "NIST_P256",
  "name": "testCsr",
  "requestType": "WITH_CSR",
  "commonName": "testCsr",
  "organization": "myOrg",
  "orgUnit": "myUnit",
  "country": "AT",
  "locality": "myCity",
  "state": "myState"
}
```

Response

```
{
  "id": 2307,
  "algorithm": "NIST_P256",
  "name": "testCSR1",
  "value": null,
  "commonName": "testCsr",
  "organization": "myOrg",
  "orgUnit": "myUnit",
  "country": "AT",
  "locality": "myCity",
  "state": "myState",
  "signingState": "CSR_CREATED",
  "signingAuthority": "EXT",
  "validUntil": null,
  "hasAuthorityKeyIdentifier": false,
  "hasSubjectKeyIdentifier": false,
  "resourceMetadata": {
    "createdBy": "apiKey: RTP",
    "createdTs": "2021-03-15T15:27:50.091Z",
    "lastModifiedBy": "apiKey: RTP",
    "lastModifiedTs": "2021-03-15T15:27:50.215Z",
    "version": 1
  }
}
```

4.2.2 Retrieve the CSR value

```
GET /rtp/intermediate-cas/{intermediateCaId}/csr
```

Once the CSR has been created in EdgeLock 2GO as described in Section 4.2.1, you need to retrieve its value and present it to an external CA. The CA will then evaluate the CSR content and generate the corresponding certificate. How to do this is beyond the scope of the present document. This API endpoint returns the value of a CSR resource uniquely identified by a unique ID. The CSR value is in PEM format.

Note: make sure to remove special escaped characters, e.g. /n, from the obtained CSR value before presenting it to the external CA.

Note: an error is returned if the intermediate certificate ID corresponds to a certificate signed by the NXP root CA.

An example call is reported below:

Request

cURL Command:

```
curl -X GET "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/
intermediate-cas/{intermediateCaId}/csr"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
```

JSON request body:

No input parameters

Response

```
{
  "value": "<CSR in PEM format>"
}
```

4.2.3 Upload intermediate certificate signed by external CA

```
PUT /rtp/intermediate-cas/{intermediateCaId}/upload-signed-
intermediate-ca
```

Once you have obtained the intermediate certificate signed by your CA using the CSR value obtained in Section 4.2.2, you can upload it to EdgeLock 2GO and start using it to sign X.509 certificate secure objects. This API endpoint allows you to upload an intermediate certificate signed by an external CA. The `certificateId` path parameter corresponds to the unique ID of the CSR resource obtained in Section 4.2.1. In the body of the request, you should specify the value of the intermediate certificate as well as the value of the root CA certificate that was used to sign the intermediate certificate. Both certificates should be included in the request in PEM format. If the request is successful, you can associate the intermediate certificate to an X.509 certificate secure object as described in Section 4.1.3.

Note: the certificates must be included in the request in one single line. Special characters must be escaped.

Note: the intermediate certificate that you upload must be an X.509 certificate in PEM format respecting the requisites listed in Section 6.

An example call is reported below:

Request

```
cURL Command:
curl -X PUT "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/
intermediate-cas/<intermediateCaId>/upload-signed-intermediate-ca"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
     -H "Content-Type: application/json"
     -d "[escaped JSON request body, see Section 3]"
```

```
JSON request body:
{
  "signedCertificate": "<Intermediate certificate in PEM format>",
  "rootCa": "<External CA root certificate in PEM format>"
}
```

Response

```
204 No content
```

4.3 Additional API endpoints for device group management

This section lists and describes the remaining EdgeLock 2GO API endpoints for management of devices and device groups.

Note: Make sure that the API key you are using has all the permissions required to execute the API calls listed in this section.

4.3.1 Retrieve products

```
GET /products
```

This API endpoint returns the list of products supported by your EdgeLock 2GO instance. Each product is uniquely identified by a 12NC code. This code is used as parameter in

many EdgeLock 2GO API endpoints described in this document. The 12NC code can be retrieved from the `nc12` parameter of the JSON response.

Note: You can filter products by hardware type (`hardwareType`), hardware variant (`hardwareVariant`), 12NC (`nc12`) and device ID (`deviceId`). If you filter by 12NC or device ID, either zero results are returned (12NC or device ID do not exist) or one single result is returned. Please refer to Section 7 and the OpenAPI documentation for the query parameters that can be used for this purpose.

An example call is shown below:

Request

```
cURL Command:
curl -X GET "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/products"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
```

```
JSON request body:
No input devices
```

Response


```
[
  {
    "nc12": "935386987471",
    "commercialName": "SE050A1HQ1/Z01SC",
    "hardwareType": "SE050",
    "hardwareVariant": "SE050A1",
    "hardwareFamilyType": "SE05",
    "softwareVersion": "0",
    "profiles": null,
    "customers": null
  },
  {
    "nc12": "935386984472",
    "commercialName": "SE050A2HQ1/Z01SH",
    "hardwareType": "SE050",
    "hardwareVariant": "SE050A2",
    "hardwareFamilyType": "SE05",
    "softwareVersion": "0",
    "profiles": null,
    "customers": null
  },
  {
    "nc12": "935398293472",
    "commercialName": "SE050C2HQ1/Z01LM",
    "hardwareType": "SE050",
    "hardwareVariant": "SE050C2",
    "hardwareFamilyType": "SE05",
    "softwareVersion": "0",
    "profiles": null,
    "customers": null
  },
  Note: other products have been removed for conciseness.
]
```


4.3.2 Retrieve hardware types

```
GET /hardware-types
```

This API endpoint returns the list of all hardware types supported by EdgeLock 2GO. For each hardware type (e.g. SE050), hardware variants are listed (e.g. SE050A1, SE050C2, etc.). For each hardware variant, associated products are also listed.

An example call is shown below:

Request

cURL Command:

```
curl -X GET "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/hardware-
types"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
```

JSON request body:

```
No input devices
```

Response

```
[
  {
    "name": "SE050",
    "hardwareVariants": [
      {
        "name": "SE050A1",
        "products": [
          {
            "nc12": "935386987471",
            "commercialName": "SE050A1HQ1/Z01SC",
            "hardwareType": "SE050",
            "hardwareVariant": "SE050A1",
            "hardwareFamilyType": "SE05",
            "softwareVersion": "0"
          },
          {
            "nc12": "121909132101",
            "commercialName": "SE050A1HQ1/Z01SF",
            "hardwareType": "SE050",
            "hardwareVariant": "SE050A1",
            "hardwareFamilyType": "SE05",
            "softwareVersion": "0"
          },
          {
            "nc12": "935381378118",
            "commercialName": "SE050A1HQ1/Z01SC",
            "hardwareType": "SE050",
            "hardwareVariant": "SE050A1",
            "hardwareFamilyType": "SE05",
            "softwareVersion": "0"
          }
        ]
        Note: other hardware types have been removed for conciseness.
      }
    ]
  }
]
```

4.3.3 Retrieve development boards

```
GET /dev-boards
```

This API endpoint returns the list of development board products supported by EdgeLock 2GO. Each product is uniquely identified by a 12NC code. This code is used as parameter in many EdgeLock 2GO API endpoints described in this document. The 12NC code can be retrieved from the `nc12` parameter of the JSON response.

An example call is shown below:

Request

```
cURL Command:
curl -X GET "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/dev-boards"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
```

```
JSON request body:
No input devices
```

Response


```
[
  {
    "name": "SE050ARD",
    "color": "black",
    "product": {
      "nc12": "935389312472",
      "commercialName": "SE050C2HQ1/Z01V3",
      "hardwareType": "SE050",
      "hardwareVariant": "SE050C2",
      "hardwareFamilyType": "SE05",
      "softwareVersion": "0"
    }
  },
  {
    "name": "SE051ARD",
    "color": "white",
    "product": {
      "nc12": "935414457472",
      "commercialName": "SE051C2HQ1/Z01XD",
      "hardwareType": "SE051",
      "hardwareVariant": "SE051C2",
      "hardwareFamilyType": "SE05",
      "softwareVersion": "1"
    }
  }
]
```


4.3.4 Retrieve device groups

```
GET /products/{12nc}/device-groups
```

This API endpoint allows you to retrieve the details of the device groups created in EdgeLock 2GO platform. This endpoint can be useful to retrieve the IDs of your device groups.

Note: You can filter device groups by name (`name`). You can also choose the number of records per page that you want to retrieve and, if the number of elements surpasses the number of records, you can specify the page you want to retrieve. Please refer to Section 7 and the OpenAPI documentation for the query parameters that can be used for this purpose.

Note: The selection of the correct hardware 12NC is fundamental for the correct execution of the request. You can retrieve the 12NC using the API endpoint described in Section 4.3.1.

An example call is reported below:

Request

cURL Command:

```
curl -X GET "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/products/
<12nc>/device-groups"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
```

JSON request body:

```
No input parameters
```

Response


```
{
  "current": "https://api.foundries.io/ota/factories/<factory>/el2g-
proxy/products/120405220204/device-groups?name=*&page=0&size=20",
  "next": null,
  "prev": null,
  "totalElements": 2,
  "pageType": "DeviceGroupPage",
  "content": [
    {
      "id": 121,
      "deviceGroupName": "mydevicegroup",
      "deviceCount": 1,
      "productRelationship": "USER",
      "productDetails": {
        "nc12": "935414457472",
        "commercialName": "SE051C2HQ1/Z01XD",
        "hardwareType": "SE051",
        "hardwareVariant": "SE051C2",
        "hardwareFamilyType": "SE05",
        "softwareVersion": "1"
      },
      "resourceMetadata": {
        "createdBy": "userId: 3636",
        "createdTs": "2021-09-27T10:35:23.887Z",
        "lastModifiedBy": "userId: 3636",
        "lastModifiedTs": "2021-09-27T10:35:23.887Z",
        "version": null
      },
      "claimCodeOverviews": null
    },
    {
      "id": 97,
      "deviceGroupName": "mydevicegroup2",
      "deviceCount": 0,
      "productRelationship": "USER",
      "productDetails": {
        "nc12": "935414457472",
        "commercialName": "SE051C2HQ1/Z01XD",
        "hardwareType": "SE051",
        "hardwareVariant": "SE051C2",
        "hardwareFamilyType": "SE05",
        "softwareVersion": "1"
      }
      "resourceMetadata": {
        "createdBy": "userId: 3636",
        "createdTs": "2021-09-27T10:35:23.887Z",
        "lastModifiedBy": "userId: 3636",
        "lastModifiedTs": "2021-09-27T10:35:23.887Z",
        "version": null
      },
      "claimCodeOverviews": null
    }
  ]
}
```


4.3.5 Retrieve device group by ID

```
GET /products/{12nc}/device-groups/{deviceGroupId}
```

This API endpoint returns the details of a device group uniquely identified by its device group ID. The device group ID is assigned during the creation of the device group (see Section 4.1.1).

Note: The selection of the correct hardware 12NC is fundamental for the correct execution of the request. You can retrieve the 12NC using the API endpoint described in Section 4.3.1.

An example call is reported below:

Request

```
cURL Command:
curl -X GET "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/products/
<12nc>/device-groups/<device_group_id>"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
```

```
JSON request body:
No input parameters
```

Response


```
{
    "id": 474,
    "deviceGroupName": "NewGroup",
    "deviceCount": 2,
    "productRelationship": "USER",
    "productDetails": {
        "nc12": "121909132103",
        "commercialName": "SE050C1HQ1/Z01SF",
        "hardwareType": "SE050",
        "hardwareVariant": "SE050C1",
        "hardwareFamilyType": "SE05",
        "softwareVersion": "0"
    },
    "resourceMetadata": {
        "createdBy": "apiKey: ApiKey",
        "createdTs": "2021-09-28T08:33:06.5Z",
        "lastModifiedBy": "apiKey: ApiKey",
        "lastModifiedTs": "2021-09-28T08:33:06.5Z",
        "version": null
     },
     "claimCodeOverviews": null
}
```


4.3.6 Retrieve devices

```
GET /devices
```

This API endpoint returns the list of devices that have been registered in EdgeLock 2GO.

Note: You can filter devices by device ID (`device-id`) and product type (`producttype`). You can also choose the number of records per page that you want to retrieve and, if the number of elements surpasses the number of records, you can specify the page you want to retrieve. Please refer to Section 7 and the OpenAPI documentation for the query parameters that can be used for this purpose.

An example call is reported below:

Request

```
cURL Command:
curl -X GET "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/devices"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
```

```
JSON request body:
No input parameters
```

Response

```
{
  "current": "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/devices?
page=0&size=20",
  "next": null,
  "prev": null,
  "totalElements": 1,
  "pageType": "DevicePage",
  "content": [
    {
      "id": "348555491056936142262187597632590342062080",
      "deviceGroupId": 474,
      "deviceGroupName": "NewGroup",
      "productDetail": null,
      "lastConnection": null,
      "resourceMetadata": {
        "createdBy": "SYSTEM_USER",
        "createdTs": "2020-08-21T13:58:17.664Z",
        "lastModifiedBy": "SYSTEM_USER",
        "lastModifiedTs": "2020-08-21T13:58:17.664Z",
        "version": 0
      }
    }
  ]
}
```

4.3.7 Retrieve device data

```
GET /products/{12nc}/device-groups/{deviceGroupId}/devices/
{deviceId}
```

This API endpoint allows you to retrieve the data of the device with the specified device ID (device UID) within the group identified by the specified device group ID.

Note: The selection of the correct hardware 12NC is fundamental for the correct execution of the request. You can retrieve the 12NC using the API endpoint described in Section 4.3.1.

An example call is reported below:

Request

```
cURL Command:
curl -X GET "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/products/
<12nc>/device-groups/<device_group_id>/devices/<device_id>"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
```

```
JSON request body:
No input parameters
```

Response

```
{
  "id": "348555491056936142262187597632590342062080",
  "deviceGroupId": 474,
  "deviceGroupName": "NewGroup",
  "productDetail": null,
  "lastConnection": null,
  "resourceMetadata": {
    "createdBy": "SYSTEM_USER",
    "createdTs": "2020-07-08T13:15:57.396Z",
    "lastModifiedBy": "userId: 161",
    "lastModifiedTs": "2021-03-17T10:20:28.894Z",
    "version": 6
  }
}
```

4.3.8 Retrieve devices assigned to a device group

```
GET /products/{12nc}/device-groups/{deviceGroupId}/devices
```

This API endpoint allows you to retrieve a list of devices assigned to a device group identified by a unique device group ID. If the request is successful, the endpoint returns an array of devices, identified by device UID, assigned to the specified device group. Each entry in the array contains the device data and a list of provisionings associated to the device. A provisioning in EdgeLock 2GO is defined as the association of a service, a device group and a device belonging to the device group. A provisioning has a provisioning state and a unique provisioning ID.

Note: You can choose the number of records per page that you want to retrieve and, if the number of elements surpasses the number of records, you can specify the page you want to retrieve. Please refer to Section 7 and the OpenAPI documentation for the query parameters that can be used for this purpose.

Note: The selection of the correct hardware 12NC is fundamental for the correct execution of the request. You can retrieve the 12NC using the API endpoint described in Section 4.3.1.

An example call is reported below:

Request

cURL Command:

```
curl -X GET "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/products/
<12nc>/device-groups/<device_group_id>/devices"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
```

JSON request body:

No input parameters

Response


```
{
  "current": "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/products/120405220204/
device-groups/474/devices?page=0&size=20",
  "next": null,
  "prev": null,
  "totalElements": 1,
  "pageType": "DeviceServiceProvisioningPage",
  "content": [
    {
      "device": {
        "id": "348555491056936142262187597632590342062080",
        "deviceGroupId": 474,
        "deviceGroupName": "NewGroup",
        "productDetail": null,
        "lastConnection": null,
        "resourceMetadata": {
          "createdBy": "SYSTEM_USER",
          "createdTs": "2020-08-21T13:58:17.664Z",
          "lastModifiedBy": "SYSTEM_USER",
          "lastModifiedTs": "2020-08-21T13:58:17.664Z",
          "version": 0
        }
      }
  ]
}
```


4.3.9 Retrieve product of a device

```
GET /devices/{deviceId}/product
```

This API endpoint returns the product information of a single device identified by its unique device ID.

An example call is reported below:

Request

cURL Command:

```
curl -X GET "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/devices/
{deviceId}/product"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
```

JSON request body:

```
No input parameters
```

Response


```
{
    "nc12": "120405220204",
    "commercialName": "SE050C1HQ1/Z01SC",
    "hardwareType": "SE050",
    "hardwareVariant": "SE050C1",
    "hardwareFamilyType": "SE05",
    "softwareVersion": "0"
}
```


4.3.10 Unclaim a list of devices

```
POST /products/{12nc}/device-groups/{deviceGroupId}/devices/
unclaim
```

This API endpoint allows you to unclaim a given list of devices belonging to the device group identified by the unique device group ID. The list of UIDs of the devices to unclaim should be placed in an array in the JSON request body. Unclaimed devices are moved

from the customer device group to the NXP generic device group and can be later assigned to other device groups. To extract the UUID of your device, refer to Section 8.

Note: The selection of the correct hardware 12NC is fundamental for the correct execution of the request. You can retrieve the 12NC using the API endpoint described in Section 4.3.1.

An example call is reported below:

Request

```
cURL Command:
curl -X POST "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/products/
<12nc>/device-groups/<device_group_id>/devices/unclaim"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
     -H "Content-Type: application/json"
     -d "[escaped JSON request body, see Section 3]"
```

```
JSON request body:
{
  "deviceIds": [
    "348555491056936142262187597632590342062080"
  ]
}
```

Response

```
{
  "successfulDeviceIds": [
    "348555491056936142262187597632590342062080"
  ],
  "failedDeviceIds": []
}
```

4.3.11 Unclaim all devices in a group

```
POST /products/{12nc}/device-groups/{deviceGroupId}/unclaim
```

This API endpoint allows you to unclaim all the devices belonging to the device group identified by the unique device group ID. Unclaimed devices are moved from the customer device group to the NXP generic device group and can be later assigned to other device groups.

Note: The selection of the correct hardware 12NC is fundamental for the correct execution of the request. You can retrieve the 12NC using the API endpoint described in Section 4.3.1.

An example call is reported below:

Request

```
cURL Command:
curl -X POST "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/products/
<12nc>/device-groups/<device_group_id>/unclaim"
     -H "accept: */*"
     -H "OSF-TOKEN: <token>"
     -H "Content-Type: application/json"
     -d "[escaped JSON request body, see Section 3]"
```

```
JSON request body:
No request body
```

Response

```
202 Accepted
```

4.3.12 Delete a device group

```
DELETE /products/{12nc}/device-groups/{deviceGroupId}
```

This API endpoint allows you to delete from EdgeLock 2GO a device group identified by a unique device group ID. The device group ID is assigned during the creation of the device group (see Section 4.1.1).

Note: a device group can only be deleted if it has no devices, services or claim codes assigned.

Note: The selection of the correct hardware 12NC is fundamental for the correct execution of the request. You can retrieve the 12NC using the API endpoint described in Section 4.3.1.

An example call is reported below:

Request

```
cURL Command:
curl -X DELETE "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/products/
<12nc>/device-groups/<device_group_id>"
     -H "accept: */*"
     -H "OSF-TOKEN: <token>"
```

```
JSON request body:
No input parameters
```

Response

```
204 No Content
```

4.3.13 Update a device group

```
PUT /products/{12nc}/device-groups/{deviceGroupId}
```

This API endpoint allows you to update some of the data of an existing device group identified by a unique device group ID. Refer to Section 4.3.4 to retrieve the ID of your device groups.

Note: The selection of the correct hardware 12NC is fundamental for the correct execution of the request. You can retrieve the 12NC using the API endpoint described in Section 4.3.1.

An example call is reported below:

Request

```
cURL Command:
curl -X PUT "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/products/
<12nc>/device-groups/<device_group_id>"
     -H "accept: */*"
     -H "OSF-TOKEN: <token>"
     -H "Content-Type: application/json"
     -d "[escaped JSON request body, see Section 3]"
```

```
JSON request body:
{
  "deviceGroupName": "newDeviceGroupName"
}
```

Response

```
204 No Content
```

4.4 Additional API endpoints for claim codes management

This section lists and describes the EdgeLock 2GO API endpoints for management of claim codes.

Note: Make sure that the API key you are using has all the permissions required to execute the API calls listed in this section.

4.4.1 Assign a claim code to a device group

```
POST /products/{12nc}/device-groups/{deviceGroupId}/claim-codes
```

EdgeLock 2GO API allows customers to assign one or more claim codes to a device group denoted by a unique device group ID. A claim code is a secret key created per device group and injected into the device during manufacturing. When the device first connects to EdgeLock 2GO, it presents the claim code. If the claim code is valid, the device is added to the device group. You can find more details about claim codes in Section 10. If the creation of the claim code is successful, a unique ID is assigned to the claim code. The claim code ID is returned in the JSON response.

Note: The selection of the correct hardware 12NC is fundamental for the correct execution of the request. You can retrieve the 12NC using the API endpoint described in Section 4.3.1.

Note: The claim code secret key must be at least 16 characters long and encoded in base64.

An example call is reported below:

Request

```
cURL Command:
curl -X POST "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/products/
<12nc>/device-groups/<device_group_id>/claim-codes"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
     -H "Content-Type: application/json"
     -d "[escaped JSON request body, see Section 3]"
```

```
JSON request body:
{
  "name": "MyClaim",
  "policy": "UNRESTRICTED",
  "behavior": "ALLOW",
  "deviceLimit": 0,
  "reusable": true,
  "secret": "MDEyMzQ1Njc4OTAxMjM0NTY3ODk=",
  "status": "ALL",
  "metadata": null
}
```

Response

```
{
  "name": "MyClaim",
  "policy": "UNRESTRICTED",
  "behavior": "ALLOW",
  "deviceLimit": 0,
  "reusable": true,
  "id": 4,
  "deviceCount": 0,
  "status": "ACTIVE",
  "metadata": {
    "createdBy": "SYSTEM_USER",
    "createdTs": 1580291098.852,
    "lastModifiedBy": "SYSTEM_USER",
    "lastModifiedTs": 1580291098.852,
    "version": 0
  }
}
```

4.4.2 Create a batch of claim codes

```
POST /products/{12nc}/device-groups/{deviceGroupId}/claim-
codes/create-batch
```

EdgeLock 2GO API allows customers to assign one or more claim codes to a device group denoted by a unique device group ID. A claim code is a secret key created per device group and injected into the device during manufacturing. When the device first connects to EdgeLock 2GO, it presents the claim code. If the claim code is valid, the device is added to the device group. You can find more details about claim codes in Section 10. To quickly create a large number of claim codes for a given device group, EdgeLock 2GO allows users to create batches of claim codes. A batch of claim codes consists of a certain number of claim codes (`numberOfClaimCodes` parameter) having the same policy. The claim code secret is automatically generated by EdgeLock 2GO and cannot be specified manually. If the request is successful, a unique claim code batch ID is returned in the JSON response. The batch ID can be used to reference the batch in other API calls.

Note: The selection of the correct hardware 12NC is fundamental for the correct execution of the request. You can retrieve the 12NC using the API endpoint described in Section 4.3.1.

Note: This API can only be used to create between 100 and 1000 claim codes.

An example call is reported below:

Request

```
cURL Command:
curl -X POST "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/products/
<12nc>/device-groups/<device_group_id>/claim-codes/create-batch"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
     -H "Content-Type: application/json"
     -d "[escaped JSON request body, see Section 3]"
```

```
JSON request body:
{
  "namePrefix": "MyClaimCode",
  "numberOfClaimCodes": 100,
  "policy": "UNRESTRICTED",
  "behavior": "ALLOW",
  "deviceLimit": 0,
  "reusable": true
}
```

Response

```
{
  "batchId": 133
}
```

4.4.3 Retrieve details of a batch of claim codes

```
GET /products/{12nc}/device-groups/{deviceGroupId}/claim-codes/
batches/{batchId}
```

This API endpoint allows you to retrieve the details associated to a claim code batch, identified by a unique batch ID, of a device group identified by a unique device group ID. To create a batch of claim codes, please refer to Section 4.4.2.

Note: The selection of the correct hardware 12NC is fundamental for the correct execution of the request. You can retrieve the 12NC using the API endpoint described in Section 4.3.1.

An example call is reported below:

Request

```
cURL Command:
curl -X POST "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/products/
<12nc>/device-groups/<device_group_id>/claim-codes/batches/<batch_id>"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
```

```
JSON request body:
No input parameters
```

Response

```
{
  "batchId": 133,
  "progressInPercentage": 100
}
```

4.4.4 Retrieve claim codes of a device group

```
GET /products/{12nc}/device-groups/{deviceGroupId}/claim-codes
```

This API endpoint allows you to retrieve the details of the claim codes assigned to a device group identified by a unique device group ID.

Note: The selection of the correct hardware 12NC is fundamental for the correct execution of the request. You can retrieve the 12NC using the API endpoint described in Section 4.3.1.

Note: You can filter claim codes by name (`name`), status (`status` - active, revoked or all) and usage status (`usageStatus` - reached, unreached or all). You can also choose the number of records per page that you want to retrieve and, if the number of elements surpasses the number of records, you can specify the page you want to retrieve. Finally, you can specify if you want to sort the results using a particular field and if those results should appear in ascending or descending order. Please refer to Section 7 and the OpenAPI documentation for the query parameters that can be used for this purpose.

An example call is reported below:

Request

cURL Command:

```
curl -X GET "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/products/
<12nc>/device-groups/<device_group_id>/claim-codes"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
```

JSON request body:

```
No input parameters
```

Response


```
{
  "current": "https://api.foundries.io/ota/factories/<factory>/
el2g-proxy/products/120405220204/device-groups/130/claim-codes?
usageStatus=ALL&status=ALL&page=0&size=20",
  "totalElements": 2,
  "content": [
    {
      "name": "MyClaim",
      "policy": "UNRESTRICTED",
      "behavior": "ALLOW",
      "deviceLimit": 0,
      "reusable": true,
      "id": 4,
      "secret": null,
      "deviceCount": 0,
      "status": "ACTIVE",
      "metadata": {
        "createdBy": "SYSTEM_USER",
        "createdTs": 1580291098.852,
        "lastModifiedBy": "SYSTEM_USER",
        "lastModifiedTs": 1580291098.852,
        "version": 0
      }
    },
    {
      "name": "MyClaim2",
      "policy": "RESTRICTED",
      "behavior": "DENY",
      "deviceLimit": 3,
      "reusable": true,
      "id": 5,
      "secret": null,
      "deviceCount": 0,
      "status": "ACTIVE",
      "metadata": {
        "createdBy": "SYSTEM_USER",
        "createdTs": 1580291188.56,
        "lastModifiedBy": "SYSTEM_USER",
        "lastModifiedTs": 1580291188.56,
        "version": 0
      }
    }
  ]
}
```


4.4.5 Retrieve a claim code by claim code ID

```
GET /products/{12nc}/device-groups/{deviceGroupId}/claim-codes/
{claimCodeId}
```

This API endpoint allows you to retrieve the details of a claim code, denoted by a unique claim code ID, assigned to a device group with a unique device group ID.

Note: The selection of the correct hardware 12NC is fundamental for the correct execution of the request. You can retrieve the 12NC using the API endpoint described in Section 4.3.1.

An example call is reported below:

Request

```
cURL Command:
curl -X GET "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/products/
<12nc>/device-groups/<device_group_id>/claim-codes/<claim_code_id>"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
```

```
JSON request body:
No input parameters
```

Response

```
{
  "name": "MyClaim",
  "policy": "UNRESTRICTED",
  "behavior": "ALLOW",
  "deviceLimit": 0,
  "reusable": true,
  "id": 4,
  "deviceCount": 0,
  "status": "ACTIVE",
  "metadata": {
    "createdBy": "SYSTEM_USER",
    "createdTs": 1580291098.852,
    "lastModifiedBy": "SYSTEM_USER",
    "lastModifiedTs": 1580291098.852,
    "version": 0
  }
}
```

4.4.6 Update a claim code

```
PUT /products/{12nc}/device-groups/{deviceGroupId}/claim-codes/
{claimCodeId}
```

This API endpoint allows you to change some of the properties of a claim code, denoted by a unique claim code ID, that has been assigned to a device group identified by a unique device group ID.

Note: The selection of the correct hardware 12NC is fundamental for the correct execution of the request. You can retrieve the 12NC using the API endpoint described in Section 4.3.1.

An example call is reported below:

Request

```
cURL Command:
curl -X PUT "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/products/
<12nc>/device-groups/<device_group_id>/claim-codes/<claim_code_id>"
     -H "accept: */*"
     -H "OSF-TOKEN: <token>"
     -H "Content-Type: application/json"
     -d "[escaped JSON request body, see Section 3]"
```

```
JSON request body:
{
  "name": "MyNewNameClaim",
  "policy": "RESTRICTED",
  "behavior": "DENY",
  "deviceLimit": 2,
  "reusable": false
}
```

Response

```
204 No Content
```

4.4.7 Revoke a claim code

```
POST /products/{12nc}/device-groups/{deviceGroupId}/claim-
codes/{claimCodeId}/revoke
```

This API endpoint allows you to revoke an existing claim code identified by a unique claim code ID.

Note: The selection of the correct hardware 12NC is fundamental for the correct execution of the request. You can retrieve the 12NC using the API endpoint described in Section 4.3.1.

Note: Once a claim code is revoked, it cannot be activated again. A revoked claim code keeps persisting in the EdgeLock 2GO database, so it is counted in your business quota.

An example call is reported below:

Request

```
cURL Command:
curl -X POST "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/products/
<12nc>/device-groups/<device_group_id>/claim-codes/<claim_code_id>/revoke"
     -H "accept: */*"
     -H "OSF-TOKEN: <token>
```

