Fortanix®

User Guide

FORTANIX DATA SECURITY MANAGER – SECURITY CONTROLS AND KEY LIFECYCLE MANAGEMENT

VERSION 3.3

Fortanix Inc. | 3910 Freedom Circle | Suite 104 | Santa Clara, CA 95052 | United States of America | ☎ +1 (650) 943-2484 |
✉ info@fortanix.com | 🌐 www.fortanix.com

TABLE OF CONTENTS

1.0 INTRODUCTION 3

2.0 DEFINITIONS 4

3.0 SECURITY OBJECT LIFECYCLE MANAGEMENT 6

3.1 SECURITY OBJECT TYPES 6

3.1.1 Import Security Objects 6

3.1.2 Generate Security Objects 10

3.2 Security Object Operations 12

3.3 Security Object States Descriptions 12

3.4 Security Object Activation Date 14

3.5 Security Object Deactivation Date 14

3.6 Enabling a Deactivated Key 15

3.7 Security Object Attributes/Tags 16

3.8 Key Rotation 17

3.8.1 Generate new key 17

3.8.2 Rotate to an Existing Key 19

3.8.3 Scheduling Key Rotation 20

4.0 DESTROY AND DELETE SECURITY OBJECTS 21

4.1 Destroy and Delete Security Objects 22

5.0 PROGRAMMATIC MANAGEMENT OF SECURITY OBJECT STATES 22

5.1 Creating a Security Object in Pre-active State 22

5.2 Activating a Security object in Pre-active State 24

5.3 Creating/Updating a Security Object with a Deactivation Date 25

5.4 Deactivating a Security Object in Pre-active or Active State 26

5.5 Transitioning a Security Object to Compromised State 27

6.0 SECURITY CONTROLS FOR APPS THAT USE SECURITY OBJECTS 29
6.1 App Permission List 29
6.2 Setting App Permissions 30
6.3 Example of App Permissions 31
6.4 Key Derivation 31
6.5 Key Transformation 32
6.6 Key Wrapping / Unwrapping 32
6.7 Application Usage Reporting in Fortanix DSM SaaS 32
6.8 Enabling Audit Log Permission for Applications 33
6.9 Quorum Approval 33
6.10 Quorum Policy 34
6.11 Quorum Approval Status 34
6.12 Usage Example 35
6.13 Legacy Applications 43
7.0 DOCUMENT INFORMATION 45
7.1 Document Location 45
7.2 Document Updates 45

1.0 INTRODUCTION

Welcome to the Fortanix Data Security Manager (DSM) User Guide for Security Controls of Application (apps) and Security Objects. This document describes the Fortanix DSM Security Object Lifecycle Management. It also contains the information related to:

- Security controls around the security object such as:
  - Key rotation
  - Deactivation
  - Disabling and enabling keys.
- Security controls for apps on using a security object such as:
  - Quorum control
  - App permissions

2.0 DEFINITIONS

- **Fortanix Data Security Manager** -

Fortanix DSM is the cloud solution secured with Intel® SGX. With Fortanix DSM, you can securely generate, store, and use cryptographic keys and certificates, as well as secrets, such as passwords, API keys, tokens, or any blob of data.

- **Accounts** -

A Fortanix DSM account is the top-level container for security objects managed by the Fortanix DSM. An account is generally associated with an organization, rather than an individual. Security objects, groups, and applications belong to exactly one account. Different accounts are fully isolated from each other. *See support for more information.*

- **Users** -

Users are associated with an email address. A user can be a member of one or more accounts. Depending on permissions, users can:

- Perform management operations like adding or modifying users or groups
- Create security objects
- Change properties of security objects
- Review logs of Fortanix DSM activity

⚠ **Users cannot perform cryptographic operations. Only applications can perform cryptographic operations.**

- **Groups** -

A group is a collection of security objects created by and accessible by users and applications which belong to the group. The user who creates a group automatically gets assigned the role of the group administrator. You can add more users to the group in the role of administrators or auditors. You can also add applications to the group to enable the applications to create and use security objects in that group. *See support for more information.*

Access policies are set at the group level, so all security objects in a group share the same access policy. Any number of users and/or applications can be assigned to a group. *Some examples of usage of groups are given in the Authorization section*.

Quorum policies can also be set at group level. A Quorum policy mandates that all security sensitive operations in that group would require a quorum approval. Such operations include using a key for cryptographic operations or deleting or updating a group. *See Quorum Policy for more information.*

- **Applications** -

An application can use Fortanix DSM to generate, store, and use security objects, such as cryptographic keys, certificates, or an arbitrary secret. Applications can authenticate to Fortanix DSM using an API key (a secret token) or a TLS client certificate. An application can interact with Fortanix DSM using the REST APIs or using the PKCS#11, JCE, or CNG providers. *See support for more information.*

- **Fortanix Data Security Manager Security Objects** –

A security object is any datum stored in Fortanix DSM (for example a key, a certificate, a password, or other security objects). Each security object is assigned to exactly one group. users and applications assigned to the group have permission to see the security object and to perform operations on it. *See support for more information.*

3.0 SECURITY OBJECT LIFECYCLE MANAGEMENT

Security objects in Fortanix DSM follow a lifecycle which is composed of a set of states in which the object can be in. During the object's existence, the SO transitions from one state to another, according to policies or events set by the user.

3.1 SECURITY OBJECT TYPES

A security object can either be imported or generated in the Fortanix DSM UI.

3.1.1 IMPORT SECURITY OBJECTS

To import a security object:

1. In the Fortanix DSM UI, click the **Security Objects** tab and click the  button to add a new security object.
2. Enter a name for the security object and assign it to an existing group or create a new group. *Refer to the Fortanix DSM Getting Started guide to learn how to create a new group.*
3. Click **IMPORT** to import a security object. To import a security object from a component, *refer to the article Key Components*.
4. Choose a type of key to import.
   - All valid DSA key sizes are allowed during import.
   - The **"SECRET"** object can be used to store and export keys of any format. For easy identification, you can set any string to **Attribute** field while importing (optional). This field will be stored as an x-format custom attribute on a secret object and will be shown in the **Info** field when viewing the secret.
     You can also add these values from the detail view of a key in the **ATTRIBUTES/TAGS** tab.
   - **BIP32** is a parent key inside a Bitcoin hierarchical deterministic wallet (HD Wallet). This key can be used to recover all keys beneath it in the tree, for example: it can be used to sign transaction hashes and derive hardened and non-hardened children.
     The supported import/export formats are **Base64** or **Hex**.
     The supported parameters are Curve: **secp256k1**, Path: **m/0'/42/6147'**, Network: **Mainnet** or **Testnet**.

