

# Enterprise Secure Key Manager

Enterprise Secure Key Manager v8.50.0

RESTful API Reference



## Imprint

Copyright 2022	Utimaco IS GmbH Germanusstr. 4 D-52080 Aachen Germany
Phone	AMERICAS +1-844-UTIMACO (+1 844-884-6226) EMEA +49 800-627-3081 APAC +81 800-919-1301
Internet e-mail	<a href="https://support.hsm.utimaco.com/">https://support.hsm.utimaco.com/</a> <a href="mailto:support@utimaco.com">support@utimaco.com</a>
Document Version	8.50.0
Date	2023-04-26
Status	
Document No.	2021-0056
All rights reserved	<p>No part of this documentation may be reproduced in any form (printing, photocopy or according to any other process) without the written approval of Utimaco IS GmbH or be processed, reproduced or distributed using electronic systems.</p> <p>Utimaco IS GmbH reserves the right to modify or amend the documentation at any time without prior notice. Utimaco IS GmbH assumes no liability for typographical errors and damages incurred due to them.</p> <p>All trademarks and registered trademarks are the property of their respective owners.</p>

# Table of Contents

<b>1</b>	<b>Introduction .....</b>	<b>4</b>
1.1	ESKM RESTful Web Services .....	4
1.1.1	Requests.....	5
1.1.2	Authentication.....	6
1.1.3	Response.....	6
1.2	Logs.....	6
1.3	Service Commands.....	9
<b>2</b>	<b>Methods.....</b>	<b>11</b>
2.1	Retrieve Key Information.....	11
2.2	Export Key bytes.....	12
2.3	Create Key .....	14
2.4	Update Key .....	15
2.5	Delete Key .....	16
2.6	Encrypt Data.....	17
2.7	Decrypt Data.....	19
2.8	Hashing .....	20
2.9	Signing .....	21
2.10	Verification.....	23
2.11	Get Key Custom Attributes.....	25
2.12	Add Key Custom Attributes .....	26
2.13	Public Key.....	27
2.14	Import Key .....	29
2.15	Certificate Signing Request.....	30
2.16	Random Number Generation .....	36
2.17	Status Codes .....	37

# 1 Introduction

This document provides the details of various ESKM REST methods, including a short description, examples for Requests and Responses, and Status Codes.

## 1.1 ESKM RESTful Web Services

The REpresentational State Transfer (REST) interface in ESKM provides an alternative method for cryptographic operations and key management. Various REST API methods can be used for this purpose. The REST server configuration in ESKM manages the secure communication between the client and server.

Requests and Responses are sent over HTTPS. ESKM version supports cryptographic operations and key management via REST.



The default port used for the communication between the client and the ESKM REST interface is 8443.

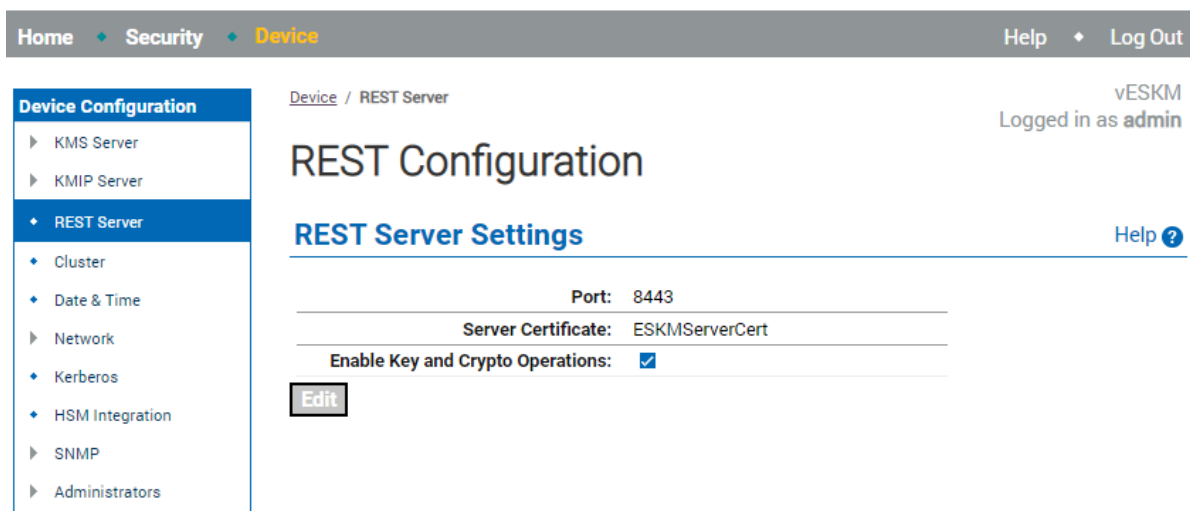


Use the format below when entering the curl command from the Windows command prompt:

```
curl -k --request POST -u <username>:<password> https://IP address/api/v6/key --data '{"keyName":"key_name","algorithm":"algorithm_name"}'
```

### Enabling Key and Crypto Operations

For performing cryptographic operations and key management using REST API methods, key operations must be enabled in **REST Configuration**. Please see the below figure.



The screenshot shows the 'REST Configuration' page. On the left is a 'Device Configuration' sidebar with a tree view containing: KMS Server, KMIP Server, REST Server (selected), Cluster, Date & Time, Network, Kerberos, HSM Integration, SNMP, and Administrators. The main content area is titled 'REST Configuration' and 'REST Server Settings'. It displays the following settings:

- Port: 8443
- Server Certificate: ESKMServerCert
- Enable Key and Crypto Operations:

There is an 'Edit' button below the settings.

Figure 1 : REST Server Settings

Refer the sub-section **REST server configuration**, in section 4 of the *Enterprise Secure Key Manager v User's Guide* for more information.

### 1.1.1 Requests

Each REST API request has:

- A request **URL**
- A **Method**
- A **Body** (for POST and PUT)

The request **URL** identifies the resource to apply the request. It defines the endpoint and the path.

The request **Method** indicates the method to be performed on the resource identified by the **URL**. The following HTTP methods are used.

- **GET** - Retrieves key information and exports key bytes
- **POST** - Creates a key
- **PUT** - Updates an existing key
- **DELETE** - Deletes a key

**POST** and **PUT** requests require a **Body** in JSON format.

## 1.1.2 Authentication

Authentication is required for all REST API requests. Basic Authentication with Username and Password is required.



The credentials of local users configured in ESKM must be used for authentication. Refer sub-section **Local users** in *Enterprise Secure Key Manager User's Guide* for more information.