```
JSON request body:
No input parameters
```

Response

```
204 No Content
```

4.4.8 Retrieve decrypted secret of a claim code

```
GET /products/{12nc}/device-groups/{deviceGroupId}/claim-codes/
{claimCodeId}/decrypt
```

This API endpoint returns the decrypted secret of a claim code identified by a unique claim code ID. The returned secret is base64 encoded.

Note: The selection of the correct hardware 12NC is fundamental for the correct execution of the request. You can retrieve the 12NC using the API endpoint described in Section 4.3.1.

Note: The response is in plain text format.

An example call is reported below:

Request

```
cURL Command:
curl -X GET "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/products/
<12nc>/device-groups/<device_group_id>/claim-codes/<claim_code_id>/decrypt"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>
     -d "[escaped JSON request body, see Section 3]"
```

```
JSON request body:
No input parameters
```

Response

```
Mjc2ZDcyMGYyMS05NTc3LTRhY2ItYTZkMi00NmI0NWU2ZWNlNmY=
```

4.4.9 Delete a single claim code

```
DELETE /products/{12nc}/device-groups/{deviceGroupId}/claim-
codes/{claim-code-id}
```

This API endpoint allows you to delete from EdgeLock 2GO a claim code, identified by a unique claim code ID, belonging to a device group having a unique device group ID.

Note: The action of deleting a claim code is irreversible. A deleted claim code is physically removed from EdgeLock 2GO database and it is not counted in your business quota.

An example call is reported below:

Request

```
cURL Command:
curl -X DELETE "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/products/
<12nc>/device-groups/<device_group_id>/claim-codes/<claim_code_id>"
     -H "accept: */*"
     -H "OSF-TOKEN: <token>"
```

```
JSON request body:
No input parameters
```

Response

```
204 No Content
```

4.4.10 Delete a batch of claim codes

```
POST /claim-codes/batch-delete
```

This API endpoint allows you to delete from EdgeLock 2GO a batch of claim codes. The claim codes to delete, identified by their unique ID, must be specified in the body of the request. You can retrieve the claim code IDs for a given group by using the API call described in Section 4.4.4.

Note: The action of deleting a claim code is irreversible. A deleted claim code is physically removed from EdgeLock 2GO database and it is not counted in your business quota.

An example call is reported below:

Request

```
cURL Command:
curl -X POST "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/claim-codes/
batch-delete"
     -H "accept: */*"
     -H "OSF-TOKEN: <token>"
     -H "Content-Type: application/json"
     -d "[escaped JSON request body, see Section 3]"
```

```
JSON request body:
{
  "claimCodeIds": [
    14874,
    14876,
    14865
  ]
}
```

Response

```
{
  "processedClaimCodeIds": [
    14874,
    14876,
    14865
  ],
  "failedClaimCodeId": []
}
```

4.4.11 Delete claim codes by criteria

```
POST /claim-codes/delete-by-criteria
```

This API endpoint allows you to delete from EdgeLock 2GO claim codes meeting a set of specified criteria. The possible criteria are: revoked claim codes, maximum usage achieved and claim code created before a specified date. If revoked or maximum usage achieved are missing or false, they will not be used as criteria. If more than one criteria is used, they will be combined using the logical AND operator, meaning that only the claim codes that match all criteria are deleted. If there is no valid criteria (revoked and maximum usage achieved are false or not set and createdBefore is not specified), an error will be returned.

Note: The action of deleting a claim code is irreversible. A deleted claim code is physically removed from EdgeLock 2GO database and it is not counted in your business quota.

An example call is reported below:

Request

```
cURL Command:
curl -X POST "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/claim-codes/
batch-delete-by-criteria"
     -H "accept: */*"
     -H "OSF-TOKEN: <token>"
     -H "Content-Type: application/json"
     -d "[escaped JSON request body, see Section 3]"
```

```
JSON request body:
{
  "revoked": true,
  "maxUsageAchieved": false,
  "createdBefore": "2020-10-29T13:40:52.516Z"
}
```

Response

```
{
  "deletedCount": 7
}
```

4.4.12 Retrieve grouped claim codes statistics

```
GET /products/{12nc}/device-groups/{deviceGroupId}/claim-codes/
statistics
```

This API endpoint allows you to retrieve the claim code statistics of a device group having a unique device group ID. The response includes the usage status and number of claim codes in the device group. Claim codes can have a defined maximum number of devices that can be claimed with a specific claim code. The usage status indicates the percentage of devices claimed before a limit is reached (LOW = less than 50%, MEDIUM = between 50% and 90%, HIGH = more than 90%).

Note: The selection of the correct hardware 12NC is fundamental for the correct execution of the request. You can retrieve the 12NC using the API endpoint described in Section 4.3.1.

An example call is reported below:

Request

```
cURL Command:
curl -X GET "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/products/
<12nc>/device-groups/<device_group_id>/claim-codes/statistics"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
```

```
JSON request body:
No input parameters
```

Response

```
{
  "groupedStatistics": [
    {
      "status": "LOW",
      "amount": 200
    }
  ],
  "totalAmount": 200
}
```

4.5 Additional API endpoints for secure object management

This section lists and describes the remaining EdgeLock 2GO API endpoints for management of secure objects.

Note: Make sure that the API key you are using has all the permissions required to execute the API calls listed in this section.

4.5.1 Retrieve device groups that can be assigned to a secure object

```
GET /rtp/secure-objects/{secureObjectId}/available-device-groups
```

This API endpoint allows you to retrieve the list of device groups that can be assigned to the secure object identified by a unique secure object ID.

Note: You can filter device groups by name (`name`). Please refer to Section 7 and the OpenAPI documentation for the query parameters that can be used for this purpose.

An example call is reported below:

Request

cURL Command:

```
curl -X GET "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/secure-objects/<secureObjectId>/available-device-groups"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
```

JSON request body:

```
No input parameters
```

Response

```
[
  {
    "id": 123,
    "deviceGroupName": "Device Group-1"
  },
  {
    "id": 124,
    "deviceGroupName": "Device Group-2"
  },
]
```

4.5.2 Unassign a secure object from a list of device groups

```
POST /rtp/secure-objects/{secureObjectId}/unassign-device-groups
```

In EdgeLock 2GO you can assign secure objects to your device groups so you can easily manage multiple devices. This API endpoint allows you to unassign a secure object from one or more device groups. To unassign a secure object from a device group, you should specify as a path parameter the unique secure object ID. The JSON body of the request must contain the device group IDs of the device groups that you want to unassign in a JSON array.

An example call is reported below:

Request

```
cURL Command:
curl -X POST "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/secure-
objects/<secureObjectId>/unassign-device-groups"
     -H "accept: */*" -H "OSF-TOKEN: <token>"
     -H "Content-Type: application/json"
     -d "[escaped JSON request body, see Section 3]"
```

```
JSON request body:
[
  121,
  125,
  146
]
```

Response

```
204 No content
```

4.5.3 Unassign a list of secure objects from a device group

```
POST /rtp/device-groups/{deviceGroupId}/unassign-secure-objects
```

In EdgeLock 2GO you can assign secure objects to your device groups so you can easily manage multiple devices. This API endpoint allows you to unassign a list of secure objects from a device group. To unassign a list of secure objects from a device group, you should specify as a path parameter the unique device group ID. The JSON body of the request must contain the secure object IDs that you want to unassign in a JSON array.

An example call is reported below:

Request

```
cURL Command:
curl -X POST "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/device-
groups/<deviceGroupId>/unassign-secure-objects"
     -H "accept: */*" -H "OSF-TOKEN: <token>"
     -H "Content-Type: application/json"
     -d "[escaped JSON request body, see Section 3]"
```

```
JSON request body:
[
  456,
  678,
  323
]
```

Response

```
204 No content
```

4.5.4 Retrieve a secure object by ID

```
GET /rtp/secure-objects/{secureObjectId}
```

This API endpoint allows you to retrieve the data of a secure object identified by a unique secure object ID. The secure object ID is assigned by EdgeLock 2GO at secure object creation as shown in Section 4.1.3.

An example call is reported below:

Request

```
cURL Command:
curl -X GET "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/secure-
objects/<secureObjectId>"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
```

```
JSON request body:
No input parameters
```

Response

```
{
  "secureObjectType": "KEYPAIR",
  "secureObjectId": 702,
  "name": "MyKeyPair",
  "objectId": "83000000",
  "policySourceType": "DEFAULT",
  "generateOnDeviceConnection": false,
  "deviceGroupCount": 1,
  "provisioningCount": 1,
  "downloadedCount": 0,
  "resourceMetadata": {
    "createdBy": "userId: 161",
    "createdTs": "2020-08-20T14:15:18.188Z",
    "lastModifiedBy": "userId: 161",
    "lastModifiedTs": "2020-08-20T14:15:18.188Z",
    "version": 0
  },
  "algorithm": "NIST_P256"
}
```

4.5.5 Retrieve all secure objects

```
GET /rtp/secure-objects
```

This API endpoint allows you to obtain a list of all the secure objects created in EdgeLock 2GO. Each secure object is uniquely identified by a secure object ID.

Note: keys and certificates, when provided, are always in PEM format.

Note: You can filter secure objects by name (`name`) and type (`secureObjectType` - MASTER_KEY, HMAC_KEY, STATIC_PUBLIC_KEY, KEYPAIR, CERTIFICATE, BINARY_FILE). You can also choose the number of records per page that you want to retrieve and, if the number of elements surpasses the number of records, you can specify the page you want to retrieve. Finally you can sort results in ascending or descending order based on a field. Please refer to Section 7 and the OpenAPI documentation for the query parameters that can be used for these purposes.

An example call is reported below:

Request

```
cURL Command:
curl -X GET "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/secure-
objects"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
```

```
JSON request body:
No input parameters
```

Response

```
{
  "current": "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/secure-
objects?sortByDirection=DESC&page=0&size=20",
  "next": null,
  "prev": null,
  "totalElements": 2,
  "pageType": "SecureObjectPage",
  "content": [
    {
      "secureObjectType": "STATIC_PUBLIC_KEY",
      "id": 5428,
      "name": "MyNewStaticPublicKey",
      "objectId": "8300000D",
      "policySourceType": "DEFAULT",
      "generateOnDeviceConnection": false,
      "deviceGroupCount": 0,
      "provisioningCount": 0,
      "downloadedCount": 0,
      "resourceMetadata": {
        "createdBy": "userId: 161",
        "createdTs": "2021-03-16T11:41:27.097Z",
        "lastModifiedBy": "userId: 161",
        "lastModifiedTs": "2021-03-16T11:41:27.109Z",
        "version": 1
      },
      "algorithm": "RSA_2048",
      "publicKey": "<public_key_PEM_format>"
    },
    {
      "secureObjectType": "BINARY_FILE",
      "id": 5427,
      "name": "MyNewNonConfidentialBinaryFile",
      "objectId": "8300000B",
      "policySourceType": "CUSTOM",
      "generateOnDeviceConnection": false,
      "deviceGroupCount": 0,
      "provisioningCount": 0,
      "downloadedCount": 0,
      "resourceMetadata": {
        "createdBy": "apiKey: RTP",
        "createdTs": "2021-03-16T11:07:03.869Z",
        "lastModifiedBy": "apiKey: RTP",
        "lastModifiedTs": "2021-03-16T11:07:03.877Z",
        "version": 1
      },
      "binaryFileType": "NON_CONFIDENTIAL"
    }
  ]
}
```

4.5.6 Retrieve secure objects assigned to a device group

```
GET /rtp/device-groups/{deviceGroupId}/assigned-secure-objects
```

This API endpoint allows you to obtain the list of secure objects assigned to a given device group identified by a unique device group ID. Each secure object is uniquely identified by a secure object ID.

Note: You can filter secure objects by name (`secureObjectName`) and type (`secureObjectType` - MASTER_KEY, HMAC_KEY, STATIC_PUBLIC_KEY, KEYPAIR,

CERTIFICATE, BINARY_FILE). You can also choose the number of records per page that you want to retrieve and, if the number of elements surpasses the number of records, you can specify the page you want to retrieve. Finally you can sort results in ascending or descending order based on a field. Please refer to Section 7 and the OpenAPI documentation for the query parameters that can be used for these purposes.

An example call is reported below:

Request

cURL Command:

```
curl -X GET "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/device-groups/<deviceGroupId>/assigned-secure-objects"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
```

JSON request body:

No input parameters

Response


```
{
  "current": "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/device-groups/474/assigned-secure-objects?sortByDirection=DESC&page=0&size=20",
  "next": null,
  "prev": null,
  "totalElements": 2,
  "pageType": "SecureObjectInDeviceGroupPage",
  "content": [
    {
      "secureObjectType": "BINARY_FILE",
      "secureObjectId": 3338,
      "secureObjectName": "MyBinaryFile",
      "objectId": 83000004,
      "deviceGroupId": 474,
      "unassignable": true,
      "secureObjectCreatedTs": "2020-12-14T10:25:23.308Z",
      "resourceMetadata": {
        "createdBy": "userId: 161",
        "createdTs": "2020-12-14T10:25:42.312Z",
        "lastModifiedBy": "userId: 161",
        "lastModifiedTs": "2020-12-14T10:25:42.312Z",
        "version": 0
      }
    },
    {
      "secureObjectType": "KEYPAIR",
      "secureObjectId": 702,
      "secureObjectName": "MyKeyPair",
      "objectId": 83000000,
      "deviceGroupId": 474,
      "unassignable": false,
      "secureObjectCreatedTs": "2020-08-20T14:15:18.188Z",
      "resourceMetadata": {
        "createdBy": "userId: 161",
        "createdTs": "2020-08-20T16:46:37.314Z",
        "lastModifiedBy": "userId: 161",
        "lastModifiedTs": "2020-08-20T16:46:37.314Z",
        "version": 0
      }
    }
  ]
}
```


4.5.7 Retrieve secure objects that can be assigned to a device group

```
GET /rtp/device-groups/{deviceGroupId}/available-secure-objects
```

This API endpoint allows you to obtain the list of secure objects that can be assigned to a given device group identified by a unique device group ID. Each secure object in the

list is uniquely identified by a secure object ID. See Section 4.1.4 to learn how to assign a secure object to a device group.

Note: public keys and certificates values are always provided in PEM format when available in the response.

An example call is reported below:

Request

cURL Command:

```
curl -X GET "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/device-
groups/<deviceGroupId>/available-secure-objects"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
```

JSON request body:

No input parameters

Response


```
[
  {
    "secureObjectType": "BINARY_FILE",
    "id": 5427,
    "name": "MyNewNonConfidentialBinaryFile",
    "objectId": "8300000B",
    "policySourceType": "DEFAULT",
    "generateOnDeviceConnection": false,
    "deviceGroupCount": null,
    "provisioningCount": null,
    "downloadedCount": null,
    "resourceMetadata": {
      "createdBy": "apiKey: RTP",
      "createdTs": "2021-03-16T11:07:03.869Z",
      "lastModifiedBy": "apiKey: RTP",
      "lastModifiedTs": "2021-03-16T11:07:03.877Z",
      "version": 1
    },
    "binaryFileType": "NON_CONFIDENTIAL"
  },
  {
    "secureObjectType": "KEYPAIR",
    "id": 5424,
    "name": "MyNewKeyPair",
    "objectId": "83000007",
    "policySourceType": "DEFAULT",
    "generateOnDeviceConnection": true,
    "deviceGroupCount": null,
    "provisioningCount": null,
    "downloadedCount": null,
    "resourceMetadata": {
      "createdBy": "apiKey: RTP",
      "createdTs": "2021-03-16T09:56:31.665Z",
      "lastModifiedBy": "apiKey: RTP",
      "lastModifiedTs": "2021-03-16T09:56:31.675Z",
      "version": 1
    },
    "algorithm": "NIST_P384"
  }
]
```


4.5.8 Retrieve device groups assigned to a secure object