The supported key operations are:

- DERIVEKEY: Accepts an index input and creates a **hardened** child.
- EXPORT: Allows to export the secret key in BIP32 format.
- SIGN: Signs a transaction.
- VERIFY: Verifies a transaction.
- APPMANAGEABLE: Allows the key to be managed by a crypto application.
- TRANSFORM: Accepts an index input and creates a non-hardened child.

After the BIP32 key is imported, you can derive a hardened or non-hardened child using **DERIVE CHILD KEY** option. It expects the following inputs:

- **Index**: A 32-bit integer.
- Use the **Hardened child** option to create a hardened child. If this option is not selected, it will create a non-hardened child.

NOTE:

- For a non-hardened child key, the Index value must be between 0 to 2^31-1 (2147483647).
- For a hardened child key, the Index value must be 2^31 to 2^32-1 (2147483648 to 4294967295).
- If the non-hardened child keys are leaked, then even the parent key will be leaked.
- The BIP32 key can only be imported in Fortanix DSM. It cannot be generated. To generate a BIP32 key, use an HMAC seed and derive the BIP32 key using the Fortanix REST API.

WARNING: Before downgrading Fortanix DSM from 4.10 to any version <=4.9, remove all the BIP32 keys from the Security Objects table to avoid panic on the Security Objects page after the downgrade.

- **BLS:** Boneh–Lynn–Shacham is a cryptographic (digital) signature scheme allowing a user to verify that a signature over data is valid, with signature aggregation. Depending on the variant, the format of the private key used by Fortanix DSM is a big-endian secret scalar (32 bytes), followed by the serialized public key (uncompressed elliptic

curve point of 97 or 192 bytes).

The supported import key formats are **Base64** or **Hex**.

You can also upload a key file using the **Upload a File** option.

- Additionally, keys of type Korean Cryptography developed are supported in Fortanix DSM.
    - **SEED** is a cipher developed by the Korea Internet and Security Agency (KISA), offering 128-bit block sizes and 128-bit key sizes. The supported modes of operation are ECB, CBC, CBCNOPAD, and CTR.
    - **ARIA** also developed by KISA, is a cipher with 128-bit data blocks and with key sizes of 128, 192, or 256 bits. The supported cipher modes of operations are ECB, CBC, CBCNOPAD, CFB, CTR, GCM, and CCM.
    - **KCDSA** (Korean Certificate-based Digital Signature Algorithm) is a digital signature algorithm similar to DSA and additionally involves a hash algorithm used for signature verification. It offers key sizes between 512 and 2048 bits. Imported keys can use any parameters. Key generation will use the following parameters: 2048/224/SHA224 and 2048/256/SHA256 generated by KISA.
    - **EC-KCDSA** is the Elliptic Curve variant of KCDSA. It is also a digital signature algorithm similar to ECDSA and involves a hash algorithm used for signature verification. It supports the use of NIST curves and SHA algorithms. *For more details, refer to Algorithm Support*.
    - **SEED** is a cipher developed by the Korea Internet and Security Agency (KISA), offering 128-bit block sizes and 128-bit key sizes. The supported modes of operation are ECB, CBC, CBCNOPAD, and CTR.

5. Sometimes, keys of type AES, DES, DES3, DSA, or HMAC imported from a file are already wrapped (encrypted) by a key from Fortanix DSM. This is done so the key will not go over the TLS in plain text format. In such scenarios, select the check box **The key has been encrypted**.
6. To import an encrypted key in a file or as a blob provide the following details:
    a. Select the corresponding **Key Encryption Key** that was previously used to encrypt your target key. The selection will be used to unwrap (decrypt) the encrypted key in

the file and will later be stored securely in Fortanix DSM. This key should have already been created or imported in Fortanix DSM.

b. **Cipher mode:** Select the cipher mode of encryption that should be applied to the key material.

There are three types of encryption cipher modes to choose from:

i. **ECB:** In this method, plain text is divided into blocks of size 64bits each. Each such block is encrypted independently of other blocks. For all blocks, the same key is used for encryption.

ii. **KW:** This method uses symmetric encryption to encapsulate key material.

iii. **KWP:** In this method**,** additional padding of bits or bytes is appended to the encapsulated key material.

NOTE: A cipher mode of operation may not be available for selection based on the source and selected wrapping key combination.

7. **Key Check Value**: The KCV of the imported key is optionally added by the admin while creating the import request.
8. Click **UPLOAD A FILE** to upload the key file in **Raw**, **Base64**, or **Hex** format.

NOTE: For Secret security objects, you can additionally upload the key file in **Text (UTF-8)** format.

9. Select the permitted key operations. *For more information on key operations, refer to Key Operations* .
10. To add **Custom Attributes**, click the **ADD ATTRIBUTE** button. Custom attributes are user-defined attributes of security object that can be added to the key's metadata.
11. Add **Activation Date**: The activation date of the security object can be set to a specified time in the future or activated immediately. On activation, the object goes into the "Activated" state. *For more information on security object activation date, refer to Section 3.4.*
12. Add **Deactivation Date**: The deactivation date of the security object can be set to a specified time in the future, or you can expire an object immediately. *For more information on security object deactivation date refer to Section 3.5.*

NOTE: If a group-level Key metadata policy is set, you can configure certain restrictions on "metadata" associated with security objects. Here "metadata" refers to certain features of security objects that are not covered by cryptographic policies, for example, custom metadata, description, expiry, and so on. *For more information about key metadata policy, refer to User's Guide: Key Metadata Policy.*

13. To store audit logs for the object in the group, enable the toggle for **Keep detailed log for the object**. The initial state of the toggle is based on the parent Crypto policy if any**.**
14. Click **IMPORT** and the key is successfully imported.

Fortanix DSM also provides an ability to view SOs of type "secret" in the UI. This enables Fortanix DSM to be used as a secret store. To view the secret object:

1. Go to the detailed view of a "secret" object that was imported.
2. In the **INFO** tab, click the **SHOW** icon to show the value of the secret object.
3. The "secret" **Object Value** will be displayed.

   NOTE: If a quorum policy is set in the group to which the "secret" object belongs, then the object value will not be displayed in the UI and an approval request is created when you click **SHOW VALUE** to view the secret.