## 1.1.3 Response

REST API returns an HTTP response code for each request. Responses for **GET** requests are received in JSON format.

### Response Codes

The success or failure of a request is indicated by the HTTP status code. In general, a status code of 200 indicates success and a status code of 4xx indicates failure.

## 1.2 Logs

ESKM maintains a log of the requests received by the REST server. Each operation is logged as a separate entry in the Activity Log. The logs can be viewed from the ESKM web console (**Device > Logs & Statistics > Log Viewer > Activity**).

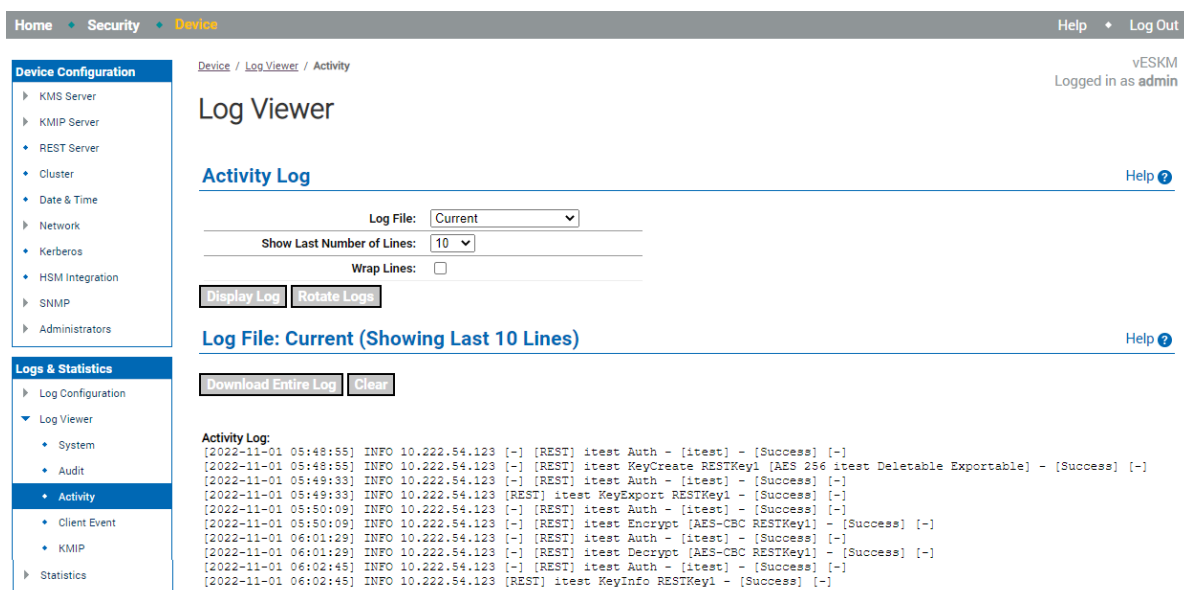


Figure 2 : Activity Log

The format of the Activity Log for the requests received by REST server is as follows:

```
<date> <priority> <ip> <REST> <user> <request type> <key> <detail>
<message>
```



The following table describes the fields that are present in the **Activity Log** for REST requests.

Table 1: Activity Log - REST requests

<b>Field</b>	<b>Description</b>
<b>date</b>	Enclosed in brackets [], the date field shows the date and time when the ESKM finished processing the request. The date and time are represented as follows: <code>yyyy-mm-dd hh:mm:ss</code> (local time zone).
<b>priority</b>	ERROR or INFO, depending on the result of the request.
<b>ip</b>	IP address of the client machine.
<b>REST</b>	The requests received by the REST server have <b>REST</b> in the Activity Log entries.

<b>Field</b>	<b>Description</b>
<code>user</code>	Authenticated user who issued the request.
<code>request type</code>	<p>Type of client request. The request type can take one of the below values.</p> <ul style="list-style-type: none"><li>▪ <code>KeyInfo</code> - Request to retrieve key information</li><li>▪ <code>KeyExport</code> - Request to export key bytes along with the key information</li><li>▪ <code>KeyCreate</code> - Request to create a key</li><li>▪ <code>KeyModify</code> - Request to update an existing key</li><li>▪ <code>KeyDelete</code> - Request to delete a key</li><li>▪ <code>Auth</code> - Authenticating the user</li><li>▪ <code>Encrypt</code> - Request to encrypt the data</li><li>▪ <code>Decrypt</code> - Request to decrypt the data</li><li>▪ <code>Hashing</code> - Request to generate digest.</li><li>▪ <code>Signing</code> - Request to generate signature.</li><li>▪ <code>Verification</code> - Request to verify the signature generated.</li><li>▪ <code>Custom Attributes</code> - Request to fetch the custom attributes for the key</li><li>▪ <code>Public Key</code> - Request to get the public key of a key</li><li>▪ <code>Import Key</code> - Request to import a key</li><li>▪ <code>Certificate Signing Request</code> - Request to sign an external certificate with ESKM CA</li><li>▪ <code>Random Key Generation</code> - Request to generate a random number of a specified bytes</li></ul>



<b>Field</b>	<b>Description</b>
<code>key</code>	<p>The name of the key in the request.</p> <p><code>Auth</code> request types have the authenticating user in [], instead of the key name.</p> <div style="background-color: #e6e6fa; padding: 5px;">  <code>Key</code> name is not applicable for hashing.         </div>
<code>detail</code>	<p>This field is enclosed in [] and lists:</p> <ul style="list-style-type: none"> <li>▪ Algorithm, size, and key owner for key operations and algorithm-mode and key used for encryption/decryption.</li> <li>▪ Algorithm for hashing.</li> <li>▪ Algorithm and key name used for signing and verification.</li> </ul> <p>Deletable and Exportable options are listed if set, for the key operations.</p> <div style="background-color: #e6e6fa; padding: 5px;">  <code>detail</code> is applicable only for <code>KeyCreate</code>, <code>Encrypt</code>, <code>Decrypt</code>, <code>Hashing</code>, <code>Signing</code> and <code>Verification</code> request types.         </div>
<code>message</code>	<p><code>[Success]</code> - If the operation is successful.</p> <p><code>[Error message]</code> followed by <code>[Failed]</code> - If the operation failed.</p> <p><code>[Verified OK]</code> - If the verification of signature is successful.</p> <p><code>[Verified Not OK]</code> - If the verification of signature is failed.</p>

### 1.3 Service Commands

The user can use the below CLI commands to start, stop, or restart the REST server.

- `service start restserver` - to start the REST server.
- `service stop restserver` - to stop the REST server.
- `service restart restserver` - to restart the REST server.