```
GET /rtp/secure-objects/{secureObjectId}/assigned-device-groups
```

This API endpoint allows you to obtain the list of device groups that have been assigned to a given secure object identified by a unique secure object ID.

Note: You can choose the number of records per page that you want to retrieve and, if the number of elements surpasses the number of records, you can specify the page you want to retrieve. Finally you can sort results in ascending or descending order based on a field. Please refer to Section 7 and the OpenAPI documentation for the query parameters that can be used for this purpose.

An example call is reported below:

Request

cURL Command:

```
curl -X GET "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/device-groups/<deviceGroupId>/available-secure-objects"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
```

JSON request body:

No input parameters

Response

```
{
  "current": "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/secure-objects/702/assigned-device-groups?sortByDirection=DESC&page=0&size=20",
  "next": null,
  "prev": null,
  "totalElements": 2,
  "pageType": "DeviceGroupInSecureObjectPage",
  "content": [
    {
      "deviceGroupId": 474,
      "deviceGroupName": "NewGroup",
      "unassignable": false
    },
    {
      "deviceGroupId": 4122,
      "deviceGroupName": "MyGroup",
      "unassignable": true
    }
  ]
}
```

4.5.9 Retrieve provisionings of a secure object

```
GET /rtp/secure-objects/{secureObjectId}/provisionings
```

This API endpoint allows you to retrieve the provisionings of a secure object identified by a unique secure object ID. A provisioning consists of the association of a secure object, a device group and a device belonging to the device group. EdgeLock 2GO assigns a unique ID to the provisioning.

Note: You can filter provisionings by device ID (`deviceId`) and provisioning state (`provisioningState` - GENERATION_TRIGGERED, GENERATION_COMPLETED, GENERATION_FAILED, PROVISIONING_COMPLETED, PROVISIONING_FAILED, GENERATION_ON_CONNECTION). You can choose the number of records per page that you want to retrieve and, if the number of elements surpasses the number of records, you can specify the page you want to retrieve. Finally you can sort results in ascending or descending order based on a field. Please refer to Section 7 and the OpenAPI documentation for the query parameters that can be used for this purpose.

An example call is reported below:

Request

cURL Command:

```
curl -X GET "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/secure-objects/<secureObjectId>/provisionings"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
```

JSON request body:

```
No input parameters
```

Response


```
{
  "current": "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/secure-objects/3338/provisionings?sortByDirection=DESC&page=0&size=20",
  "next": null,
  "prev": null,
  "totalElements": 1,
  "pageType": "SecureObjectProvisioningPage",
  "content": [
    {
      "provisioningId": 29960,
      "secureObjectId": 3338,
      "secureObjectName": "MyBinaryFile",
      "secureObjectType": "BINARY_FILE",
      "deviceId": "348555491056936142262187597632590342062080",
      "deviceGroupId": 474,
      "deviceGroupName": "NewGroup",
      "provisioningState": "GENERATION_COMPLETED",
      "resourceMetadata": {
        "createdBy": "SYSTEM_USER",
        "createdTs": "2021-03-17T10:20:28.901Z",
        "lastModifiedBy": "SYSTEM_USER",
        "lastModifiedTs": "2021-03-17T10:20:29.168Z",
        "version": null
      }
    }
  ]
}
```


4.5.10 Retrieve provisionings of a device

```
GET /rtp/devices/{deviceId}/secure-object-provisionings
```

This API endpoint allows you to retrieve the secure object provisionings of a device identified by a unique device ID. A provisioning consists of the association of a secure object, a device group and a device belonging to the device group. EdgeLock 2GO assigns a unique ID to each provisioning.

Note: You can choose the number of records per page that you want to retrieve and, if the number of elements surpasses the number of records, you can specify the page you want to retrieve. Finally you can sort results in ascending or descending order based on a field. Please refer to Section 7 and the OpenAPI documentation for the query parameters that can be used for this purpose.

An example call is reported below:

Request

```
cURL Command:
curl -X GET "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/devices/
<deviceId>/secure-object-provisionings"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
```

```
JSON request body:
No input parameters
```

Response


```
{
  "current": "<current_page_URL>",
  "next": null,
  "prev": null,
  "totalElements": 1,
  "pageType": "SecureObjectProvisioningPage",
  "content": [
    {
      "provisioningId": 29960,
      "secureObjectId": 3338,
      "secureObjectName": "MyBinaryFile",
      "secureObjectType": "BINARY_FILE",
      "deviceId": "348555491056936142262187597632590342062080",
      "deviceGroupId": 474,
      "deviceGroupName": "NewGroup",
      "provisioningState": "GENERATION_COMPLETED",
      "resourceMetadata": {
        "createdBy": "SYSTEM_USER",
        "createdTs": "2021-03-17T10:20:28.901Z",
        "lastModifiedBy": "SYSTEM_USER",
        "lastModifiedTs": "2021-03-17T10:20:29.168Z",
        "version": null
      }
    }
  ]
}
```


4.5.11 Download certificate of an X.509 secure object provisioning

```
GET /rtp/provisionings/{provisioningId}/download-certificate
```

This API endpoint allows you to retrieve the X.509 certificate in PEM format associated to an X.509 certificate secure object provisioning. The unique X.509 certificate secure object provisioning ID can be obtained using the API endpoints described in Section 4.5.9 and Section 4.5.10.

An example call is reported below:

Request

```
cURL Command:
curl -X GET "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/
provisionings/<provisioningId>/download-certificate"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
```

```
JSON request body:
No input parameters
```

Response

```
<X.509_certificate_in_PEM_format>
```

4.5.12 Download intermediate certificate of an X.509 secure object provisioning

```
GET /rtp/provisionings/{provisioningId}/download-intermediate-
ca
```

This API endpoint allows you to retrieve the intermediate X.509 certificate in PEM format associated to an X.509 certificate secure object provisioning. The unique X.509 certificate secure object provisioning ID can be obtained using the API endpoints described in Section 4.5.9 and Section 4.5.10.

An example call is reported below:

Request

```
cURL Command:
curl -X GET "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/
provisionings/<provisioningId>/download-intermediate-ca"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
```

```
JSON request body:
No input parameters
```

Response

```
<X.509_intermediate_certificate_in_PEM_format>
```

4.5.13 Delete a secure object

```
DELETE /rtp/secure-objects/{secureObjectId}
```

This API endpoint allows you to delete a secure object identified by a unique secure object ID. The secure object ID is assigned by EdgeLock 2GO at secure object creation as shown in Section 4.1.3.

Note: a secure object cannot be deleted if it has been assigned to a device group. Unassign the secure object from the device group to be able to delete it.

Note: the action of deleting a secure object is not reversible.

An example call is reported below:

Request

```
cURL Command:
curl -X DELETE "https://api.foundries.io/ota/factories/<factory>/el2g-
proxy/iothubapigateway/v1/rtp/secure-objects/<secureObjectId>"
     -H "accept: */*"
     -H "OSF-TOKEN: <token>"
```

```
JSON request body:
No input parameters
```

Response

```
204 No Content
```

4.5.14 Update a secure object

```
PUT /rtp/secure-objects/{secureObjectId}
```

This API endpoint allows you to update some of the data (name and policies) of an existing secure object identified by a unique secure object ID. Refer to Section 4.5.5 to retrieve the ID of all your secure objects.

Note: *policies can only be added to secure objects that have been created with custom usage policies. See Section 9 for more details on how to format policies.*

An example call is reported below:

Request

cURL Command:

```
curl -X PUT "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/secure-objects/<secureObjectId>"
     -H "accept: */*"
     -H "OSF-TOKEN: <token>"
     -H "Content-Type: application/json"
     -d "[escaped JSON request body, see Section 3]"
```

JSON request body:

```
{
  "name": "newSecureObjectName",
  "policies": {
        "1219091321O3": {
            "policyType": "ACCESS_RULE",
            "policySourceType": "CUSTOM",
            "accessRuleConfiguration": [
                {
                    "objectId": "00000000",
                    "type": "UNAUTHENTICATED",
                    "accessRules": [
                        "POLICY_OBJ_ALLOW_DELETE",
                        "POLICY_OBJ_ALLOW_SIGN",
                        "POLICY_OBJ_ALLOW_WRITE",
                        "POLICY_OBJ_ALLOW_GEN",
                        "POLICY_OBJ_ALLOW_KA"
                    ]
                },
                {
                    "objectId": "83000006",
                    "type": "CUSTOM",
                    "accessRules": [
                        "POLICY_OBJ_FORBID_ALL"
                    ]
                },
                {
                    "objectId": "F0000021",
                    "type": "MANDATORY",
                    "accessRules": [
                        "POLICY_OBJ_ALLOW_DELETE"
                    ]
                }
            ]
        }
    }
}
```

Response

```
204 No Content
```

4.5.15 Retrieve policies of a secure object

```
GET /rtp/secure-objects/{secureObjectId}/policies
```

This API endpoint allows you to retrieve the usage policies (default or custom) applied to a secure object identified by a unique secure object ID. For each usage policy, the access rule configuration is provided (mandatory, unauthenticated and custom if any).

Note: *default usage policies are only listed if the secure object has been assigned to a device group.*

Note*: you can filter policies by 12NC (`nc12`). You can also choose the number of records per page that you want to retrieve and, if the number of elements surpasses the number of records, you can specify the page you want to retrieve. Please refer to Section 7 and the OpenAPI documentation for the query parameters that can be used for this purpose.*

An example call is reported below:

Request

cURL Command:

```
curl -X GET "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/secure-objects/<secureObjectId>/policies"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
```

JSON request body:

```
No input parameters
```

Response

```
{
  "current": "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/secure-objects/21322/policies?page=0&size=20",
  "next": null,
  "prev": null,
  "totalElements": 1,
  "pageType": "SecureObjectPolicyPage",
  "content": [
    {
      "commercialName": "SE050C2HQ1/Z01V3",
      "nc12": "935389312472",
      "assignedDeviceGroups": [],
      "secureObjectPolicyConfiguration": {
        "policyType": "ACCESS_RULE",
        "policySourceType": "CUSTOM",
        "accessRuleConfiguration": [
          {
            "objectId": "00000000",
            "type": "UNAUTHENTICATED",
            "accessRules": [
              "POLICY_OBJ_ALLOW_DELETE",
              "POLICY_OBJ_ALLOW_SIGN",
              "POLICY_OBJ_ALLOW_WRITE",
              "POLICY_OBJ_ALLOW_GEN",
              "POLICY_OBJ_ALLOW_KA"
            ]
          },
          {
            "objectId": "F0000021",
            "type": "MANDATORY",
            "accessRules": [
              "POLICY_OBJ_ALLOW_DELETE"
            ]
          }
        ]
      }
    }
  ]
}
```

4.5.16 Retrieve default usage policy for secure object

```
GET /rtp/settings/secure-object/policy-template/{nc12}/
{secureObjectType}
```

This API endpoint allows you to retrieve the policies that are applied by default to a specific product and secure object type. OID configurations for policy access rules are also returned. The product 12NC and secure object type (*MASTER_KEY, HMAC_KEY, STATIC_PUBLIC_KEY, KEYPAIR, CERTIFICATE, BINARY_FILE*) must be specified in the request path.

In some cases, access rules and policies might change based on the algorithm type set in the secure object (e.g. for key pair secure objects, static public key secure objects and X.509 certificates). In this case, it is also mandatory to set the algorithm type as query parameter (`?algorithm=<algorithm>`).

Supported algorithm values are:

- NIST_P192, NIST_P224, NIST_P256, NIST_P384, NIST_P521
- SECP160K1, SECP192K1, SECP224K1, SECP256K1
- BRAINPOOLP160R1, BRAINPOOLP192R1, BRAINPOOLP224R1, BRAINPOOLP256R1, BRAINPOOLP320R1, BRAINPOOLP384R1, BRAINPOOLP512R1
- ECC_ED_25519, ECC_MONT_DH_448, ECC_MONT_DH_25519
- RSA_1024, RSA_2048, RSA_3072, RSA_4096, RSA_1024_CRT, RSA_2048_CRT, RSA_3072_CRT, RSA_4096_CRT
- AES128, AES256
- BINARY_FILE_CONFIDENTIAL, BINARY_FILE_NON_CONFIDENTIAL

The *policyType* query parameter is always mandatory and should be set to `ACCESS_RULE` (future releases of EdgeLock 2GO Managed will support other options).

An example call is reported below for a NIST-P256 keypair:

Request

cURL Command:

```
curl -X GET "https://api.foundries.io/ota/factories/<factory>/el2g-
proxy/rtp/settings/secure-object/policy-template/935389312472/KEYPAIR?
algorithm=NIST_P256&policyType=ACCESS_RULE"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
```

JSON request body:

```
No input parameters
```

Response


```
{
  "secureObjectPolicyTemplate": {
    "secureObject": {
      "keyPairAlgorithmType": "NIST_P256",
      "type": "KEYPAIR"
    },
    "enabled": true,
    "policyType": "ACCESS_RULE",
    "accessRuleConfigurationTemplate": [
      {
        "objectId": "F0000021",
        "type": "MANDATORY",
        "accessRules": [
          "POLICY_OBJ_ALLOW_DELETE"
        ]
      },
      {
        "objectId": "00000000",
        "type": "UNAUTHENTICATED",
        "accessRules": [
          "POLICY_OBJ_ALLOW_DELETE",
          "POLICY_OBJ_ALLOW_SIGN",
          "POLICY_OBJ_ALLOW_WRITE",
          "POLICY_OBJ_ALLOW_GEN",
          "POLICY_OBJ_ALLOW_KA"
        ]
      }
    ],
    "allowedAccessRuleOptions": [
      "POLICY_OBJ_ALLOW_DELETE",
      "POLICY_OBJ_REQUIRE_SM",
      "POLICY_OBJ_FORBID_ALL",
      "POLICY_OBJ_ALLOW_SIGN",
      "POLICY_OBJ_ALLOW_VERIFY",
      "POLICY_OBJ_ALLOW_WRITE",
      "POLICY_OBJ_ALLOW_GEN",
      "POLICY_OBJ_ALLOW_KA",
      "POLICY_OBJ_ALLOW_READ",
      "POLICY_OBJ_ALLOW_ATTESTATION"
    ]
  },
  "accessRuleOidConfiguration": {
    "mandatoryUserOid": "F0000021",
    "unauthenticatedUserOid": "00000000",
    "objectIdRanges": {
      "SUGGESTED": [
        {
          "minValue": "83000000",
          "maxValue": "8300FFFF"
        }
      ],
      "FORBIDDEN": [
        {
          "minValue": "00000000",
          "maxValue": "00000000"
        },
        {
          "minValue": "7FFF0200",
          "maxValue": "7FFF0202"
        },
        {
          "minValue": "7FFF0204",
          "maxValue": "7FFF0204"
        },
        {
          "minValue": "F0000000",
          "maxValue": "FFFFFFFF"
        }
      ]
    }
  }
}
```


4.5.17 Retrieve OID validation rules