3.1.2 GENERATE SECURITY OBJECTS

To generate a security object:

1. In the Fortanix DSM UI, click the **Security Objects** tab and click  button to add a new security object.
2. Enter a name for the security object and assign it to an existing group or create a new group. *Refer to the Fortanix DSM Getting Started guide to learn how to create a new group.*
3. Click **GENERATE** to generate a security object.
4. Choose a type of key to generate the key. *For more information on the key types, refer to the previous section.*

   Additionally, the following keys are supported in Fortanix DSM.

- **LMS:** LMS is a hash-based digital signature algorithm**.** It manages the cryptographic keys within a cryptosystem. It deals with generating, exchanging, storing, using, and replacing keys as and when required at the user level. *For more information, refer to RFC 8554*.
    - L1 Tree height: This indicates the height of the top-level tree.
    - L2 Tree height: This indicates the height of the secondary level tree.

  To know more about the Fortanix implementation of LMS keys, refer to LMS Keys - FAQs.

  NOTE: Ensure that the sum of the L1 tree height and L2 tree height must not exceed 20.
- **DSA:** The Subgroup Size is the size of the parameter in 'q' bits, where 'q' is the parameter used in the generation of the DSA security object.
- **BLS**: Boneh–Lynn–Shacham is a cryptographic (digital) signature scheme allows a user to verify that a signature over data is valid, with signature aggregation. Fortanix DSM supports the following two variants:
    - **Minimal signature size:** Public keys are raw serialized points of G_2 (192 bytes), signatures are raw serialized points of G_1 (97 bytes - SEC1 sec. C.4).
    - **Minimal public key size:** Public keys are raw serialized points of G_1 (97 bytes), signatures are raw serialized points of G_2 (192 bytes).

  The format of the private keys used by Fortanix DSM are serialized as the big-endian secret scalar (32 bytes), followed by the serialized public key (97 or 192 bytes depending on the variant).

5. Select the permitted key operations.
6. Add custom attributes by clicking the **ADD ATTRIBUTE** button.
7. By default, the security object is set to be activated immediately. To specify activation date to a future time-period, click **EDIT.**
8. By default, the deactivation date of the security object is set to 'Never'. To specify the deactivation date to a future time-period, click **EDIT**.

   NOTE: If a group-level Key metadata policy is set, you can configure certain restrictions on "metadata" associated with security objects. Here "metadata" refers to

certain features of security objects that are not covered by cryptographic policies, for example, custom metadata, description, expiry, and so on. *For more information about key metadata policy, refer to User's Guide: Key Metadata Policy.*

9. To store audit logs for the object in the group, enable the toggle for **Keep detailed log for the object**.
10. Click **GENERATE** to generate the key. Afterwards, the key is successfully generated.

3.2 SECURITY OBJECT OPERATIONS

The SO operations can be classified into the following two groups:

- **Process Operations**: Verify, Decrypt, UnwrapKey, MacVerify.
- **Protect Operations**: Sign, Encrypt, Wrapkey, Derivekey, MacGenerate, AgreeKey.

NOTE: The set of allowed operations that a SO can perform will depend on the initial SO operations set by the user at creation time and the current state in which the SO is at.

3.3 SECURITY OBJECT STATES DESCRIPTIONS

- **Pre-active:** When the "`activation_date`" field of a SO is set to a future date, then the SO's initial state will be "`PreActive`" state. When a SO is in "`PreActive`" state, no cryptographic operation can be performed with the SO.
- **"Pre-active" to "active" transition:** When the SO reaches the "`activation_date`", it will automatically transition to "`Active`" state. Note that once the SO is active it cannot transition back to "`PreActive`" state.
  SOs created using the Fortanix DSM will set the "`activation_date`" to the current server time so they will automatically be initialized in "`Active`" state.
- **Active**: When a SO is in "`Active`" state it can be used for all processes and perform cryptographic operations set by the user.
- **Enabled / Disabled sub-states:** While an SO is in "`Active`" state, the user can "Enable/Disable" the SO. When the SO is in "`Enabled`" state, it can perform all the cryptographic operations set by the user. When the SO is in "`Disabled`" state the SO cannot perform any cryptographic operation.

The transition between "`Enabled`" and "`Disabled`" states is reversable. For any SO, a user can make the transition between these states an arbitrary number of times. This operation can be performed by updating the "`Enabled`" field of the SO to "`true`" (Enabled) or "`false`" (Disabled).

Active to deactivated transition: A security object transitions from "`Active`" to "`Deactivated`" state for the following reasons:

- When the "`deactivation_date`" is reached, the SO will automatically transition to "`Deactivated`" state. Note that once the SO is deactivated it cannot transition back to "`Active`" state.
- The client calls the Revoke API (*see **Fortanix Data Security Manager API reference***) for the SO. If the Revoke API is called, the server will automatically set the "`deactivation_date`" to the current server time and the SO will transition to "`Deactivated`" state. The Revoke operation may also be executed from the Fortanix DSM UI.

When a security object transitions to the "`Deactivated`" state it will also be set to "`Disabled`" sub-state. A user may change the SO sub-state to "`Enabled`" (*see the "**Deactivated**" section*).

- **Deactivated:** When an SO is in "`Deactivated`" state, it can perform certain cryptographic operations depending on its "`Enabled/Disabled`" sub-state:
  - **Enabled:** The SO can perform "Process Operations" only.
  - **Disabled:** The SO cannot perform any cryptographic operation.
- **Compromised transition:** A security object transitions to "`Compromised`" state when the client calls the Revoke API (*see **Fortanix Data Security Manager API reference***) for the SO and the "`reason`" field in the request body is set to "`COMPROMISED`" (*see **Fortanix Data Security Manager API reference***). The client can optionally set a date for the "compromised date" and if it is not provided, the server will set the time to the time when the request is received. Additionally, a user may provide a "revocation reason" string for audit purposes.

  A SO can directly transition to "`Compromised`" state from any other state. The "`Compromised`" state is final and cannot transition to any other state.

To mark a security object as "Compromised" from the Fortanix DSM GUI:

1. Go to the detailed view of the security object, and in the **INFO** tab scroll to the bottom and click the **DEACTIVATE NOW** button.

- **Compromised**: When an SO is in "`Compromised`" state it can perform certain cryptographic operations depending on its "Enabled/Disabled" sub-state:

  - **Enabled:** The SO can perform "Process Operations" only.
  - **Disabled:** The SO cannot perform any cryptographic operation.