Refer the sub-section **CLI commands > Service commands**, in section 7 of the *Enterprise Secure Key Manager v8.41.0 User's Guide* for more information.

## 2 Methods

### 2.1 Retrieve Key Information

<b>Function</b>	To retrieve the key information
<b>Method</b>	GET
<b>URL</b>	https://<IP address>:<port>/api/v6/key/:keyname/info
<b>Requires Authentication</b>	Yes
<b>Request</b>	
<b>Example Request</b>	https://10.222.55.135:8443/api/v6/key/Key_1/info
<b>Body</b>	none
<b>Curl Command</b>	<pre>curl -k --request GET -u &lt;username&gt;:&lt;password&gt; https://&lt;IP address&gt;:&lt;port&gt;/api/v6/key/&lt;keyname&gt;/ info</pre>
<b>Response</b>	
<b>Content Type</b>	JSON

<b>Body</b>	<pre> {   "keyName": "Key_1",   "keyOwner": "kms-user1",   "algorithm": "AES",   "keyLength": 256,   "deletable": "Yes",   "exportable": "Yes",   "fingerprint": "87E81E8F436D89D2",   "defaultIV": "315331434023EA9469AAC72DAFECEC8B" } </pre>
<b>Additional Notes</b>	<ul style="list-style-type: none"> <li>▪ The key owner or the members of the Groups with <b>Full</b> permission to the key can retrieve the key information.</li> <li>▪ Supported algorithms and key length: <ul style="list-style-type: none"> <li>• AES-128, AES-192, AES-256.</li> <li>• HmacSHA1-160.</li> <li>• RSA-2048, RSA-3072, RSA-4096.</li> </ul> </li> </ul>

## 2.2 Export Key bytes

<b>Function</b>	To export the key bytes along with the key information
<b>Method</b>	GET
<b>URL</b>	https://<IP address>:<port>/api/v6/key/:keyname/
<b>Requires Authentication</b>	Yes
<b>Request</b>	
<b>Example Request</b>	https://10.222.55.135:8443/api/v6/key/Key_1/
<b>Body</b>	none

<b>Curl Command</b>	<pre>curl -k --request GET -u &lt;username&gt;:&lt;password&gt; https://&lt;IP address&gt;:&lt;port&gt;/api/v6/key/&lt;keyname&gt;</pre>
<b>Response</b>	
<b>Content Type</b>	JSON
<b>Body</b>	<pre>{   "keyName": "Key_1",   "keyOwner": "kms-user1",   "algorithm": "AES",   "keyLength": 256,   "deletable": "Yes",   "exportable": "Yes",   "fingerprint": "87E81E8F436D89D2",   "defaultIV": "315331434023EA9469AAC72DAFECEC8B",   "keyBytes":     "3DBD21BF3095FBE986AAB42A638E1936364A2823E5B2A4BFCEA     85CA456890420" }</pre>
<b>Additional Notes</b>	<ul style="list-style-type: none"> <li>▪ Only the keys that are configured as <b>Exportable</b> can be exported.</li> <li>▪ An exportable key can be exported by the key owner or the members of the Groups with <b>Full</b> permission to the key.</li> <li>▪ Supported algorithms and key length:       <ul style="list-style-type: none"> <li>• AES-128, AES-192, AES-256.</li> <li>• HmacSHA1-160.</li> <li>• RSA-2048, RSA-3072, RSA-4096.</li> </ul> </li> </ul>

## 2.3 Create Key

<b>Function</b>	<b>To create a key with group permission</b>
Method	POST
URL	https://<IP address>:<port>/api/v6/key/
Requires Authentication	Yes
<b>Request</b>	
Example Request	https://10.222.55.135:8443/api/v6/key/
Body	<pre>{   "keyName": "key_name",   "algorithm": "algorithm_name",   "keyOwner": "owner_name",   "keyLength": "key_length",   "exportable": "yes or no",   "deletable": "yes or no"   "permission": "group name" }</pre>
Curl Command	<pre>curl -k --request POST -u &lt;username&gt;:&lt;password&gt; https://&lt;IP address&gt;:&lt;port&gt;/api/v6/key --data '{"keyName":"key_name","keyOwner":"owner_name","algorithm" :"algorithm_name","keyLength":"key_length","exportable" :"yes or no","deletable":"yes or no","permission" : "cloud group"}'</pre>
<b>Response</b>	

<b>Function</b>	<b>To create a key with group permission</b>
<b>Content Type</b>	Text message
<b>Body</b>	none
<b>Additional Notes</b>	<ul style="list-style-type: none"> <li>▪ The “keyName” and “algorithm” are mandatory in the Request body.</li> <li>▪ The default “keyOwner” is the authenticated user.</li> <li>▪ The default “keyLength” is the default key length for the algorithm in the request.</li> <li>▪ The default value for “exportable” and “deletable” is “no”.</li> <li>▪ Supported algorithms and key length: <ul style="list-style-type: none"> <li>• AES-128, AES-192, AES-256.</li> <li>• HmacSHA1-160.</li> <li>• RSA-2048, RSA-3072, RSA-4096.</li> </ul> </li> </ul>

## 2.4 Update Key

<b>Function</b>	To update an existing key
<b>Method</b>	PUT
<b>URL</b>	https://<IP address>:<port>/api/v6/key/:keyname
<b>Requires Authentication</b>	Yes
<b>Request</b>	
<b>Example Request</b>	https://10.222.55.163:8443/api/v6/key/Key_9

<b>Body</b>	<pre>{   "keyOwner": "owner_name" }</pre>
<b>Curl Command</b>	<pre>curl -k --request PUT -u &lt;username&gt;:&lt;password&gt; https://&lt;IP address&gt;:&lt;port&gt;/api/v6/key/&lt;keyname&gt; -- data '{"keyOwner":"owner_name"}'</pre>
<b>Response</b>	
<b>Content Type</b>	Text message
<b>Body</b>	none
<b>Additional Notes</b>	<ul style="list-style-type: none"> <li>▪ “keyOwner” is mandatory in the request body.</li> <li>▪ Only the key owner can be modified using REST commands.</li> <li>▪ The current key owner or the members of the Groups with Full permission to the key can update the key.</li> <li>▪ Supported algorithms and key length: <ul style="list-style-type: none"> <li>• AES-128, AES-192, AES-256.</li> <li>• HmacSHA1-160.</li> <li>• RSA-2048, RSA-3072, RSA-4096.</li> </ul> </li> </ul>

## 2.5 Delete Key

<b>Function</b>	To delete an existing key
<b>Method</b>	DELETE
<b>URL</b>	https://<IP address>:<port>/api/v6/key/: <i>keyname</i>