```
GET /rtp/settings/secure-object/SE05/reserved-oids
```

When creating a secure object, an OID must be assigned to the object. This API endpoint allows you to retrieve the validation rules that EdgeLock 2GO uses to validate if the OID provided is valid. The API call returns suggested OID ranges and forbidden OID ranges for EdgeLock SE05x devices.

An example call is reported below:

Request

```
cURL Command:
curl -X GET "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/settings/
secure-object/SE05/reserved-oids"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
```

```
JSON request body:
No input parameters
```

Response

```
{
  "suggestedObjectIdRanges": [
    {
      "minValue": "83000000",
      "maxValue": "8300FFFF"
    }
  ],
  "forbiddenObjectIdRanges": [
    {
      "minValue": "00000000",
      "maxValue": "00000000"
    },
    {
      "minValue": "7FFF0200",
      "maxValue": "7FFF0202"
    },
    {
      "minValue": "7FFF0204",
      "maxValue": "7FFF020B"
    },
    {
      "minValue": "F0000000",
      "maxValue": "FFFFFFFF"
    }
  ]
}
```

4.6 Additional API endpoints for intermediate certificates management

This section lists and describes the remaining EdgeLock 2GO API endpoints for management of intermediate certificates that are used to sign X.509 certificate secure objects.

Note: Make sure that the API key you are using has all the permissions required to execute the API calls listed in this section.

4.6.1 Create intermediate certificate signed by NXP root certificate

```
POST /rtp/intermediate-cas
```

This API endpoint allows you to generate a new X.509 intermediate certificate in EdgeLock 2GO signed by either the NXP root certificate or a custom root CA. This section will only cover the generation of an intermediate certificate signed by the NXP root CA. For instructions related to the generation of a certificate signed by a custom root certificate, please refer to Section 4.2. To generate a certificate signed by NXP

root CA, the *WITHOUT_CSR* parameter must be set in the JSON request body. The JSON request body must also contain the name of the certificate and the cryptographic algorithm used to create the certificate (either NIST_P256, NIST_P384,NIST_P521, RSA_2048, RSA_4096). If the generation is successful, the intermediate certificate can be used to sign X.509 certificate secure objects. Refer to Section 4.1.3 for more details on the creation of X.509 certificate secure objects.

Note: the generated intermediate certificate is only available in PEM format.

An example call is reported below:

Request

cURL Command:

```
curl -X POST "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/
intermediate-cas"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
     -H "Content-Type: application/json"
     -d "[escaped JSON request body, see Section 3]"
```

JSON request body:

```
{
  "algorithm": "NIST_P256",
  "name": "TestCertificate",
  "requestType": "WITHOUT_CSR"
}
```

Response

```
{
  "id": 51,
  "algorithm": "NIST_P256",
  "name": "TestCertificate",
  "value": "<X.509 INTERMEDIATE CERTIFICATE in PEM format>",
  "commonName": "testcertificate-0000000000000033-00000000-vE200",
  "organization": "NXP",
  "orgUnit": "EdgeLock2GoRemoteTpPlugAndTrustCA",
  "country": "AT",
  "signingState": "SIGNED",
  "signingAuthority": "NXP",
  "validUntil": "2021-03-11T15:54:27.360Z",
  "hasAuthorityKeyIdentifier": false,
  "hasSubjectKeyIdentifier": false,
  "resourceMetadata": {
     "createdBy": "userId: 339",
     "createdTs": "2021-03-11T15:54:27.360Z",
     "lastModifiedBy": "userId: 339",
     "lastModifiedTs": "2021-03-11T15:54:27.360Z",
     "version": 1
  }
}
```

4.6.2 Retrieve intermediate certificate by ID

```
GET /rtp/intermediate-cas/{intermediateCaId}
```

This API endpoint allows you to retrieve the data of an intermediate certificate for the RTP service identified by a unique certificate ID. The certificate ID is assigned by EdgeLock 2GO at certificate creation as shown in Section 4.6.1 and Section 4.2.

An example call is reported below:

Request

```
cURL Command:
curl -X GET "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/
intermediate-cas/<intermediateCaId>"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
```

```
JSON request body:
No input parameters
```

Response

```
{
  "id": 51,
  "algorithm": "NIST_P256",
  "name": "TestCertificate",
  "value": "<X.509 INTERMEDIATE CERTIFICATE in PEM format>",
  "commonName": "testcertificate-0000000000000033-00000000-vE200",
  "organization": "NXP",
  "orgUnit": "EdgeLock2GoRemoteTpPlugAndTrustCA",
  "country": "AT",
  "signingState": "SIGNED",
  "signingAuthority": "NXP",
  "validUntil": "2021-03-11T15:54:27.360Z",
  "hasAuthorityKeyIdentifier": false,
  "hasSubjectKeyIdentifier": false,
  "resourceMetadata": {
     "createdBy": "userId: 339",
     "createdTs": "2021-03-11T15:54:27.360Z",
     "lastModifiedBy": "userId: 339",
     "lastModifiedTs": "2021-03-11T15:54:27.360Z",
     "version": 1
  }
}
```

4.6.3 Retrieve all intermediate certificates

```
GET /rtp/intermediate-cas
```

This API endpoint allows you to obtain a list of all the intermediate certificates for the remote trust provisioning service stored in EdgeLock 2GO. Each certificate is uniquely identified by a certificate ID. For each certificate, you can also find some additional information regarding its state, such as if the certificate has been signed and by whom (NXP or an external certificate authority) and if the certificate is valid or expired.

Note: the intermediate certificates are in PEM format.

Note: You can filter certificates by name (`name`), status (`status` - CSR_CREATED, SIGNED), signer (`signer` - NXP, EXT), validity (`validity` - VALID, EXPIRED) and algorithm (`algorithm` - NIST_P256, NIST_P384, NIST_P521, RSA_2048, RSA_4096). You can also choose the number of records per page that you want to retrieve and, if the number of elements surpasses the number of records, you can specify the page you want to retrieve. Finally you can sort results in ascending or descending order based on a field. Please refer to Section 7 and the OpenAPI documentation for the query parameters that can be used for these purposes.

An example call is reported below:

Request

cURL Command:

```
curl -X GET "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/
intermediate-cas"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
```

JSON request body:

No input parameters

Response


```
{
  "current": "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/
intermediate-cas?sort-by-direction=DESC&page=0&size=20",
  "next": null,
  "prev": null,
  "totalElements": 2,
  "pageType": "RtpIntermediateCertificatePage",
  "content": [
    {
      "id": 474,
      "algorithm": "NIST_P256",
      "name": "MySigningCertificate",
      "value": "<X.509 certificate PEM value>",
      "commonName": "MySigningCertifi-00000000000001da-00000000-vE200",
      "organization": "NXP",
      "orgUnit": "EdgeLock2GoRemoteTpPlugAndTrustCA",
      "country": null,
      "locality": null,
      "state": null,
      "signingState": "SIGNED",
      "signingAuthority": "NXP",
      "validUntil": "2035-08-20T14:30:33Z",
      "hasAuthorityKeyIdentifier": false,
      "hasSubjectKeyIdentifier": false,
      "resourceMetadata": {
        "createdBy": "userId: 161",
        "createdTs": "2020-08-20T14:30:33.227Z",
        "lastModifiedBy": "userId: 161",
        "lastModifiedTs": "2020-08-20T14:30:33.471Z",
        "version": 1
      }
    },
    {
      "id": 397,
      "algorithm": "NIST_P256",
      "name": "testCsrCert",
      "value": null,
      "commonName": "testCsrCert",
      "organization": null,
      "orgUnit": null,
      "country": "AT",
      "locality": null,
      "state": null,
      "signingState": "CSR_CREATED",
      "signingAuthority": "EXT",
      "validUntil": null,
      "hasAuthorityKeyIdentifier": false,
      "hasSubjectKeyIdentifier": false,
      "resourceMetadata": {
        "createdBy": "userId: 161",
        "createdTs": "2020-08-18T16:14:36.445Z",
        "lastModifiedBy": "userId: 161",
        "lastModifiedTs": "2020-08-18T16:14:37.93Z",
        "version": 1
      }
    }
  ]
}
```

4.6.4 Download intermediate certificate

```
GET /rtp/intermediate-cas/{intermediateCaId}/download-
intermediate-ca
```

This API endpoint allows you to retrieve the X.509 intermediate certificate in PEM format associated to an intermediate CA for the RTP service. To retrieve the certificate, you must provide the unique intermediate CA ID that you obtained when you created the intermediate CA as described in Section 4.6.1 and Section 4.2.

An example call is reported below:

Request

cURL Command:

```
curl -X GET "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/
intermediate-cas/<intermediateCaId>/download-intermediate-ca"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
```

JSON request body:

```
No input parameters
```

Response

```
<X.509_certificate_in_PEM_format>
```

4.6.5 Create a verification certificate signed by the intermediate CA

```
POST /rtp/intermediate-cas/{intermediateCaId}/create-
verification-certificate
```

This API endpoint allows you to create an X.509 verification certificate in PEM format signed by an intermediate CA for the RTP service. To retrieve the verification certificate you must provide the unique intermediate CA ID that you obtained when you created the intermediate CA as described in Section 4.6.1 and Section 4.2. The body of the request must contain the verification code as provided by your cloud service provider (e.g. Azure or AWS). The verification code will be included in the Common Name (CN) of the verification certificate.

An example call is reported below:

Request

cURL Command:

```
curl -X POST "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/
intermediate-cas/<intermediateCaId>/create-verification-certificate"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
     -H "Content-Type: application/json"
     -d "[escaped JSON request body, see Section 3]"
```

JSON request body:

```
{
  "verificationCode": "<your_verification_code>"
}
```

Response

```
<X.509_certificate_in_PEM_format>
```

4.6.6 Retrieve secure objects assigned to an intermediate certificate

```
GET /rtp/intermediate-cas/{intermediateCaId}/secure-objects
```

This API endpoint allows you to retrieve the secure objects assigned to the intermediate certificate identified by the unique ID specified in the URL path parameters.

Note: You can choose the number of records per page that you want to retrieve and, if the number of elements surpasses the number of records, you can specify the page you want to retrieve. Please refer to Section 7 and the OpenAPI documentation for the query parameters that can be used for this purpose.

An example call is reported below:

Request

cURL Command:

```
curl -X GET "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/
intermediate-cas/<intermediateCaId>/secure-objects"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
```

JSON request body:

```
No input parameters
```

Response

```
{
  "current": "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/
intermediate-cas/474/secure-objects?page=0&size=20",
  "next": null,
  "prev": null,
  "totalElements": 2,
  "pageType": "IntermediateCaCertificatesPage",
  "content": [
    {
      "secureObjectType": "CERTIFICATE",
      "name": "MyCertificate1",
      "assignmentDateTime": "2021-03-16T10:19:13.279Z"
    },
    {
      "secureObjectType": "CERTIFICATE",
      "name": "MyCertificate2",
      "assignmentDateTime": "2020-08-20T14:50:02.605Z"
    }
  ]
}
```

4.6.7 Delete an intermediate certificate

```
DELETE /rtp/intermediate-cas/{intermediateCaId}
```

This API endpoint allows you to delete an intermediate certificate (for signing X.509 certificate secure objects) identified by a unique certificate ID. The certificate ID is assigned by EdgeLock 2GO at certificate creation as shown in Section 4.6.1 and Section 4.2.

Note: an intermediate certificate cannot be deleted if it has been used to sign an X.509 certificate secure object. Before deleting the intermediate certificate, remove the secure objects assigned as described in Section 4.5.13.

Note: the action of deleting a certificate is not reversible.

An example call is reported below:

Request

```
cURL Command:
curl -X DELETE "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/
intermediate-cas/<intermediateCaId>"
     -H "accept: */*"
     -H "OSF-TOKEN: <token>"
```

```
JSON request body:
No input parameters
```

Response

```
204 No Content
```

4.6.8 Retrieve supported algorithms

```
GET /rtp/intermediate-cas/supported-algorithms
```

When creating an X.509 certificate secure object, it is required to assign to the secure object both a key pair secure object and an intermediate CA to sign the certificate object. However, not all combinations of key pair and intermediate CA algorithms are supported. This API endpoint allows you to retrieve the supported combinations of key pair and intermediate CA algorithms. For each supported combination, the `status` field will indicate if there is a security issue with a particular combination (e.g. the intermediate CA is using a key strength weaker than the key pair). In that case, the `status` field is set to `WARNING`.

An example call is reported below:

Request

```
cURL Command:
curl -X GET "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/
intermediate-cas/supported-algorithms"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
```

```
JSON request body:
No input parameters
```

Response


```
{
  "supportedAlgorithms": [
    {
      "status": "NONE",
      "keyPairAlgorithm": "NIST_P192",
      "intermediateCaAlgorithm": "NIST_P256"
    },
    {
      "status": "NONE",
      "keyPairAlgorithm": "NIST_P192",
      "intermediateCaAlgorithm": "NIST_P521"
    },
    {
      "status": "NONE",
      "keyPairAlgorithm": "NIST_P224",
      "intermediateCaAlgorithm": "NIST_P256"
    },
    {
      "status": "NONE",
      "keyPairAlgorithm": "NIST_P224",
      "intermediateCaAlgorithm": "NIST_P521"
    },
    {
      "status": "NONE",
      "keyPairAlgorithm": "NIST_P256",
      "intermediateCaAlgorithm": "NIST_P256"
    },
    {
      "status": "NONE",
      "keyPairAlgorithm": "NIST_P256",
      "intermediateCaAlgorithm": "NIST_P521"
    },
    {
      "status": "WARNING",
      "keyPairAlgorithm": "NIST_P384",
      "intermediateCaAlgorithm": "NIST_P256"
    },
    {
      "status": "NONE",
      "keyPairAlgorithm": "NIST_P384",
      "intermediateCaAlgorithm": "NIST_P521"
    },
    {
      "status": "WARNING",
      "keyPairAlgorithm": "NIST_P521",
      "intermediateCaAlgorithm": "NIST_P256"
    },
    … Note: some supported algorithm combinations have been removed for conciseness
  ]
}
```


4.7 Additional API endpoints for activity management

This section lists and describes the remaining EdgeLock 2GO API endpoints for management of user activity.

Note: Make sure that the API key you are using has all the permissions required to execute the API calls listed in this section.

4.7.1 Generate an activity report

```
POST /audits/create-report
```

This API endpoint can be used to generate a report of all the activities and events that a user performed in a given timeframe. The JSON request body must be filled with the time interval for which you want to generate the report. If all the provided parameters are correct, a CSV file with the generated report will be returned. Each line of the CSV

contains the company name, ID, event type and number of occurrences of each event for all months included in the specified timeframe.

An example call is reported below:

Request