  The transition between "`Enabled`" and "`Disabled`" states is reversible. For any SO, a user can make the transition between these states an arbitrary number of times. This operation can be performed by updating the "`enabled`" field of the SO to "`true`" (Enabled) or "`false`" (Disabled).

3.4 SECURITY OBJECT ACTIVATION DATE

When a SO is created, its initial state can be either: "`PreActive`" or "`Active`" depending on the value of the "`activation_date`" field provided in the SO creation request. The possible cases are:

- "`activation_date`" is not provided: The activation date will be set by the server to the current time. The initial state of the SO will be "`Active`".
- "`activation_date`" is provided: The activation date will be set to the one provided by the user. If the "`activation_date`" is in the future, the SO's initial state will be "`PreActive`" while if it is in the past the initial state will be "`Active`".

 **NOTE**: After the SO is created, the "`activation_date`" of the SO can be modified as long as the SO is still in "`PreActive`" state. Once the SO transitions to "`Active`" state the "`activation_date`" can no longer be modified.

3.5 SECURITY OBJECT DEACTIVATION DATE

An SO also contains a "`deactivation_date`" field that determines when the SO will transition from "`Active`" to "`Deactivated`" state. If:

- The "`deactivation_date`" field is not set; the SO will remain in "`Active`" state.
- The "`deactivation_date`" field is set, the SO will transition to "`Deactivated`" state upon that date.

An SO may also transition to "`Deactivated`" state for other type of events explained in sub-section "***Active to Deactivated Transition***" state under the section "***Security Objects State Descriptions***".

NOTE: The "`deactivation_date`" field can be modified by updating this field as long as the SO is still in "`Active`" or "`PreActive`" state. The server will reject any date that is before the "`activation_date`". If the client sets the "`deactivation_date`" to a time in the past the SO will transition to "`Deactivated`" state.

3.6 ENABLING A DEACTIVATED KEY

A deactivated key is disabled by default. You can enable a deactivated key by setting the "`mark_key_disable_when_deactivated`" to `false` as shown below:

```
#if the user sets the value as 'true', then the key will be disabled
after deactivation.
"mark_key_disable_when_deactivated": true

#if the user set the value as 'false', then key is enabled after
deactivation.
"mark_key_disable_when_deactivated": false
```

The following is the sample `PATCH` API request.

```
curl -X PATCH -H "Authorization: Bearer
60_Y3sjMP9s8__nnCDKI4G8K395Nlfgtb0pLUkL50H2yFauTk5WgrN25jMgAUNfNdVzLwf
VtEn_xd9qzYHeWEA" -H "Content-Type:application/json" -d
'{"mark_key_disable_when_deactivated":false}' -k
'https://<cluster_ip>/api/sys/v1/accounts/dcd6fd3a-125c-46b2-946d-
f9063caad5f6'
```

Where, the `<cluster_ip>` is the IP address of the cluster.

Response:


```
{
  "acct_id": "dcd6fd3a-125c-46b2-946d-f9063caad5f6",
  "auth_config": {
    "password": {
      "require_2fa": false,
      "administrators_only": false
    }
  },
  "created_at": "20230321T200950Z",
  "enabled": true,
  "logging_configs": {


  },
  "mark_key_disable_when_deactivated": false,
  "max_operation": 10000,
  "name": "Navendu",
  "state": "PendingConfirmation",
  "subscription": {
    "trial": {
      "expires_at": null
    }
  }
```


3.7 SECURITY OBJECT ATTRIBUTES/TAGS

Every security object contains attributes such as PKCS #11, CNG, and Custom attributes.

- PKCS #11 and CNG attributes – These are standard attributes for their corresponding specifications. For details see the PKCS #11/CNG specification. The SO attribute values are mapped to the corresponding name in PKCS #11/CNG.
- Custom attributes – These are user defined SO attributes that can be added to the key's metadata.

3.8 KEY ROTATION

A security object can be rotated using the Fortanix DSM Key Rotation feature. A Key is rotated when you want to retire an encryption key and replace that old key by generating a new cryptographic key. Based on the business requirements, a key can be rotated at any time. Fortanix DSM allows you to rotate the key in either of the following ways:

3.8.1 GENERATE NEW KEY

Perform the following steps to rotate a key when the **Generate new key** option is selected:

1. In the Fortanix DSM UI, go to the detailed view of a security object.
2. Under the name of the security object, click the **ROTATE KEY** link.
3. To deactivate the original key after the key is rotated, select the **Deactivate original key after the rotation** check box so that the old key will only be able to perform operations such as decrypt/verify/unwrap/mac verify. This can also be set in the "Key rotation policy". *For more details, refer to Section 3.7.3.*

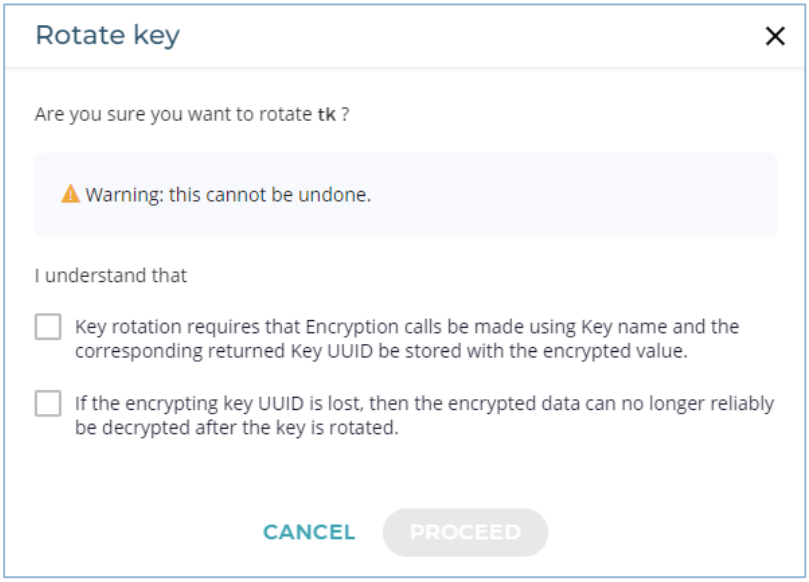
   NOTE:

   - The **Deactivate original key after the rotation** check box is not applicable when you are rotating keys of type "Secret".
   - This check box is disabled If the key belongs to a group with a quorum policy.
4. Select the **Generate new key** option.