<b>Requires Authentication</b>	Yes
<b>Request</b>	
<b>Example Request</b>	https://10.222.55.163:8443/api/v6/key/Key_10
<b>Body</b>	none
<b>Curl Command</b>	<pre>curl -k --request DELETE -u &lt;username&gt;:&lt;password&gt; https://&lt;IP address&gt;:&lt;port&gt;/api/v6/key/&lt;keyname&gt;</pre>
<b>Response</b>	
<b>Content Type</b>	Text message
<b>Body</b>	none
<b>Additional Notes</b>	<ul style="list-style-type: none"> <li>▪ Only the keys that are configured as <b>Deletable</b>, can be deleted.</li> <li>▪ A deletable key can be deleted by the key owner or the members of the Groups with <b>Full</b> permission to the key.</li> <li>▪ Supported algorithms and key length: <ul style="list-style-type: none"> <li>• AES-128, AES-192, AES-256.</li> <li>• HmacSHA1-160.</li> <li>• RSA-2048, RSA-3072, RSA-4096.</li> </ul> </li> </ul>

## 2.6 Encrypt Data

<b>Function</b>	To encrypt data using an existing key.
<b>Method</b>	POST

<b>URL</b>	https://<IP address>:<port>/api/v6/key/<key name>/enc
<b>Requires Authentication</b>	Yes
<b>Request</b>	
<b>Example Request</b>	https://10.222.55.135:8443/api/v6/key_1/enc/
<b>Body</b>	<pre>{   "algorithm": "AES-CBC", "input": "SaiSDCElXAFtM5JVczRylwe4ZaAMzVh736T0tH2W6sQ=" }</pre>
<b>Curl Command</b>	<pre>curl -k --request POST -u &lt;username&gt;:&lt;password&gt; https://&lt;IP address&gt;:&lt;port&gt;/api/v6/key_1/enc --data {"algorithm": "AES-CBC", "input": "SaiSDCElXAFtM5JVczRylwe4ZaAMzVh736T0tH2W6sQ="}</pre>
<b>Response</b>	
<b>Content Type</b>	JSON
<b>Body</b>	<pre>{ "output" : "QthKahJRLUtKyDxAyiAknA==" }</pre>

<b>Additional Notes</b>	<ul style="list-style-type: none"> <li>▪ Algorithm can be AES-CBC or AES-ECB.</li> <li>▪ Input and output are base64 encoded.</li> <li>▪ Maximum length of input plaintext is 2048.</li> <li>▪ Only AES keys can be used for encryption.</li> <li>▪ Only the keys that are configured as Exportable can be used for encryption.</li> <li>▪ An exportable key can be used for encryption by the key owner or the members of the Groups with Full permission to the key.</li> <li>▪ The "input" and "algorithm" are mandatory in the Request body.</li> </ul>
-------------------------	---

## 2.7 Decrypt Data

<b>Function</b>	To decrypt data using an existing key.
<b>Method</b>	POST
<b>URL</b>	https://<IP address>:<port>/api/v6/key/<key name>/dec
<b>Requires Authentication</b>	Yes
<b>Request</b>	
<b>Example Request</b>	https://10.222.55.135:8443/api/v6/key_1/dec/
<b>Body</b>	<pre style="border: 1px solid #ccc; padding: 10px;">{   "algorithm": "AES-CBC", "input": "SaiSDCElXAFtM5JVczRylwe4ZaAMzVh7 36T0tH2W6sQ=" }</pre>

<b>Curl Command</b>	<pre>curl -k --request POST -u &lt;username&gt;:&lt;password&gt; https://&lt;IP address&gt;:&lt;port&gt;/api/v6/key_1/dec --data {"algorithm":"AES-CBC","input":"SaiSDCElXAFtM5JVczRylwe4ZaAMzVh736T0tH2W6sQ="}</pre>
<b>Response</b>	
<b>Content Type</b>	JSON
<b>Body</b>	<pre>{ "output" : "QthKahJRLUtKyDxAyiAknA==" }</pre>
<b>Additional Notes</b>	<ul style="list-style-type: none"> <li>▪ Algorithm can be AES-CBC or AES-ECB.</li> <li>▪ Input and output are base64 decoded.</li> <li>▪ Maximum length of input plaintext is 2048.</li> <li>▪ Only AES keys can be used for decryption.</li> <li>▪ Only the keys that are configured as Exportable can be used for decryption.</li> <li>▪ An exportable key can be used for decryption by the key owner or the members of the Groups with Full permission to the key.</li> <li>▪ The "input" and "algorithm" are mandatory in the Request body.</li> </ul>

## 2.8 Hashing

<b>Function</b>	To generate digest.
<b>Method</b>	POST
<b>URL</b>	https://<IP address>:<port>/api/v6/digest/:algorithmName/
<b>Requires Authentication</b>	Yes

<b>Request</b>	
<b>Example Request</b>	https://10.222.55.135:8443/api/v6/digest/:SHA-256/
<b>Body</b>	<pre>{   "input" : "&lt;base64 encoded text&gt;" }</pre>
<b>Curl Command</b>	<pre>curl -k --request POST -u &lt;username&gt;:&lt;password&gt; &lt;https://&lt;IP&gt; address&gt;:&lt;port&gt;/api/v6/digest/ &lt;SHA-256&gt; --data '{"input" : "aGVsbG8gd29ybGQ="}'</pre>
<b>Response</b>	
<b>Content Type</b>	JSON
<b>Body</b>	<pre>{"output" : &lt;digest generated&gt;}</pre>
<b>Additional Notes</b>	<ul style="list-style-type: none"> <li>▪ Algorithm can be SHA-1, SHA-512, SHA-256, MD5.</li> <li>▪ Input is base64 encoded.</li> <li>▪ Maximum length of input plaintext is 2048.</li> <li>▪ The "input" is mandatory in the Request body.</li> </ul>

## 2.9 Signing

<b>Function</b>	To generate signature from the specified RSA key and algorithm.
<b>Method</b>	POST
<b>URL</b>	https://<IP address>:<port>/api/v6/key/:keyName/sign/