```
cURL Command:
curl -X POST "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/audits/
create-report"
 -H "accept: application/json"
 -H "OSF-TOKEN: <token>"
 -H "Content-Type: application/json"
 -d "[escaped JSON request body]"
```

```
JSON request body:
{
    "startMonthYear": {
        "month": 1,
        "year": 2022
    },
    "endMonthYear": {
        "month": 1,
        "year": 2022
    }
}
```

Response

```
Company Name,Company ID,Event Type,Jan-22
Coffee Shop,19,NEW_DEVICE,0
Coffee Shop,19,DEVICE_CLAIMED,16
Coffee Shop,19,DEVICE_UNCLAIMED,16
Coffee Shop,19,PROVISION_SERVICE,0
Coffee Shop,19,REVOKE_SERVICE_DEVICE,3
Coffee Shop,19,NEW_COS_INTERMEDIATE_CA_NXP,3
Coffee Shop,19,NEW_COS_INTERMEDIATE_CA_CSR,0
Coffee Shop,19,COS_INTERMEDIATE_CA_CSR_SIGNED,0
Coffee Shop,19,DELETE_COS_INTERMEDIATE_CA,0
Coffee Shop,19,RTP_INTERMEDIATE_CA_NXP_CREATED,6
Coffee Shop,19,RTP_INTERMEDIATE_CA_CSR_CREATED,0
```

5 Annex: HTTP status codes and errors

HTTP status codes and errors

Status code	Description
200 OK	Request is successful and a value is typically returned, usually as a JSON object
202 Accepted	The request has been accepted for processing, but processing has not been completed.
204 No Content	The request has been successful, but there is no content to return.
400 Bad Request	Request cannot be processed because of invalid parameters.
401 Unauthorized	The request could not be executed because it lacked proper authentication credentials.
403 Forbidden	The request could not be executed because the user is not authorized to perform the action.
404 Not Found	The requested resource could not be found.
405 Method Not Allowed	The targeted resource does not support the HTTP method of the request.
409 Conflict	There is a conflict in the targeted resource; e.g. the resource has already been updated by another user.
413 Payload Too Large	The request payload is too big and cannot be accepted by the server.
415 Unsupported Media Type	The request payload is in a format not supported by the server.
422 Unprocessable Entity	Format and syntax of the request are correct, but the contained instructions could not be processed.
500 Internal Server Error	Internal error occurred in the server.
502 Bad Gateway	The server, while acting as a gateway or proxy, received an invalid response from the upstream server.
503 Service Unavailable	The server is currently unable to handle the request due to a temporary overload or scheduled maintenance.

6 Annex: Intermediate certificate requirements

External Issuer CA profile requirements

Field	Restriction on value
Version	Allow only V3
Serial Number	Mandatory
Signature Algorithm	Allow only: SHA256withRSA SHA384withRSA SHA256withECDSA SHA384withECDSA
Signature Hash Algorithm	Allow only: SHA256 SHA384
Issuer	Must be the same as Issuer Subject (this is default)
Valid from	Must be valid in the moment of upload
Valid to	Must not be valid longer than CA Issuer
Subject	Must contain DN fields from CSR
Public key	Must contain the same key as in CSR
Public key param	Allow only: ECDSA_P256 (secp256r1)
Basic Constraint	Subject CA must be CA Path Length Constrain must be 0 Must be marked as critical
Key Usage	Certificate Signing must be set. Must be marked as critical.

7 Annex: Pagination and filtering

Some EdgeLock 2GO API endpoints return a collection of items. EdgeLock 2GO provides a way of manipulating and filtering such collections using a set of query parameters that can be appended to the request. Collections will always be returned as pages which means that if the number of objects is greater than a fixed page size, e.g. 50, then only 50 objects will be returned with a link to the next page. The supported query parameters are:

- **page**: specifies the page that you want to retrieve.
- **size**: specifies the maximum number of objects per page.

The example call reported below limits the number of objects per page to 5 and retrieves the first page (pages are 0-indexed):

```
GET /rtp/secure-objects?size=5&page=0
```

Note: *the maximum number of items per page that can be retrieved using the size parameter is 100.*

The JSON response will contain a link to the current, previous and next page as shown below:

```
{
  "current": "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/
secure-objects?page=0&size=5",
  "prev": null,
  "next": "https://api.foundries.io/ota/factories/<factory>/el2g-
proxy/iothubapigateway/v1/rtp/secure-objects?page=1&size=5",
  "totalElements": 6,
  ...
}
```

Note: *the 'prev' parameter is always null in the first page (page 0). The 'next' parameter is null if there are no further pages to retrieve.*

Depending on the API endpoint, it might be possible to filter the results based on some parameters, sort the results according to a field (`sortBy` query parameter) and order the results in ascending or descending order (`sortByDirection` query parameter) as shown in the example below:

```
GET /rtp/secure-objects?
secureObjectType=MASTER_KEY&sortBy=name&sortByDirection=DESC&page=0&size=20
```

Please refer to the OpenAPI documentation for the list of query parameters available for each API function.

8 Annex: obtain the UID of the device

The device UID can be obtained using the *ssscli* tool by issuing the following commands:

```
sudo ssscli connect se05x t1oi2c none
sudo ssscli se05x uid
```

The output of the last command contains the device UID in HEX format and should be similar to the one shown below:

```
user@imx8mm:~$ sudo ssscli se05x uid
INFO:sss.se05x:04005001d706ca37a1e4ed047e6dda0f6880
Unique ID: 04005001d706ca37a1e4ed047e6dda0f6880
```

9 Annex: Configure secure object policies

When creating a secure object, it is possible to configure the policies that will be applied to the object when it is provisioned in EdgeLock SE05x. Policies are used to allow or restrict some operations that can be performed with the object. For example you can prevent an ECC key from being overwritten or you can restrict its usage to just signing operations. Policy rules in EdgeLock SE05x are associated to authentication objects. Different policy access rules can be configured for each authentication object, so you can fine-tune permissions for different users of the secure element.

Note: *please note that authentication objects cannot be created and managed through EdgeLock 2GO Managed and they must be handled separately.*

Not all policy access rules can be applied to all products supported by EdgeLock 2GO Managed. For this reason, different policy access rules must be defined for each supported product (i.e. for each 12NC). EdgeLock 2GO Managed provides users with two ways of handling this:

- **Default usage policy**: for each product supported by the platform, EdgeLock 2GO Managed defines a default policy to apply to different types of secure objects. The default policy is automatically selected by EdgeLock 2GO Managed when a secure object is assigned to a device group. The policy is selected based on the product type (12NC) of the devices that belongs to the device group.
- **Custom usage policy**: if you want to have a more granular control of the policies that will be applied to secure objects, EdgeLock 2GO Managed allows you to define for each secure object up to 5 custom usage policies, each one tied to a specific product. When the secure object is assigned to a device group, the correct custom usage policy will be applied based on the product type of devices that belongs to the device group. For each custom usage policy you can define:
  - **Unauthenticated access policy:** defines which operations are permitted when no authentication is performed to EdgeLock SE05x. EdgeLock 2GO Managed provides a suggested configuration that can be fine-tuned according to your needs. Optionally, it is possible to disable all policy rules so that unauthenticated access will not be allowed.
  - **Custom policies (for authentication objects):** you can add up to 5 additional custom policies, each one associated to a different authentication object. Please note that creation of authentication objects in EdgeLock SE05x must be handled separately and is outside the scope of EdgeLock 2GO Managed.
  - **Mandatory policy:** this policy is used by the EdgeLock 2GO system for technical reasons and it is applied automatically to all objects. It is not possible to change or disable this policy.

The EdgeLock 2GO API can be used to create secure objects and to assign usage policies and policy access rules (as described in Section 4.1.3).

- To create a secure object with default usage policy, simply set the `policySourceType` property to `DEFAULT` as shown in the example below:

Request

```
cURL Command:
curl -X POST "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/secure-
objects"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
     -H "Content-Type: application/json"
     -d "[escaped JSON request body, see Section 3]"
```

```
JSON request body:
{
  "name": "MyNewKeyPair",
  "algorithm": "NIST_P384",
  "secureObjectType": "KEYPAIR",
  "objectId": "83000007",
  "generateOnDeviceConnection": false,
  "policySourceType": "DEFAULT",
  "policies": null
}
```

- To create a secure object with custom usage policies, you must first set the `policySourceType` property to `CUSTOM` and then provide a list of 12NC for which you want to define a policy (inside the `policies` object).
  Each 12NC entry must then contain `accessRuleConfiguration`: an array that defines the access rules for authentication objects. You must at least add UNAUTHENTICATED and MANDATORY access rules configurations. CUSTOM access rules are optional. See the example below:

Request

```
cURL Command:
curl -X POST "https://api.foundries.io/ota/factories/<factory>/el2g-proxy/rtp/secure-
objects"
     -H "accept: application/json"
     -H "OSF-TOKEN: <token>"
     -H "Content-Type: application/json"
     -d "[escaped JSON request body, see Section 3]"
```

```
JSON request body:
{
  "algorithm": "NIST_P256",
  "policySourceType": "CUSTOM",
  "policies": {
    "935389312472": {
      "policyType": "ACCESS_RULE",
      "policySourceType": "CUSTOM",
      "accessRuleConfiguration": [
        {
          "objectId": "00000000",
          "type": "UNAUTHENTICATED",
          "accessRules": [
            "POLICY_OBJ_ALLOW_DELETE",
            "POLICY_OBJ_ALLOW_SIGN",
            "POLICY_OBJ_ALLOW_WRITE",
            "POLICY_OBJ_ALLOW_GEN",
            "POLICY_OBJ_ALLOW_KA"
          ]
        },
        {
          "objectId": "00000001",
          "type": "CUSTOM",
          "accessRules": [
            "POLICY_OBJ_ALLOW_VERIFY",
            "POLICY_OBJ_FORBID_ALL"
          ]
        },
        {
          "objectId": "F0000021",
          "type": "MANDATORY",
          "accessRules": [
            "POLICY_OBJ_ALLOW_DELETE"
          ]
        }
      ]
    }
  },
  "name": "MyKey",
  "secureObjectType": "KEYPAIR",
  "objectId": "8300000D",
  "generateOnDeviceConnection": true
}
```

10 Annex: Claim codes

EdgeLock 2GO Managed requires devices to be registered to the platform. Devices can be added using their *UIDs* or using *claim codes*. A claim code is a random base64-encoded string of 16 to 255 characters generated for a particular device group. Devices presenting this claim code will be automatically added and assigned to the correct device group. The process is the following:

1. The customer creates a *device group* within its EdgeLock 2GO account;
2. The customer generates a *claim code* for this device group;
3. The customer loads their claim code into their devices before deploying them;
4. The device attempting to connect to EdgeLock 2GO Managed *presents* its identity together with the pre-loaded claim code.
5. EdgeLock 2GO Managed *validates* the claim code and *assigns* the device to the corresponding device group.

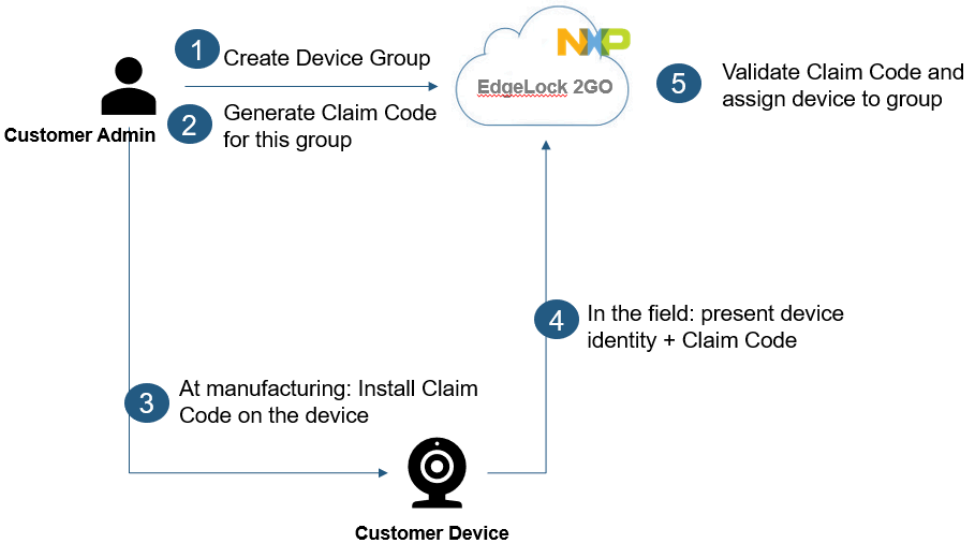

Figure 4. Add devices using claim codes

A claim code within a device group can be created, deleted or revoked at any time, not only during device group creation. Please note that when a claim code has been revoked or deleted, any device presenting that claim code will not be added in EdgeLock 2GO Managed. When you delete a claim code, instead of simply revoking it, the claim code is completely removed from the EdgeLock 2GO Managed database so you can keep under control business quotas and the number of claim codes in the system.

In addition:

- The number of device activations possible per claim code can be limited. When this limit is reached, we can configure that an alert is received or that no more devices are assigned to the device group.
- The number of times a claim code can be used per device can be limited. When the device uses the claim code the first time, it will not be possible for the device to use it another time.

11 Annex: Key and certificate formatting

Some EdgeLock 2GO API functions require to specify in the JSON body of the request the value of a key, certificate or PGP encrypted file in PEM format. When adding the content of a PEM file to the JSON body of an API function, the following requirements must be met:

- The content of the PEM file must be adapted to fit in one single line, i.e. newline characters must be removed;
- Removed newline characters must be substituted with `\n`.

For instance, consider the static public key in PEM format shown in Figure 5:

***Note**: the key used in the example has been shortened for illustrative purposes.*

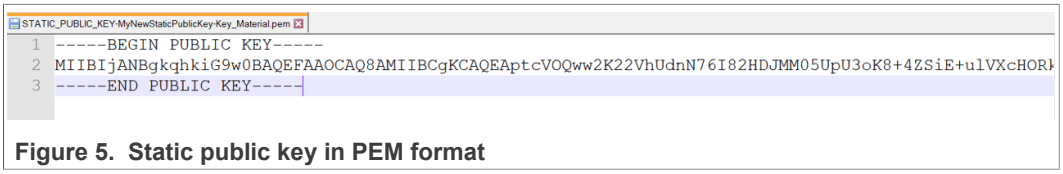

Figure 5. Static public key in PEM format

First, the content of the PEM file shall be adapted to fit in a single line as shown in Figure 6. Then newline characters must be added as shown in Figure 7.

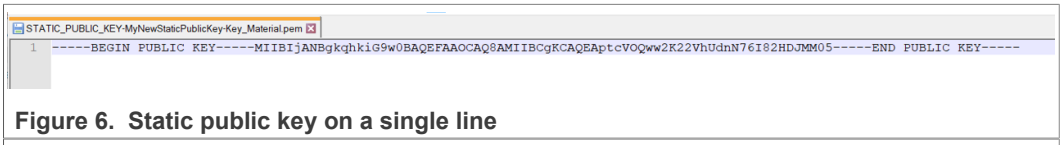

Figure 6. Static public key on a single line

Figure 7. Adding newline characters to static public key

You can now use the key value in the body of the API request as shown below:

```
{
  "name": "MyNewStaticPublicKey",
  "secureObjectType":"STATIC_PUBLIC_KEY",
  "secureObjectId":"8300000D",
  "publicKey": "-----BEGIN PUBLIC KEY-----\nMIIBIjANBgkqh[omitted]I82HDJMM05\n-----END PUBLIC KEY-----",
  "algorithm":"RSA_2048",
  "policies":[
      {
         "objectId":"00000000",
         "accessRules":[
            "POLICY_OBJ_ALLOW_WRITE",
            "POLICY_OBJ_ALLOW_VERIFY",
            "POLICY_OBJ_ALLOW_READ",
            "POLICY_OBJ_ALLOW_DELETE"
         ]
      }
   ]
}
```

12 Encrypt and sign AES keys, HMAC keys and confidential binary files using PGP

Before importing AES key material, HMAC key material or confidential binary files in EdgeLock 2GO Managed, they must be signed and encrypted using PGP. To do this, you can use any PGP software available for your operating system. The AES key, HMAC key or confidential binary file needs to be encrypted using the public key in the EdgeLock 2GO keyring and signed using any PGP key-pair that belongs to you. The PGP public key associated to the key-pair will be used by EdgeLock 2GO Managed to verify the validity of the signature.

EdgeLock 2GO supports the following algorithms while processing PGP messages:

- The customer must use an RSA 2048-bit key or an RSA 4096-bit key to sign the key material;
- NXP provides an RSA 4096-bit key to encrypt the key material;
- Public Key Algorithm: RSA for both encryption and signature.
- Symmetric Key Algorithms: AES-256, AES-192, AES-128
- Compression Algorithms: Uncompressed, ZLIB
- Hash Algorithms: SHA-256

Please make sure that your software is using the abovementioned algorithms.

In this section we will demonstrate how to encrypt and sign AES keys, HMAC keys and confidential binary files using the Kleopatra software for Windows. This is part of the Gpg4win package that can be downloaded at https://www.gpg4win.org/get-gpg4win.html. Follow these instructions:

1. Open the Kleopatra software and create a new key pair (if don't have one already) as shown in Figure 8:
   (1) Click on *File → New Key Pair*;
   (2) In the popup window that appears select the option *Create Personal OpenPGP Key pair*.

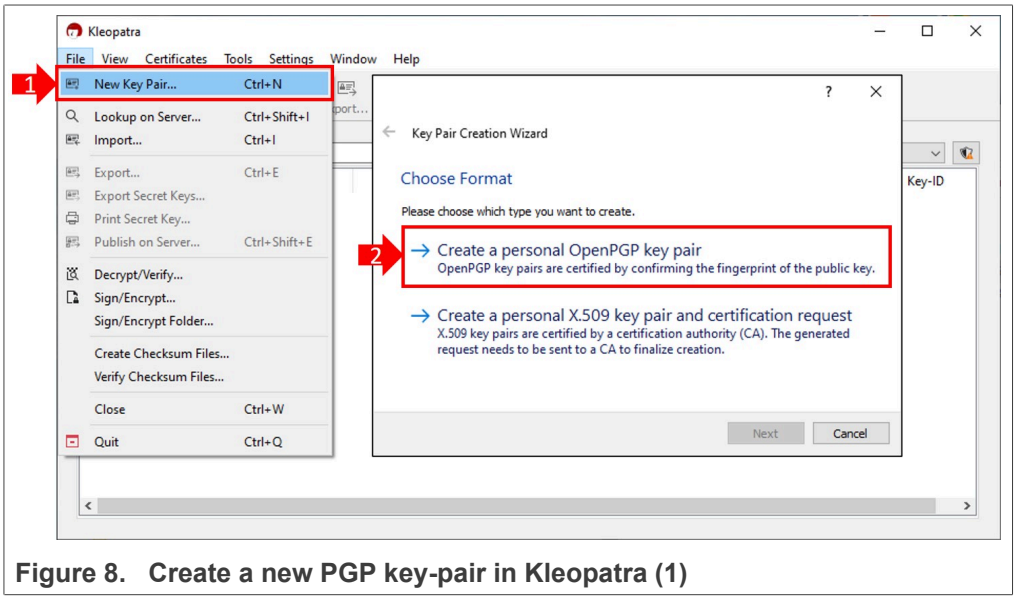

Figure 8. Create a new PGP key-pair in Kleopatra (1)

2. Select the properties of the new key pair as shown in Figure 9:

(1) Provide a name and an email address to associate to the key pair;
(2) Click on the *Advanced Settings* button to configure the key;
(3) In the popup window that appears make sure to select RSA. You can select the size of the key that you prefer;
(4) Confirm the settings;
(5) Click *Next* to proceed with the creation wizard.

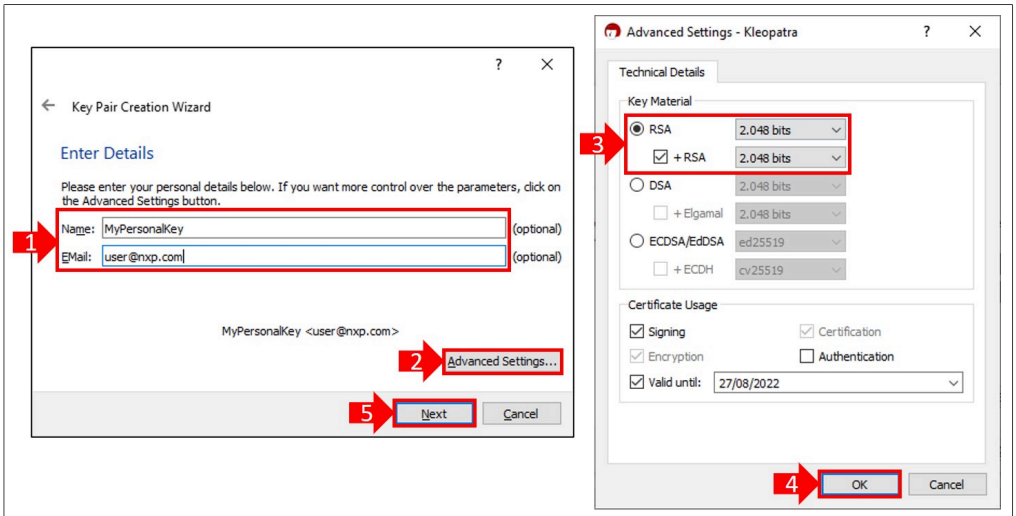

Figure 9. Create a new PGP key-pair in Kleopatra (2)

3. Confirm the settings you provided and provide a password for the key pair as shown in Figure 10:
(1) Check that the data is correct and then click the *Create* button;
(2) In the next window, provide a password for the key pair and confirm the choice.

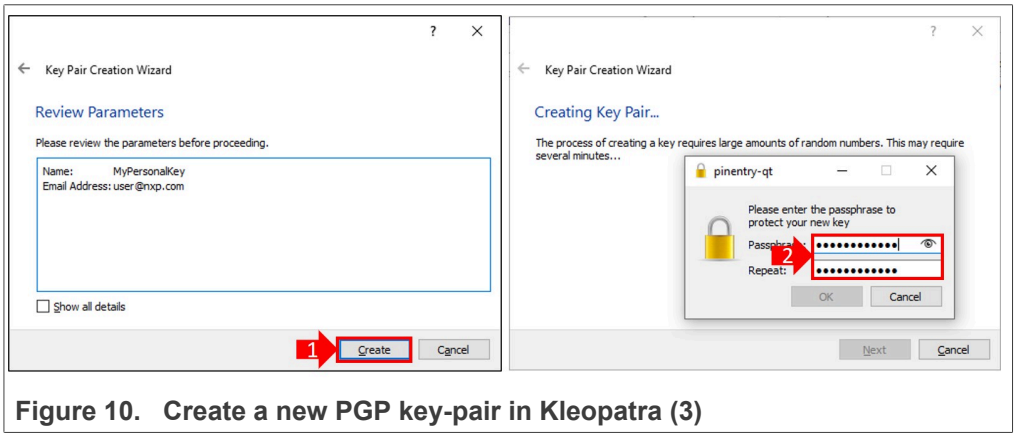

Figure 10. Create a new PGP key-pair in Kleopatra (3)

4. Finally (1) click on the *Finish* button to complete the key creation process. (2) You should be able to see the key pair in the list as shown in Figure 11:

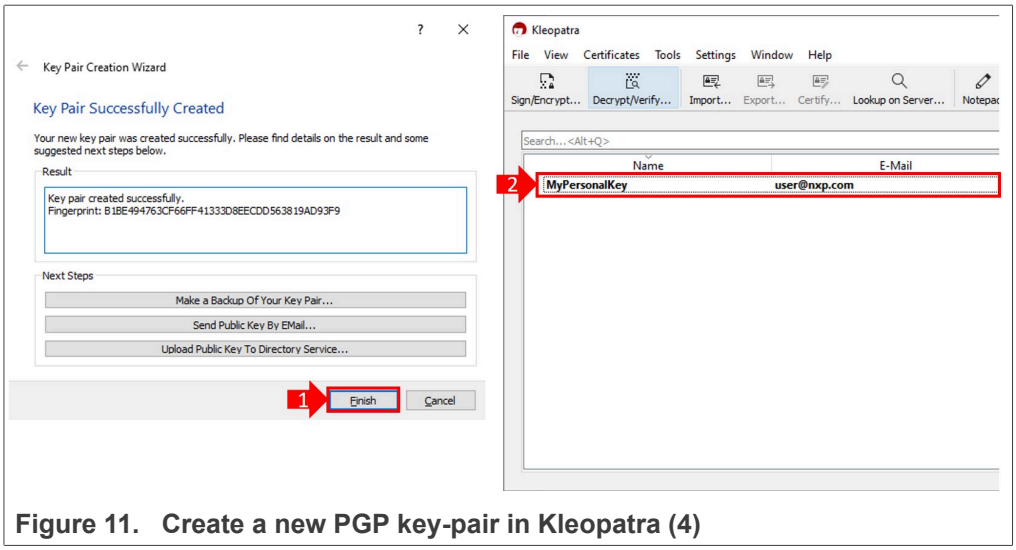

Figure 11. Create a new PGP key-pair in Kleopatra (4)

5. You need to import the EdgeLock 2GO public PGP key in order to encrypt the AES key, HMAC key or the confidential binary file. Copy the public key provided in Section 13 in a text file with *.asc* extension (e.g. *E2GO_public_key.asc*), and then import it in Kleopatra as shown in Figure 12:
(1) Click on the *Import* button in the top bar;
(2) Select the *.asc* file containing the EdgeLock 2GO public key and then click on *Open*;
(3) You should now see the key in Kleopatra with the name *Edgelock2go.support@nxp.com*.

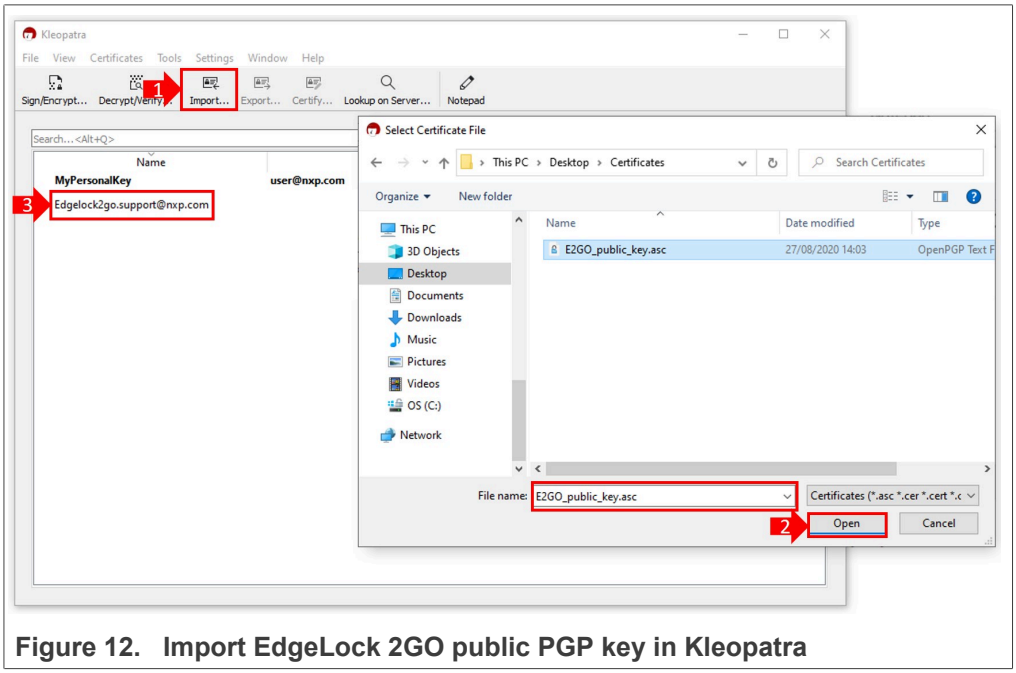

Figure 12. Import EdgeLock 2GO public PGP key in Kleopatra

6. You can now encrypt and sign the AES key, HMAC key or the confidential binary file. First, copy and paste the AES key, HMAC key or the confidential binary file content in Kleopatra Notepad as shown in Figure 13:

(1) Open the Kleopatra Notepad;
(2) **AES key**: copy and paste the key in the text field. The AES key MUST be in binary format (not ASCII encoded). It MUST only include hexadecimal characters and it MUST have a length of exactly 32 characters for a 128 bit AES key or 64 characters for a 256 bit AES key. Spaces, tabs and newline characters are allowed and will be filtered out by the system.
(3) **HMAC key**: copy and paste the key in the text field. The HMAC key MUST be in binary format (not ASCII encoded). It MUST only include hexadecimal characters and it MUST have a length between 2 characters (1 byte) and 512 characters (256 bytes). The length in bytes of the key must correspond to the length specified during object creation. Spaces, tabs and newline characters are allowed and will be filtered out by the system.
(4) **Confidential binary file:** copy and paste the binary file content in the text field. The content MUST be in binary format (not ASCII encoded). It MUST only include hexadecimal characters and it MUST have a size of up to 500 bytes (1000 hexadecimal characters). Spaces, tabs and newline characters are allowed and will be filtered out by the system.