   To rotate an existing key of type "Secret" and keep a history of all previous versions of the same secret:

   a. In the KEY ROTATION window, enter a new object value for the secret in **Text (UTF-8)**, **Raw**, **Base64**, or **Hex** format. This step is mandatory.
5. To set the deactivation date of the newly rotated key to a future date, click **EDIT.**
6. Click **ROTATE KEY** in the Key Rotation window to confirm the key rotation.

A cautionary message will appear on the screen, ensuring that you are fully aware of the implications of key rotation. You must select both the check boxes to enable the **PROCEED** button.



If the key belongs to a group with a quorum policy, the key rotation will require quorum approval. Also, if any of the linked keys is a part of a group with a quorum policy, instead of directly rotating the key, a quorum approval request will be generated when you click on **ROTATE KEY.**

7. The key will be successfully rotated. The new key/secret will carry over the same name as the old key/secret whereas the old key/secret will be renamed with the timestamp of key rotation. A new key with a different UUID will be added to the Security Objects table. The old keys/secrets will also show under the **KEY LINKS** tab in the detailed view of the new rotated key/secret.
8. Notice the new UUID of the rotated key.

NOTE:

- The OPAQUE and CERTIFICATE key types cannot be rotated.
- If the cluster is unhealthy when the key is rotated, then the key rotation job will be retried every 10 mins until the rotation is successful.
- For the rotation of secret keys, the key permissions are not editable.

3.8.2 ROTATE TO AN EXISTING KEY

Prerequisites for rotating an old DSM key to a new DSM key:

- The key type, size, and permissions of both keys must be identical for the rotation to happen.
- Both the keys involved in the rotation must belong to the same group.
- Both the keys involved in the rotation must have the same padding or signature policy, key rotation policy, and key access justification policy.

Perform the following steps to rotate an old DSM key to a new DSM key when the **Rotate to an existing key** option is selected:

1. In the Fortanix DSM UI, go to the detailed view of a security object.
2. Under the name of the security object, click the **ROTATE KEY** link.
3. Select **Rotate to existing key** option.
4. Select the existing key that you want to rotate it with from the **Select existing key** drop down menu.

   When you rotate an old DSM key (Key A) to a new DSM key (Key B).

   - The new DSM key (Key B) will get the original name (that is, Key A), and the old DSM key (Key A) will be renamed automatically with the timestamp of key rotation.
   - The UUID of the new DSM key will remain the same as before (that is, the same UUID of Key B as before).
   - The key material of the new DSM key will remain the same as before (that is, the same key material of Key B as before).
   - The new DSM key will link to the old DSM key so that the old DSM key can still be used for decryption operations.
   - The old DSM key can be deleted, disabled, or deactivated after successful rotation.
   - The **Deactivate original key after rotation** check box, **Deactivation Date** section, and **EDIT** permissions section will be disabled for this type of rotation.
   - This feature does not apply to keys that are externally backed.
5. Click **ROTATE KEY** in the Key Rotation window to confirm the key rotation.

If the key belongs to a group with a quorum policy, the key rotation will require quorum approval. Also, if any of the linked keys is a part of a group with a quorum policy, instead of directly rotating the key, a quorum approval request will be generated when you click on **ROTATE KEY.**

NOTE: The "Rotate to an existing key" option is not supported for tokenization keys.

6. The old DSM key will be successfully rotated. The old keys/secrets will also show under the **KEY LINKS** tab in the detailed view of the new rotated key/secret.

NOTE:

- If the old DSM key has metadata or policies already configured, but the new DSM key does not have metadata or policies, then the metadata and policies will be copied from the old DSM key to the new DSM key.
- If the old DSM key does not have metadata or policies and the new DSM key has metadata or policies, then the rotation will be disallowed.
- If neither old DSM key nor new DSM key has metadata or policies, then the rotation will be allowed.
- If both old DSM key and new DSM key have metadata or policies but they are different, then the rotation will be disallowed.

For the rotation of secret keys, the automatic key rotation policies to schedule key rotation described in the next section are not applicable.

3.8.3 SCHEDULING KEY ROTATION

Fortanix DSM also allows key rotation to be scheduled for a future time to be done automatically by setting a key rotation policy. If the key rotation policy is set, the key will be periodically rotated as a measure of extra protection for the key material. The scheduled key rotation can be applied at a security object level by setting a time period for automatic key rotation. The time will have the following components.

- Start date of the schedule (for example, start key rotation on **06/08/2020**)
- Frequency of rotation (rotate the key every **10 days/weeks/months/years**)

- Time of rotation (rotate the key every 10 days starting 06/08/2020 at **12.00 pm**)

To schedule a key rotation policy for a security object:

1. Go to the detailed view of a security object in the Fortanix DSM UI.
2. In the detailed view, click the **KEY ROTATION** tab and click the **ADD POLICY** button.
3. Enter the key rotation schedule by specifying the rotation frequency, start date, and time.
4. To deactivate the old key after key rotation, select the **Deactivate original key after the rotation** check box.
5. Click **SAVE POLICY** to save the policy. To edit the policy in the future, click the **EDIT POLICY** button.

   TIP:

   - If the number of days in the frequency of rotation is greater than 28 days, use the "rotate by month" option.
   - If the number of weeks in the frequency of rotation is greater than 4 weeks, use the "rotate by month" option.
   - If the number of months in the frequency of rotation is greater than 12 months, use the "rotate by year" option.

4.0 DESTROY AND DELETE SECURITY OBJECTS

A security object may be destroyed or deleted using the Destroy and Delete button in the detailed view of the key.

- **Destroy** - When you destroy a key, the key loses its key material, however, it retains the key metadata. The data encrypted with the key can no longer be used once the key is destroyed. The key goes to a "disabled" state which is irreversible.
- **Delete** – When you delete a key, the key is permanently deleted along with its metadata. This is an irreversible operation.

  For more details about the Destroy and Delete behaviour using the Key Undo Policy, refer to User's Guide: Key Undo Policy.

4.1 DESTROY AND DELETE SECURITY OBJECTS

If you destroy a Fortanix DSM key without Key undo policy, then the key deletion is a 1-step process with a confirmation.

1. Go to the detailed view of the security object and click the **DESTROY KEY** button.
2. In the DESTROY KEY confirmation window, click the check box(es) which are a warning that a user should read and select before destroying the security object. Once this check box(es) is selected, it will enable the **DESTROY** button.

   NOTE: If the Security Object has quorum approval policy set, then an approval request will be initiated once you click the **DESTROY** button in the window below.
3. Click **DESTROY** to enter the "destroyed" state. The user also has an option to "Cancel" the Key Destroy operation using the **CANCEL** button.
4. You could also start the key destroy process for a key from the SO table view. Select the security object and click the **DESTROY SELECTED** button.