<b>Requires Authentication</b>	Yes
<b>Request</b>	
<b>Example Request</b>	https://10.222.55.135:8443/api/v6/key_1/:keyName/sign/
<b>Body</b>	<pre>{   "input" : "&lt;base64 encoded plain text&gt;"   "algorithm" : "PSS_SHA256" }</pre>
<b>Curl Command</b>	<pre>curl -k --request POST -u &lt;username&gt;:&lt;password&gt; &lt;https://&lt;IP&gt; address&gt;:&lt;port&gt;/api/v6/ key_1&lt;KeyName&gt;sign --data '{"input" : "aGVsbG8gd29ybGQ=", "algorithm" : "PKCS1_SHA256"}'</pre>
<b>Response</b>	
<b>Content Type</b>	JSON
<b>Body</b>	<pre>{"output" : &lt;base64 encoded signature&gt;}</pre>

<b>Additional Notes</b>	<ul style="list-style-type: none"> <li>▪ Key Algorithms can be RSA-2048,RSA-3072,RSA-4096.</li> <li>▪ Supported padding are PSS and PKCS1.</li> <li>▪ Supported algorithms are SHA-256 and SHA-512.</li> <li>▪ Input and output are base64 encoded.</li> <li>▪ Maximum length of input plaintext is 2048.</li> <li>▪ Only RSA keys can be used for generating signature.</li> <li>▪ The algorithm will be a combination of padding and hashing algorithm:             <ul style="list-style-type: none"> <li>• PSS_SHA256</li> <li>• PKCS1_SHA256</li> <li>• PSS_SHA512</li> <li>• PKCS1_SHA512</li> </ul> </li> <li>▪ The "input" and "algorithm" are mandatory in the Request body.</li> </ul>
-------------------------	--

## 2.10 Verification

<b>Function</b>	To verify the signature with the specified RSA key and algorithm.
<b>Method</b>	POST
<b>URL</b>	https://<IP address>:<port>//api/v6/key/:keyName/verify/
<b>Requires Authentication</b>	Yes
<b>Request</b>	
<b>Example Request</b>	https://10.222.55.135:8443//api/v6/key/:key_1/:keyName/verify/

<b>Body</b>	<pre>{   "signature" : "&lt;base64 encoded signature&gt;"   "algorithm" : "PSS_SHA256"   "input" : "&lt;base64 encoded plain text to be verified&gt;" }</pre>
<b>Curl Command</b>	<pre>curl -k --request POST -u &lt;username&gt;:&lt;password&gt; &lt;https://&lt;IP&gt; address&gt;:&lt;port&gt;/api/v6/ key:&lt;KeyName&gt;verify --data '{"signature" : "c53ii/h2yPhZy6SWqQSD EqjCaMZiGmdLe+/ sJ9G7IP6cY1pf1jm4xqbu /MqUTy/eLU+du4TbS02G3XSus+zYoiYvwpmIq+NPE6GDtd/ bvJ7R0+ln+G9n0nKKs4pfmfVhym +R5w5UjVUHyt757A8ziVfm2VtVeifWdjikySnrGRiDJygrg PK2Kqp2RQCc27ZGYDPFCMqQz2c5LS9QYax443qMZJsx4famM73Ye +wwi39eeY8zYsWxTij0vS4pwg46/ kCh0X8qt3l0Ti50hc7NbYZIjINi1WXG2Mci7PWPl9Ni7XE0F7wn /2XI01zfDtN9osVoqztWJH1vqXR4XV2i8Q==" , "algorithm" : "PKCS1_SHA256","input" : "aGVsbG8gd29ybGQ="}'</pre>
<b>Response</b>	
<b>Content Type</b>	JSON
<b>Body</b>	<pre>{"verified" : "&lt;OK/Not OK&gt;"}</pre>



<b>Additional Notes</b>	<ul style="list-style-type: none"> <li>▪ Key Algorithms can be RSA-2048,RSA-3072,RSA-4096.</li> <li>▪ Supported padding are PSS,PKCS1.</li> <li>▪ Supported hashing are SHA-256,SHA-512.</li> <li>▪ Input and signature are base64 encoded.</li> <li>▪ Maximum length of input plaintext is 2048.</li> <li>▪ The RSA key used to generate signature can be used for verifying the signature.</li> <li>▪ The “input”, “signature” and “algorithm” are mandatory in the Request body.</li> </ul>
-------------------------	--

## 2.11 Get Key Custom Attributes

<b>Function</b>	To fetch the custom attributes for the key
<b>Method</b>	GET
<b>URL</b>	https://<IP address>:<port>/api/v6/key/: <i>keyname</i> /attributes
<b>Requires Authentication</b>	Yes
<b>Request</b>	
<b>Example Request</b>	https://10.222.55.135:8443/api/v6/key/Key_1/attributes
<b>Body</b>	none
<b>Curl Command</b>	<pre>curl -k --request GET -u &lt;username&gt;:&lt;password&gt; https://&lt;IP address&gt;:&lt;port&gt;/api/v6/key/&lt;keyname&gt;/ attributes</pre>

<b>Response</b>	
<b>Content Type</b>	JSON
<b>Body</b>	<pre>{   "attribute1" : "12345" ,   "attribute2" : "6789" }</pre>
<b>Additional Notes</b>	<ul style="list-style-type: none"> <li>▪ Used only for fetching the custom attributes that are already available.</li> <li>▪ The activities (Success/Failure) are logged in activity log.</li> </ul>

## 2.12 Add Key Custom Attributes

<b>Function</b>	To add key custom attributes
<b>Method</b>	POST
<b>URL</b>	<<https://<IP>> address>:<port>/api/v6/key/key3/attributes -d
<b>Requires Authentication</b>	Yes
<b>Request</b>	
<b>Example Request</b>	<<https://<IP>> address>:<port>/api/v6/key/key3/attributes -d

<b>Body</b>	<pre>{   "Name1": "Value1"   "Name2": "Value2"   "Name3": "Value3"   "Name4": "Value4"   "Name5": "Value5" }</pre>
<b>Curl Command</b>	<pre>curl -k --request POST -u user1:atalla1234 &lt;https://&lt;IP&gt; address&gt;/api/v6/key/key3/attributes -d '{"Name1": "Value1", Name1: "Value1", Name2: "Value2", Name3: "Value3", Name4: "Value4", Name5: "Value5", Name6: "Value6", N ame7: "Value7", Name8: "Value8", Name9: "Value9", Name10: "V alue10", Name11: "Value11"}</pre>
<b>Response</b>	
<b>Content Type</b>	Text message
<b>Body</b>	None
<b>Additional Notes</b>	<ul style="list-style-type: none"> <li>▪ Can add up to a max of 32 attributes per request.</li> <li>▪ The same request may be used to add and update custom attributes.</li> </ul>

## 2.13 Public Key

<b>Function</b>	To get the public key of a key.
<b>Method</b>	GET
<b>URL</b>	https://<IP address>:<port>/api/v6/key/: <i>keyname</i> /pub
<b>Requires Authentication</b>	Yes