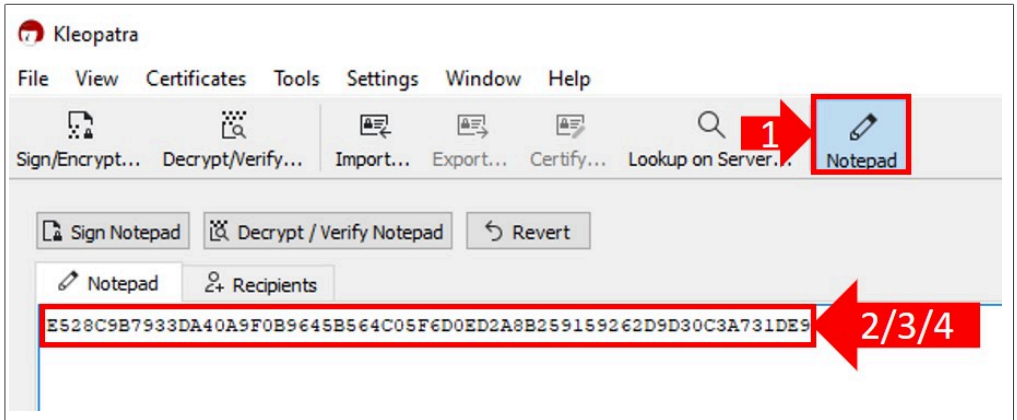

Figure 13. Paste AES key to encrypt in Kleopatra

7. Encrypt and sign the AES key, HMAC key or confidential binary file as shown in Figure 14:
(1) Open the *Recipients* tab;
(2) Tick the box next to *Sign as* label and select the key pair that you want to use to sign;
(3) Tick the box next to *Encrypt for others* label and select the EdgeLock 2GO public key imported before (*Edgelock2go.support@nxp.com*)
(4) Click on the *Sign / Encrypt Notepad* button to sign and encrypt the AES key, HMAC key or confidential binary file. You can now find the signed and encrypted PGP message in the Notepad tab as shown in Figure 15.
Copy and paste the content in a text file with *.asc* extension and save the file. You will need this file later when creating the AES key secure object, HMAC key secure object or the confidential binary file secure object in EdgeLock 2GO Managed.

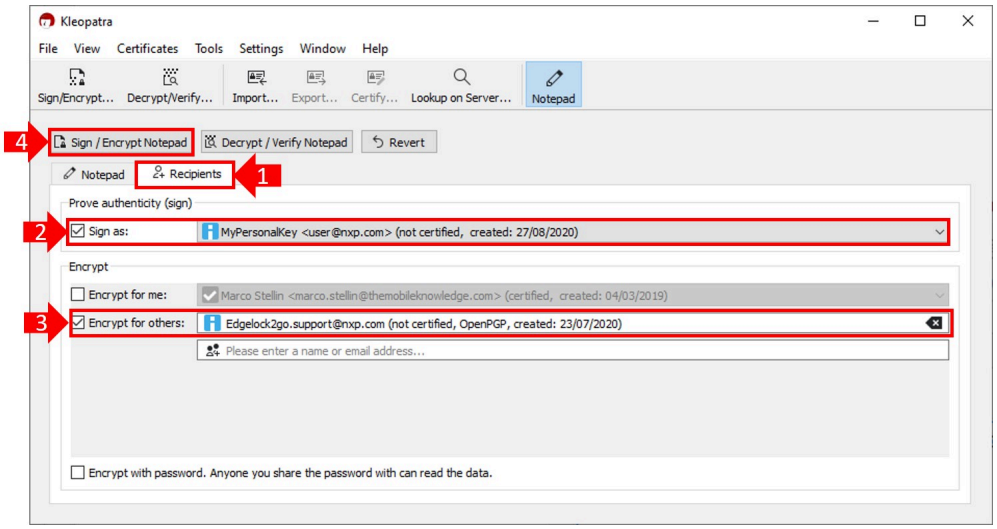

Figure 14. Encrypt and sign AES key in Kleopatra

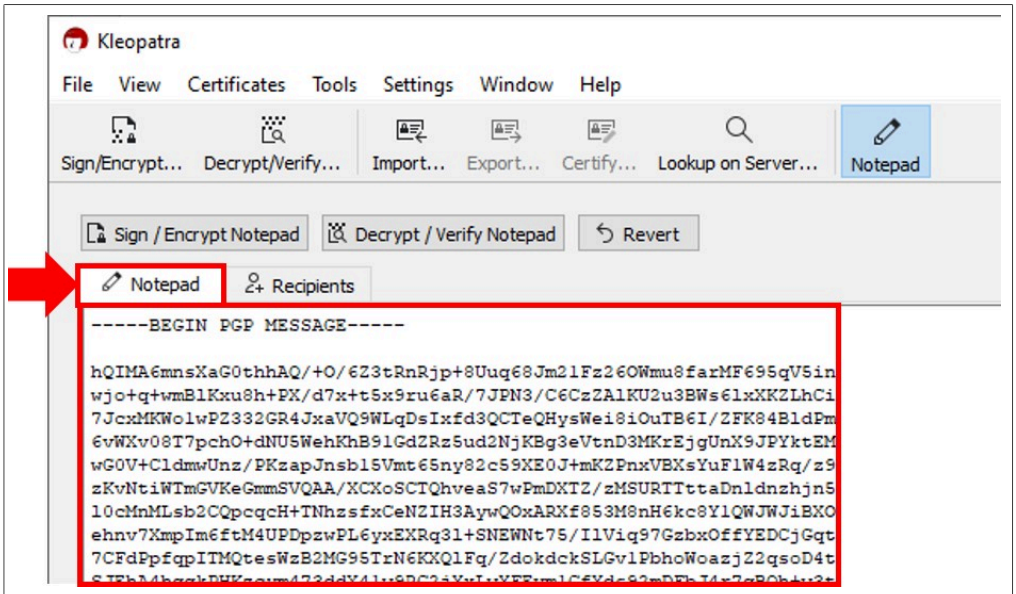

Figure 15. Encrypted and signed PGP message generated in Kleopatra

8. Finally, export and save the public key corresponding to the key pair that you used to sign the PGP message containing the key material or binary data as shown in Figure 16:
   (1) Select the key pair that you used to sign the PGP message;
   (2) Click on the *Export* button in the top bar;
   (3) Select the name of the public key and save it in your file system with *.asc* extension. You will need this file when creating the AES secure object, HMAC secure object or the confidential binary file secure object in EdgeLock 2GO Managed.

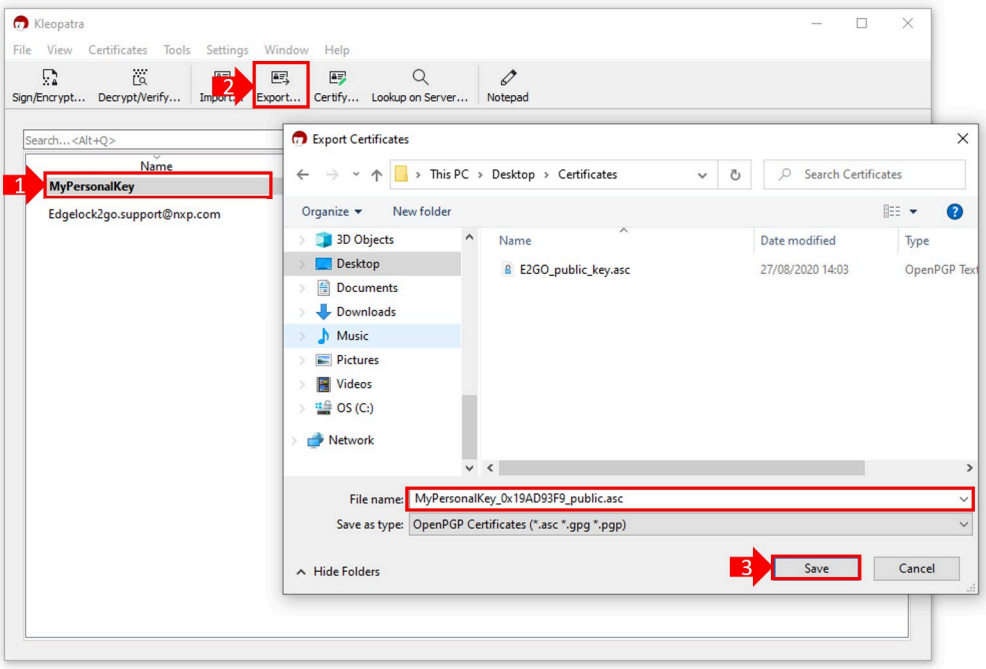

Figure 16. Export key-pair public key in Kleopatra

13 Annex 5: EdgeLock 2GO public PGP key

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: BCPG v1.60

mQINBF8ZP0ABEADrglEde2IFW4svYA7j7TVUxsZL48LuGLxgWpGfnooUaynQNCcG
e3EOt+jZT1tU8Y0kxovDnw9uX2tKiHqgKki7oUw4/+xkti2EK4vl2l1FdAWuX6yZ
DN/bQhNT9BZX78ycVJ+708h59k+Wm4PoXXKBDkHwrHif1LcxCj+sFE0fcbRTL1qh
bjOhdOVx6uiFEPzGjtM5MXwWOwkhPb1vEvxB2gGCZlG59yZkbLyhyf1P5cXjxVN0
drNixbKImj7d/77WkXtpHD5BSAWf8b8sFCL/lWjDLUGkg0SqlcKXlw4KYDhAL/Pa
5k9dQGEzgdCKmFaSiFPLzECiaUqlKzL13Gl2GyK2Jcw+vT5Dp475ZdUI86BLQ60x
qaQMfHOrK+8Gcq7dbLWyVuUHujOA75FWivLnCNtviKPggC3kgv6HWEDhSFgbFP5H
mLMwRJvXMIUCuQIhAQIoOseorVGs7pRjrEfySYIgnDtIdcEelXszCtlwvKLEM1zw
Psu0BT8M5MnoPPYW370oQB/633Lvt8XhaXBBeTMON9zSyRQHFuuXJT03H3Ljc4BP
C3s8vhvhkGZ2lTbaHmOB2648BzTwA9Y/f3aM7CslTvrLgHRRdw7tF8D8KGML7FmJ
tJKnKxnowD4l+OUKhPXBrH1St6SJh4cuXzBgJK2YIQurkYjZJLUsrgSfywARAQAB
tBtFZGdlbG9jazJnby5zdXBwb3J0QG54cC5jb22JAjIEEwECABwFAl8ZP0ACGwMF
CwkIBwIDFQgCAhYCAh4BAheAAAoJEL+cMm/OLyCPvxUP+wcqHBjK52OLZ/5H8/Bw
Dzt6cNZYxq+EsxjpZnaCtb9VceKQoII+0bnflB50hbABHhKjo2BWiQvBb7OoFdJl
nous8+vprOij/P4Kw6rUtFlZVJ67nWtRPECkdpcUJU/p38NWSaCNI5R5wanGrBrD
6PPrbQi7iVVGd/NUqRUUmY8te2HuVmvWrcp6VclbWvG24b+vRzSby/pVmhFqPgkP
IksgAJyL9FMbZ2G61u1WO9hT3k3niX90H/qDJknW6QR0zT3ajwN84BKMa7Xx+CzT
BK4+g+35Wy2iXR+bfpzj5RcNQtl9XOM0e4UUibcRORwqXJZ/6O4wOHhbWtrv9pWx
1/CVkY2m2hXOZZxWYZnAeOY3oo3xGYMvSd/aXRbq8Otwa2qWSQfaujyCf/7t/KYT
ssKhZxtSCikl9ZQ0M1vSK9OD4flVvc4Vl2yr8f0bAHcF6+K/zIV8evOROvR5zocT
34heqjW87X+xxe3nfyrVa9sK6KG+b7YKnrECbHzQR6++Y6HuAr9fjZ0/avqPG3QF
gU3WQOelotDGlTZPsMkRZArlVqSt7oqKdRIczM6/E71CsOG2TywZ7xETeQnFkVS4
bBTOAIlDRAQAqBc7VeOSvzehJTQ1CB/JcyaXyIjNXnhwXlXClZea4cFc4X2J+e2C
dkuGA430Zh6FLcyW1j2UrT7TuQINBF8ZP0ABEAC6lxbGyyEtzUz903zXpBfgKwxY
DDcfYnijX7FpAM6fErjWSOXrEG/MAisxICA9DVf2iMXyuRBIxIsi+Sau2yDtUpDR
7HJbK4KS0CUpYUZr7S+V8Qq64VULBy9R8BPWPel8I7UFZfMFx7tm/FQNMBR0m+fX
CNItcm/+nwVAUDdCrNyXPLeRYIu1D4yPL2nmggX7VUEivEZlWv3Kw7jpERgAeoSS
W1LfdUkIHzx2CS/MFFCv1kcQjrHO7uNE1xmK16Qqmcf24HjJzxArTSHQ94ToxKJB
p2LwiHRE1w7I4fgXO+x+v9/YKUKATfNYyTpz/rWVyYD3YpF7tnyONNERSWrfAqy1
BON8LsMqeEL5RSIVJgrGgmt2rDrUgipxcpClHM1wI2rAHYR4O6kktoMFpeUcBjZ4
r6BlK6NmVTE55lhl55+TiOJjhm+FOF6B1O2fKBVMeD/ONSq5J510gzkzKqoiMFq7
0KthitE8bgnmlqhbjpbW4e9/wSSph639Nfvj+/rik44S50V0qEx/3e/P6wO1MWEj
iadlWBc6d2CwwNXXqDmBn1t4KUUn4NgZFwdE0EonLp29T2gb2gNVRaCvXK0IWEHx
HFemsEOWhdBncOCoTlG4DP/4aYKVyKdhZTicijKF1m80ygqgCaBLhDk6DzZE3u+H
0SVvcCk9bC8/7Vu12QARAQABiQIfBBgBAgAJBQJfGT9AAhsMAAoJEL+cMm/OLyCP
tREP/1WZI0ghqW+qtCLT/EOQwceI6giyTZtTf3HZZEBujRGe/eEMPGlOxwVWCcd8
nvTsK6q3BYzXiRrynOh1Uivvldn19GPAgcco8HD4Goxl34MHXCN9o3U2TUkomXIZ
qN4/9EV8N69pp4Rk1bHY2spvru7nV5L3Bj+PQFpRnxAK3KnMSeK/GgtOrm2b6SGx
BaO+uxComJ9Awugfotnt4i589m1VCeyvGYWs8JKDSaxqUtYk0+X8o7RC0R58mWCO
wRPPwelsvdu7hrYcsmypEKJMUv1VBOO2HSkOyuAkRWDdATmQ1jIAoXYqKRMsB18T
tSJP0BGHS0A7zVfbKRwsyadWOU3P7rf8hkdiFpZs6k59J0btpnsyOVxVmGylcvIc
s/rUOI9q2EMHBwMMxH/ckyyIx5TRg3N6qqtuv2i9JZ89hWuzdRWVLYP4EdVj7FVI
/rMltHboedpVafgVlFkRc4c5nlbPD+iqP8uAFdlu8vV/QL5skfFOzBeue27I+AeA
On/DYWC3G0jmrge2IBGwNINJYoOtIgUT3boMKwt64JHOLa5qUPvG7y0XH30+crj4
6gCUg1ChiZSRY+IieBEm2wUfewnNW/P8N21N7Wf1VNm/xNjwb884XKatE95ZSes3
ggi+5+RmRrwd0W51siPe/LBc09t5P6MBz70WvBwiWqPKA7F1
=Bt0l
-----END PGP PUBLIC KEY BLOCK-----
```

14 Legal information

14.1 Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

14.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this data sheet expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

14.3 Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

EdgeLock — is a trademark of NXP B.V.

Figures

Contents

AN12642

EdgeLock 2GO developer guide for Foundries.io users

For more information, please visit: http://www.nxp.com

Date of release: 19 September 2022
Document identifier: AN12642
Document number: 614310