   Hover on the key to see that the key is in "Destroyed" state. Notice that the colour of the destroyed key icon is black ■ to indicate that the key is destroyed and ready to be deleted.
5. To delete the key metadata, click the **DELETE KEY** button to permanently delete the key metadata. If the Key undo policy is added, then the key delete operation is reversible until the specified time period. *For more details, refer to the User's Guide: Key Undo Policy*.
6. In the **DELETE SECURITY OBJECT** window, select the check boxes to confirm that you do not need the key metadata anymore and want to delete the key permanently. Once the check boxes are selected it will enable the **PROCEED** button.
7. Click **PROCEED** to confirm the key delete operation.

5.0 PROGRAMMATIC MANAGEMENT OF SECURITY OBJECT STATES

5.1 CREATING A SECURITY OBJECT IN PRE-ACTIVE STATE

By default, the Fortanix DSM UI creates a SO in "`Active`" state. Users can create SOs in "`PreActive`" state using the Fortanix DSM REST API by following either of the below methods:

1. Setting the "`activation_date`" to a time in the future. For example:

```
POST Error! Hyperlink reference not valid.
```

Request body:

```
{
"name": "mysymmkey101",
"description": "This is symmetric key 1",
"activation_date": "20191207T023624Z",
"key_size": 256,
"obj_type": "AES",
"enabled": true,
"custom_metadata": { "name-1": "value-1", "name-2": "value-2"},
"attributes": {"app_manageable": true},
"group_id": "{{grp1_id}}"
}
```

2. Explicitly setting the initial "`state`" of the key to "`PreAactive`". For example:

```
POST https://{{server}}/crypto/v1/keys
```

Request body:

```
{
"name": "mysymmkey102",
"description": "This is symmetric key 1",
"key_size": 256,
"obj_type": "AES",
"enabled": true,
"custom_metadata": { "name-1": "value-1", "name-2": "value-2"},
"attributes": {"app_manageable": true},
```

```
"state": "PreActive",
"group_id": "{{grp1_id}}"
}
```

In this scenario the SO will not transition to "Active" state until the user explicitly calls the "activate" API endpoint for the key, or the SO is updated with an "activation_date" that becomes current.

NOTE: Users can provide both a "state" and "activation_date" in the same request but if these two parameters are incoherent (that is, "state" being "Active" but an "activation_date" is in the future, then the backend will return a "400 - Bad Request" error code, and the key will not be created)

5.2 ACTIVATING A SECURITY OBJECT IN PRE-ACTIVE STATE

A SO can be explicitly activated using the Fortanix DSM API by:

- Using the following command:

```
POST https://{{server}}/crypto/v1/keys/{{key_id}}/activate
```

The "activation_date" of the SO will be set to the time at which the backend received the request.

If the SO is not in "PreActive" state, the API request will fail with "Error 400 – Bad Request".

- Patching the "activation_date" of the SO to the current or past time. For example:

```
PATCH https://{{server}}/crypto/v1/keys/{{key_id}}
```

Request body:

```
{
"activation_date": "22190411T004313Z"
}
```

5.3 CREATING/UPDATING A SECURITY OBJECT WITH A DEACTIVATION DATE

- Users can set the "`deactivation_date`" of a SO when it is created. For example:

```
POST Error! Hyperlink reference not valid.
```

Request body:

```
{
"name": "mysymmkey101",
"description": "This is symmetric key 1",
"deactivation_date": "20191207T023624Z",
"key_size": 256,
"obj_type": "AES",
"enabled": true,
"custom_metadata": { "name-1": "value-1", "name-2": "value-2"},
"attributes": {"app_manageable": true},
"group_id": "{{grp1_id}}"
}
```

NOTE: If the "`deactivation_date`" has already passed the API request will fail with "Error 400 – Bad Request".

- Users can also update a key and set/unset/modify the "`deactivation_date`" by sending a PATCH request. For example:

```
PATCH https://{{server}}/crypto/v1/keys/{{key_id}}
```

Request body:

```
{
"deactivation_date": "22190411T004313Z"
}
```

5.4 DEACTIVATING A SECURITY OBJECT IN PRE-ACTIVE OR ACTIVE STATE

An SO can be explicitly deactivated using the Fortanix DSM API by:

- Making a "`revoke`" request. For example:

```
POST https://{{server}}/crypto/v1/keys/{{key_id}}/revoke
```

Request body:

```
{
"code": "Unspecified",
"message": "Lost it in the bus"
}
```

The request body contains the following fields:

- `Code`: Revocation reason. It can be one of the following: `Unspecified`, `AffiliationChange`, `Superseded`, `CessationOfOperation`, `PrivilegeWithdrawn`.
- `message` (optional): A string for specifying a note on why this key is being deactivated.

If the SO is not in "`PreActive`" or "`Active`" state, the API request will fail with "Error 400 – Bad Request".

When the key is "revoked" the SO will contain a "`revocation_reason`" field as shown below:

```
"revocation_reason": {
            "code": "Unspecified",
            "message": "Lost it in the bus",
            "compromise_occurance_date": null
        }
```

- Patching the "`activation_date`" of the SO to the current or past time. For example:

```
PATCH https://{{server}}/crypto/v1/keys/{{key_id}}
```

Request body:

```
{
"deactivation_date": "22190411T004313Z"
}
```

5.5 TRANSITIONING A SECURITY OBJECT TO COMPROMISED STATE

An SO can be explicitly updated to "`compromised`" using the Fortanix DSM API by making a "`revoke`" request. For example:

```
POST https://{{server}}/crypto/v1/keys/{{key_id}}/revoke
```

Request body:

```
{
"code": "KeyCompromise",
"message": "Lost it in the bus",
"compromise_ocurance_date": "22190411T004313Z"
}
```

The request body contains the following fields:

- `code`: Revocation reason. It can be one of the following: `KeyCompromise`, `CACompromise`.
- `message` (optional): A string for specifying a note on why this key is being deactivated.
- `compromise_occurance_date` (optional): Date in which it was detected that the SO was compromised.

If the SO is already in "`Compromised`" state, the API request will fail with "Error 400 – Bad Request".

When the SO is in "`Compromised`" state it will contain a "`revocation_reason`" field as show below:

```
"revocation_reason": {
"code": "KeyCompromise",
"message": "Lost it in the bus",
"compromise_occurance_date": "22190411T004313Z"
 }
```

 **NOTE**: In all the API calls described in the previous sections, the backend will always ensure that the "`activation_date`" is always before the "`deactivation_date`" and "`compromised_occurance_date`". Any request that tries to violate this condition will fail with "Error 400 – Bad Request".