<b>Request</b>	
<b>Example Request</b>	https://10.222.55.135:8443/api/v6/key/Key_1/pub
<b>Body</b>	none
<b>Curl Command</b>	<pre>curl -k --request GET -u &lt;username&gt;:&lt;password&gt; https://&lt;IP address&gt;:&lt;port&gt;/api/v6/key/&lt;keyname&gt;/ pub</pre>
<b>Response</b>	
<b>Content Type</b>	JSON
<b>Body</b>	<pre>{   "publicKey" : "-----BEGIN RSA PUBLIC KEY----- MIIBCgKCAQEAuuSCDA6AYFGJeM/2Bd1YTcQy/ 4KPM1cwUViTOMIMucvnzs873C6q Ncw7cVh1u8v41XsQ51oKfGLGAPEeM5xF4EKZgr1xn+J+MbKngDaC wy+P8AYCRKGD qXJ/ 4MqE9BKeCTWQGlqiJii8gS9ttSoBPoNe+Euz6kpzuBo7vX3NEmNM VVR1aSji Z6PJNkR9j92WwF3BrqZWrDMS721zQbuPVb3Mdx1h2EB66SXV7IrX 77rIZ0w066xu 5QDZ8HeWyGdsfljF2JU0crzvB1RAWPHVtfo4YY0oMwzdr8HAUfiJ 5pwFiSTfRUKn JN7uYBDv8mEXqEDXCTWRwcNAstwjwjEIQIDAQAB -----END RSA PUBLIC KEY----- " }</pre>
<b>Additional Notes</b>	<ul style="list-style-type: none"> <li>▪ Only the keys that are configured as <b>Exportable</b> can be exported.</li> <li>▪ Supported algorithms and key length: <ul style="list-style-type: none"> <li>• RSA-2048, RSA-3072, RSA-4096.</li> </ul> </li> </ul>

## 2.14 Import Key

<b>Function</b>	To import a key
<b>Method</b>	POST
<b>URL</b>	https://<IP address>:<port>/api/v6/key/
<b>Requires Authentication</b>	Yes
<b>Request</b>	
<b>Example Request</b>	https://10.222.55.135:8443/api/v6/key/
<b>Body</b>	<pre> {   "keyName": "key_name",   "algorithm": "algorithm_name",   "keyOwner": "owner_name",   "exportable": "yes or no",   "deletable": "yes or no",   "keyBytes": "key_bytes" } </pre>
<b>Curl Command</b>	<pre> curl -k --request POST -u &lt;username&gt;:&lt;password&gt; https://&lt;IP address&gt;:&lt;port&gt;/api/v6/key --data '{"keyName":"key_name","keyOwner":"owner_name","algorithm" :"algorithm_name","exportable":"yes or no","deletable" :"yes or no","keyBytes": "key_bytes"}' </pre>
<b>Response</b>	
<b>Content Type</b>	Text message

<b>Body</b>	none
<b>Additional Notes</b>	<ul style="list-style-type: none"> <li>▪ The “keyName”, “keyBytes” and “algorithm” are mandatory in the Request body.</li> <li>▪ The default “keyOwner” is the authenticated user.</li> <li>▪ The default value for “exportable” and “deletable” is “no”.</li> <li>▪ Supported algorithms and key length: <ul style="list-style-type: none"> <li>• AES-128, AES-192, AES-256</li> <li>• DES-EDE-168</li> <li>• HmacSHA1-160</li> <li>• RSA-2048, RSA-3072, RSA-4096</li> </ul> </li> </ul>

## 2.15 Certificate Signing Request

<b>Function</b>	To sign an external certificate with ESKM CA
<b>Method</b>	POST
<b>URL</b>	<https://<IP> address>:<port>/api/v6/ca/:caname/sign
<b>Requires Authentication</b>	Yes
<b>Request</b>	
<b>Example Request</b>	https://10.222.55.135:8443/api/v6/ca/:ESKMCA/sign

**Body**

```
{ "certRequest":  
  "-----BEGIN CERTIFICATE REQUEST-----  
  \nMIIDAzCCAesCAQAwgZsxDTALBgNVBA  
  MTBEVTS00xFTATBgNVBAoTDE9yZ2FuaXph  
  \ndGlvbjEdMBsGA1UECzMUSW5mb3JtYXRpb24gU2  
  VjdXJpdHkxETAPBgNVBACTCENh  
  \nbXBiZWxsMQswCQYDVQQIEwJDQ  
  TELMAkGA1UEBhMCMVVMxJzAlBgkqhkiG9w0BCQEW  
  \nGGluZm9zZWNAb3JnYW5pemF0aW9uLmNvbTCC  
  ASIwDQYJKoZIhvcNAQEBBQADggEP\nADCCAQoC  
  ggEBAJ1+SfudpgUash+fgkg5Q5GGZBshBCehPxLgAB  
  Lptomxbtxiags0\nel0UI9cG0wAK4uLSkhFDzozX  
  LUx1J6t5mc2G1lOXAXZ4ys5RTvfk1PX8Smm0kjUE  
  \nDz6Gr2I7ELMsSqMQYItHL9uewG/rNxc7uuJ0p  
  EhGvJ7L8BFVD6J8VbDijKHY8bFF\n1K+8/LzrQK  
  wgZ6uqcLey74rF0KQ4n7miADq0ySK1LgTYasvBq  
  0hnWjgbNaPTMxtR\n7DE6Dqk1egFqJbihkPlSmW  
  b4rM+o0Nm0YcPk2ujW7dFv5NAP6pUpuH98reKIuMG1  
  \nGb5tgtlyNGiWJn4XF3lWqsZ80+RlL68o18kCAwE  
  AAaAiMCAGCSqGSIB3DQEJDjET\nMBEwDwYDVR0RBAgw  
  BocECt43hzANBqkqhkiG9w0BAQsFAA0CAQEaiizL5S  
  OJ0uZE\nnvsbDT9XkECcCxlnDwAXDaqVcQN9Bqc6K6r
```

J32vcVZHp1XAt+a+crBDlN9VzTMgYn0xMD86JnKj8rp

N05lMz9r+yYXUywogYL9YqeJdLUoXQj6V87AY4seFw

1kVl bvTaS\n8dB5orskAnbR72omYiWdmKih

PpPrakVCzD7fT8KskK+KZouKrxwY1TMjLHq6XULS

\n2ZH/Vqhn7Ek0rAGA2ftr48KLEY1gr/wQGvjIT

baggSL2Y58Iiwy1LE+vXqRngBCp\n+5NwoHtmtfj

VeD161ZWqY1fYJTY1SUqmKoj5Ug1B0RKL8TiAogb

BEsQbFT3RRaLZ\nouPqvY52xQ==

\n-----END CERTIFICATE REQUEST-----"