To mark a security object as "Compromised" from the Fortanix DSM GUI:

1. Go to the detailed view of the security object, and in the **INFO** tab scroll to the bottom and click the **DEACTIVATE NOW** button.

6.0 SECURITY CONTROLS FOR APPS THAT USE SECURITY OBJECTS

Since version 3.6, Fortanix DSM provides fine-grained control that define what operations an application (app) can perform within each group the app belongs to. In addition to the “Key Operations” that can be given to a security object, users can assign operations to a group that an App can perform.

An app using a security object has the following security controls:

- Quorum control
- App permissions

6.1 APP PERMISSION LIST

App permissions resemble the "**Key Operations**" that can be allocated to security objects. App permissions may be added after App creation.

The following is a list of possible App permissions:

- Encrypt
- Decrypt
- WrapKey
- UnwrapKey
- DeriveKey
- Transform
- MacGenerate
- MacVerify
- Manage
- Sign
- Verify
- AgreeKey
- Export

The operations that an App is allowed to perform include the "**Manage**" permission operation. When this operation is enabled, it enables the app to carry out the following operations on a security object:

- **Create**: Allows an app to generate or import a key using `/crypto/v1/keys` POST/PUT API.
- **Copy**: Allows an app to copy a key from one group to another using `/copy` API.
- **Rotate**: Allows an app to rotate a key using `/rekey` API.
- **Activate**: Allows an app to transition key state from preactive to active state using `/activate` API.
- **Revoke**: Allows an app to deactivate or compromise a key using `/revoke` API.
- **Revert**: Allows an app to revert a key to its previous state using `/revert` API.
- **Move**: Allows an app to move a key from one group to another using `/crypto/v1/keys` PATCH API.
- **Update Profile:** Allows an app to update the security object's name, description, custom metadata, links, and publish public key information.
- **Update enabled state:** Allows an app to enable or disable a security object.
- **Update policies:** Allows an app to update Key Access Junctions (KAJ) and update the security object's size during `rekey` operations.
- **Update keyops:** Allows an app to update allowed key operations on the security object.
- **Delete key material:** Allows an app to delete the security objects by calling /`delete` API or through scheduled deletion.
- **Restore External So:** Allows an app to restore key material of an external security object. The external security object should have been created by copying an existing security object.
- **Calculate Digest:** Allows an app to utilize security object to compute the digest of any arbitrary data.
- **Destroy**: Allows an app to destroy the key using `/destroy` API.
- **Delete**: Allows an app to delete the key using `/crypto/v1/keys` DELETE API.

6.2 SETTING APP PERMISSIONS

During the app creation, a user may set the permissions for each group the app is added to.

1. Select the groups in which you want the app to be included.
2. After selecting the groups for the app, click the squares next to the group names. This opens a new window where you can restrict the app's group permissions.

   NOTE: By default, an app is assigned all possible permissions.

6.3 EXAMPLE OF APP PERMISSIONS

An app with permission to perform encryption, but not decryption, in **Group1**.

1. Create **Key1** in **Group1** with "**Encrypt**" and "**Decrypt**" permissions.
2. Now, create **App1** in **Group1** with "**Encrypt**" permission.
   With this setup, **App1** will be able to perform encrypt operations using **Key1**.
3. Create **App2** with "**Encrypt**" and "**Sign**" operations.
4. With this setup, **App2** will be able to perform encrypt operations using **Key1**. Even though **App2** has "**Sign**" permission on **Group1**, the sign operations using **Key1** will still fail because **Key1** has disabled the "**Sign"** key permission. All operations other than "**Encrypt**" and "**Sign**" performed by **App2** on **Group1** will be forbidden.

6.4 KEY DERIVATION

An app that performs a "**DeriveKey**" operation must have the following two permissions:

- "**DeriveKey**" permission on the group for the key that is used to derive another key.
- "**Manage**" permission in the group of the newly created key.

For example:

App1 wants to derive a key using **Key A** that belongs to **Group1** and wants the derived key to belong to **Group2**. Then, **App1** needs to have "**DeriveKey**" permission in **Group1** and "**Manage**" permission in **Group2**.

NOTE: For the above derive operation, **Key A** should also have the "**DeriveKey**" permission set so that **App1** can derive from it.

6.5 KEY TRANSFORMATION

An app that performs a "**Transform**" operation must have the following permission:

- "**Transform**" permission enabled at the group and app levels.

For example:

If **App1** wants to transform a key using **Key A** that belongs to **Group1,** then **App1** needs to have "**Transform**" permission in **Group1.**

NOTE: For the above transform operation, **Key A** should also have the "**Transform**" permission set (**Derive**, **Sign**, and **Verify**) so that **Key A** can derive from it.

6.6 KEY WRAPPING / UNWRAPPING

An App that wants to perform a "**Wrap**" operation must have:

- "**WrapKey**" permission on the group that has the wrapping key.
- "**Export**" permission on the group that has the wrapped key.

For example:

App1 wants to wrap **Key B** in **Group2** with **Key A** in **Group1**. Then, **App1** must have "**WrapKey**" permission in **Group1** and "**Export**" permission in **Group2**.

NOTE: For the above wrap operation, **Key A** should also have the "**WrapKey**" permission and **Key B** should have the "**Export**" permission for **App1** to use it for the wrap operation.

6.7 APPLICATION USAGE REPORTING IN FORTANIX DSM SAAS

You can track the usage of a Fortanix DSM SaaS application using the **USAGE** tab in the detailed view of the application. This tab lists all the crypto operations that the app performed in a particular month along with the date when the operation was performed. To access the **USAGE** tab:

1. Click the detailed view of a Fortanix DSM SaaS application.
2. Select the **USAGE** tab.

The app can perform the following cryptographic operations:

- Tokenization: Tokenize and detokenize operations, that is, Encrypt or Decrypt in format-preserving encryption (FPE) mode.
- Secret: Export operation.
- General: Rest of the operations, that is, Encrypt, Decrypt, WrapKey, UnwrapKey, Sign, Verify, MacGenerate, and others.