"certUsage":"Server" "certDuration":365}



**Curl Command**

```
curl -k --request POST -u <username>:<password>  
  
https://<IP address>:<port>/api/v6/ca:<caname>/sign --  
data  
  
'{"certRequest":"-----BEGIN CERTIFICATE REQUEST-----  
\nMIIDAzCCAesCAQAwgZsxDALBgNVBAMTBEBVTS00  
  
xFTATBgNVBAoTDE9yZ2FuaXph\ndGlvbjEdMBsGA1UECx  
  
MUSW5mb3JtYXRpb24gU2VjdXJpdHkxETAPBgNVBACtCENh  
  
\nbXBiZWxsMQswCQYDVQIEwJDQTElMAkGA1UEBhMCMVVMx  
  
JzAlBgkqhkiG9w0BCQEW\nGGluZm9zZWNAb3JnYW5pemF0a  
  
W9uLmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEP\nADCCAQ  
  
oCggEBAJ1+SfudpgUash+fgkg5Q5GGZBshBCehPxLgABLpt  
  
omxbtxiags0\nE10UI9cG0wAK4uLSkhFDzozXLUx1J6t5mc  
  
2G1lOXAXZ4ys5RTvfk1PX8Smm0kjUE\nDz6Gr2I7ELMsSqM  
  
QYIthL9uewG/rNxc7uuJ0pEhGvJ7L8BFVD6J8VbDiJkHY8b  
  
FF\n1K+8/LzrQKwgZ6uqcLey74rF0KQ4n7miAdq0ySK1LgT  
  
YasvBq0hnWjgbNaPTMxtR\n7DE6Dqk1egFqJbikhPlSmWb  
  
4rM+o0Nm0YcPk2ujW7dFv5NAP6pUpuH98reKIuMG1\nGb  
  
5tgtlyNGiWJn4XF3lWqsZ80+RlL68o18kCAwEAAaAiMCAGC  
  
SqGSib3DQEJDjET\nMBEwDwYDVR0RBAgwBocECt43hzANBg  
  
kqhkiG9w0BAQsFAAOCQAQEaiizL5S0J0uZE\nnvsbDT9XkECc  
  
Cxl9NdwAXDaqVcQN9Bqc6K6rJ32vcVZHp1XAt+a+crBDlN9  
  
VzTMgY\n0xMD86JnKj8rpN05lMz9r+yYXUywogYL9YqeJdL  
  
UoXQj6V87AY4seFw1kVlbtas\n8dB5orskAnbR72omYiWdmK
```

	<pre>ihPpPrakVCzD7fT8KskK+KZouKrxwY1TMjLHq6XULS\n2ZH/ Vqhn7Ek0rAGA2ftr48KLEY1gr/wQGvjITbaggSL2Y58Iiwy1LE +vXqRngBCp\n+5NwoHtmtfjVeD161ZWqY1fYJTY1SUqmKoj5Ug 1B0RKL8TiAogbBEsQbFT3RRaLZ\nouPqvY52xQ== \n-----END CERTIFICATE REQUEST-----", "certUsage":"Server","certDuration":365}</pre>
<b>Response</b>	
<b>Content Type</b>	JSON

<b>Body</b>	<pre>{ "certificate" : "-----BEGIN CERTIFICATE----- MIID6zCCAt0gAwIBAgIBLDANBgkqhkiG9w0BAQsFADCB0jEL MAkGA1UEBhMCVVMxCzAJBgNVBAGTAkNBMRUwDwYDVQQHEwhD YW1wYmVsbDEVMBMGA1UEChMNT3JnYW5pemF0aW9uMR0wGwYD VQQLExRJbmZvcmlhdGlvbiBTZW51cm10eTEUMBIGA1UEAxM LRVNLTUxvY2FsQ0ExJzAlBgkqhkiG9w0BCQEWGGluZm9zZW NAb3JnYW5pemF0aW9uLmNvbTAeFw0yMjA1MTcyMjU3MTFaF w0yMjA3MDIyMjU3MTFaMIGbMQswCQYDVQQGEwJVUzELMAk GA1UECBMCQ0ExETAPBgNVBACTCENhbXBibWZwXsMRUwEwYDV QQKEwxPcmdhbm16YXRpb24xHTAbBgNVBAsTFEluZm9zYW51 0aW9uIFNlY3VyaXR5MQ0wCwYDVQQDEwRFU0tNMSUwDwYDV KoZIHvcNAQkBFhpbmZvc2VjQG9yZ2FuaXphdGlvbi5jb 20wggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQA Cdfkn7naYFGrIfn4JI0U0RhmQbIQQnoT8S4AAS6baJsW7 cYmoLDhJdFCPXbtMACuLi0pIRQ86M1y1MdSereZnNhtZT lwF2eMr0UU735NT1/EpptJI1BA8+hq9i0xCzLEqjEGCLRy /bnsBv6zcX07riTqRIRryey/ARVQ+iffWw4iZB2PGxRdSv vPy860CsIGerqnc3su+KxdCk0J+5ogA6jskitS4E2GrLwat IZ1o4GzWj0zMbUewx0g6pNXoBaiW4oZD5Up1m+KzPqNDZ jmHD5Nro1u3Rb+TQD+qVKbh/fk3iiljBtRm+bYLZcjRoliz +Fxd5VqrM/NPkZS+vKNfJAgMBAAGjMTAvMAkGA1UdEwQ CMAAwEQYJYIZIAyb4QgEBBAQDAgZAMA8GA1UdEQQIMAa</pre>
-------------	---

	<pre> HBAreN4cwDQYJKoZIhvcNAQELBQADggEBAAYquRUiGiuy pvs0iI6bi993yRFCTx4zx5/CM5N48v1lGnodgzXbeQdzJW nfk49XstSqVm5cVWoQ1W0fY0htM++L W08KG0r0GNVqaq50Tb10F6LBsewcc/ddfT4fGHdXJJr /jM342Bd7R0uK86atKgtRWct/hooX3lmdiuaD63R+ 9Tb89vStrkB7Vc5Ftsfr+kt9wszQqRe1j7xbSZ0lX tobPx0WxZDRlatQcon6i0nPBa70zuiiyAjYyGD7if +l1RbU0WIIup63RJbUAN2Jjl10aUhKV6SWKzrR6dwQ5sWsS0HErr /QDW4tcR3Zg7PwbEgdgqL9+ZkZqsmpsw9s18= -----END CERTIFICATE----- " }                 </pre>
<p><b>Additional Notes</b></p>	<ul style="list-style-type: none"> <li>▪ The "certRequest" and "certUsage" are mandatory in the Request body.</li> <li>▪ "certDuration" is optional. If the user does not specify the duration, the specified CA duration will be allocated for the certificate.</li> <li>▪ "certUsage" can be "Server", "Client" or "ServerClient".</li> </ul>

## 2.16 Random Number Generation

<p><b>Function</b></p>	<p>To generate a random number of a specified bytes</p>
<p><b>Method</b></p>	<p>GET</p>
<p><b>URL</b></p>	<p>https://&lt;IP address&gt;:&lt;port&gt;/api/v6/randgen/:size</p>
<p><b>Requires Authentication</b></p>	<p>Yes</p>

<b>Request</b>	
<b>Example Request</b>	https://10.222.55.191:8443/api/v6/randgen/128
<b>Body</b>	none
<b>Curl Command</b>	<pre>curl -k --request GET -u &lt;username&gt;:&lt;password&gt; https://10.222.55.191:8443/api/v6/randgen/128</pre>
<b>Response</b>	
<b>Content Type</b>	JSON
<b>Body</b>	<pre>{ "randomNumber" : "wyy1323XBxcIJVId" }</pre>
<b>Additional Notes</b>	<ul style="list-style-type: none"> <li>▪ Generate random number up to 4096 bytes.</li> <li>▪ The output is base 64 encoded.</li> </ul>

## 2.17 Status Codes

Code	Error message	Description
200	Operation Successful	Successfully retrieved the key information.
	OK	Verification successful.
	Not OK	Verification not successful.
	Operation Successful, but failed to save properties	Successfully retrieved the key information but failed to save the properties.

	Successfully updated custom attribute(s).	Successfully added and updated custom attributes
	Internal Server Error	Failed to add and update custom attributes when function validation (iconset metadata) fails.
400	Invalid JSON	Request body is missing. or A non-updatable field is given in the body.
		If there is a type mismatch for algorithm and input provided.
		If <i>certDuration</i> is not an acceptable integer or not specified.
	Invalid or missing data size	If the specified size is: <ul style="list-style-type: none"> <li>▪ More than 4096</li> <li>▪ Negative No</li> <li>▪ Zero</li> <li>▪ Non Integer</li> </ul>
401	Authentication failed	The request lacks valid authentication credentials.
		If user authentication for the request fails.
403	Insufficient permission	The authenticated user is neither the key owner nor belongs to a group with full permission to the key.

		<p>If the user trying to encrypt is not owner of key being used and also does belong to group with full permissions on the key.</p>
		<p>If the user trying to decrypt is not owner of key being used and also does belong to group with full permissions on the key.</p>
	Key is not deletable	The key in the request is not configured as deletable.
	Key is not exportable	If the key being used for encryption is not exportable.
		If the key being used for decryption is not exportable.
404	Server not available or unsupported operation	Server Certificate has expired.
	Not found	Custom attributes are not found.
	Invalid Certificate to sign	Certificate signing request is corrupted.
409	Key already exists	Duplicate key name. The key name provided in the request already exists in ESKM.
412	Key not found	The requested key is not present in the ESKM.
		If the key being used for encryption does not exist in ESKM.
		If the key being used for decryption does not exist in ESKM
	Key is not exportable	The key is not configured as exportable.

Invalid or missing key name	The "keyName" is not specified in the request body or the key name is invalid.
Invalid or missing algorithm	The "algorithm" is not specified in the request body.
Invalid or missing key bytes	The "keyBytes" are not specified in the request body or the key bytes are invalid.
Invalid Certificate usage	The CertUsage is other than Server, Client, ServerClient.
Missing or invalid CA name	The entered CA is not present in the ESKM.
Missing certificate usage	when certificate usage is not specified.
Missing certificate signing request	when certificate signing request is not specified.
Unknown algorithm	The algorithm in the request is not a valid one.
Invalid or missing key owner	The "keyOwner" in the request body does not exist in ESKM.
Invalid or missing key length	The key length has a non integer value.
Unsupported key size	The key length in the request is not supported for the algorithm in the request.
User does not exist	The "keyOwner" in the request body does not exist in ESKM



Global keys are not supported	If the key being used for encryption is a global key.
Too few parameters	If the number of parameters in the request is less than the mandatory parameters.
Too many parameters	If the number of parameters in request body is greater than the total number of mandatory and optional parameters.
Invalid parameters	If any other parameter other than algorithm or input is specified in request.
Input is not base64 encoded	If input is not base64 encoded.
Input is too long	If the input for encryption exceeds maximum set limit.
Invalid key name	The "keyName" is not specified in the request body or the key name is invalid.
Invalid algorithm	The "algorithm" is not specified in the request body.
Invalid key owner	The "keyOwner" in the request body does not exist in ESKM.
Missing signature	If the signature parameter is not specified.
Signature is too long	If the signature for verification exceeds maximum set limit.
Signature is not base64 encoded	If signature is not base64 encoded
Unsupported algorithm or padding	If the algorithm or padding specified is not supported.

	Unsupported key for operation	If key's algorithm is not supported.
	Unsupported padding	If the padding specified is not supported.
	Current security settings do not allow global key usage	The appliance is FIPS compliant and the information of a global key is requested.
	Current security settings do not allow the key size	The appliance is FIPS compliant and the information of a key with unsupported algorithm is requested.
	Group does not exist	If an invalid group name is specified in the request body.
	Missing Group Name	If "Group Name" is not specified in the request body.
500	Failed to retrieve the key	Failed to retrieve the information of the requested key.
	Failed to retrieve the public key	Failed to retrieve the information of the requested key.
	Invalid token supplied	Request URL is invalid.
	Maximum key capacity has been reached.	The number of keys has reached the limit.
	Could not generate key	Failed to create the key.
	Could not import key	Could not import the key.
	Unsupported algorithm or mode	If the algorithm specified is not supported.
	Unknown algorithm	If algorithm is empty string.

Failed to encrypt	If encryption has failed.
Failed to decrypt	If decryption has failed.
Failed to verify	If the verification fails.
Failed to generate private RSA	Failed to generate private RSA from the specified key.
Failed to generate signature	Failed to generate the signature.
Failed to generate public RSA	Failed to generate public RSA from the specified key.
Key and crypto operations are disabled in REST server configuration	Key and crypto operations are not enabled in REST server configuration.