6.8 ENABLING AUDIT LOG PERMISSION FOR APPLICATIONS

You may allow applications to write to audit logs by using the configuration in the following UI screens:

- **Add application page**:
  a. When you create a new application and assign the application to a group, click the squares icon next to the group name.
  b. In the "Set app permissions for objects in the group" window, select the check box **Allow access to audit log** to enable audit log access for the application.
- **Application detailed view page**:
  a. To enable an application to access the audit log from the detailed view of an application, go to the application detailed view and click the **App permission** icon next to the group name.
- In the "Set app permissions for objects in the group" window, select the check box **Allow access to audit log** to enable audit log access for the application.

6.9 QUORUM APPROVAL

Quorum Approval is a Fortanix DSM feature that provides an additional layer of control and protection to the operations performed by, or done on, Security Objects and Plugins. As an example, by setting a "Quorum Approval" policy on a group, cryptographic operations such as "signing" require the approval of a certain number of members in the group before it can be performed.

"Quorum Approval" policy is set on a group basis, and it affects all the SOs that belong to that group.

6.10 QUORUM POLICY

A Quorum Policy is composed of one or more rules. The following are the options for a Quorum Policy rule:

- Quorum Group: A set of members in the group that are needed to approve an operation.
- Administrator: Minimum number of administrators that need to approve the operation.
- Application: an application that approves a sensitive operation for a specific use case.
- Using a second factor security key to approve the request.
- Password re-entry required to approve the request.

In addition, the Quorum Policy can establish whether "all" or "any" of the Quorum Policy rules are required to approve the requested operation.

6.11 QUORUM APPROVAL STATUS

The following diagram shows the states of a quorum approval request:

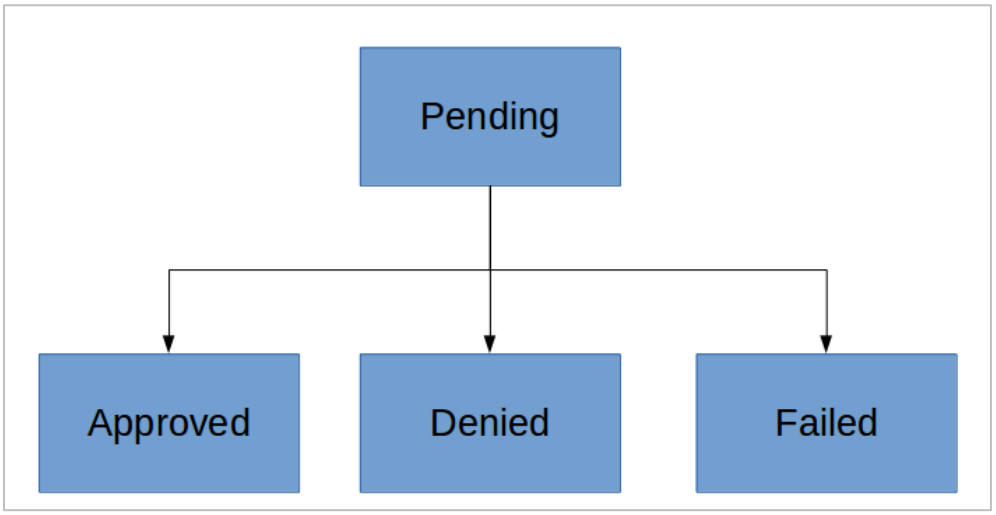


FIGURE 1: QUORUM APPROVAL REQUEST STATES

1. "Pending": This is the initial state of an approval request. This state means that the approval process has not yet matched Quorum Policy requirements.
2. "Approved": The approval request met the Quorum Policy requirements and provided the result of the request.

3. "Denied": A user of the Quorum Group has denied the operation. It takes only one user to deny the operation.
4. "Failed": The requested operation achieved quorum, but the operation requested fails. For example: the data provided for encryption has an invalid format.

 **NOTE**: The states "Approved", "Denied" and "Failed" are final states.

6.12 USAGE EXAMPLE

In the following example, a Group with a Quorum Policy is created. Afterwards, a quorum approval for data encryption is requested. When a quorum is achieved, the requester can perform the encryption operation.

In the following example "Quorum Group" is created with two Quorum Policy rules:

1. Rule 1:
    a. Members: Administrator 1 & Administrator 2
    b. Two administrators must approve
2. Rule 2:
    a. Members: Administrator 3 & Administrator 4
    b. One member must approve

The quorum policy is that **any** of the Policy rules is enough for acquiring quorum.

The group can also be created using the following API request:

```
POST https://{{server}}/sys/v1/groups
```

Request Body:

```
{
    "name": "Quorum Group",
    "description": "Group with quorum policy",
    "acct_id": "{{acct_id}}",
    "approval_policy": {
        "quorum": {
```

```
            "n": 1,
            "members": [
                {
                    "quorum": {
                        "n": 2,
                        "members": [
                            {
                                "user": "{{admin1}}"
                            },
                            {
                                "user": "{{admin2}}"
                            }
                        ],
                        "require_2fa": false,
                        "require_password": false
                    }
                },
                {
                    "quorum": {
                        "n": 1,
                        "members": [
                            {
                                "user": "{{admin3}}"
                            },
                            {
                                "user": "{{admin4}}"
                            }
                        ],
                        "require_2fa": false,
                        "require_password": false
                    }
                }
            ]
        }
    }
}
```

In the JSON object above, a Quorum Policy with multiple rules is mapped to a list of quorum objects. The parameter “n” specifies the number of users needed to obtain quorum. Using the API directly provides the following additional operations:

1. Specifying the exact number of quorum groups that are needed to achieve quorum for an operation.
2. Adding users in quorum groups that are group members. The Web UI only allows adding Group Administrators to quorum groups. It is recommended that administrators approve sensitive operations.

In the example, assume that an App will create the approval request to encrypt data, and a Security Object (SO) was created. please refer to the appropriate documents on how to create an SO and an App.

If the App sends a request to the backend to perform a crypto operation it will fail with a HTTP Error 403 – Forbidden and the message "This operation requires approval". For example:

```
POST https://{{server}}/crypto/v1/keys/{{symm_key_uuid}}/encrypt
```

Request body:

```
{
  "alg": "AES",
  "mode": "KW",
  "plain": "VGhpcyBpcyBteSBzZWNyZXQ="
}
```

The operations fails with error "403 - Forbidden" and message "This operation requires approval".

Approval requests are created by sending a `POST` request to `/sys/v1/approval_requests`.

An approval request body contains the following three fields:

1. `Operation`: URL path to the operation to be performed/approved.
2. `Method`: Corresponding HTTP method for the operation to be performed/approved.
3. `Body`: Object with the required parameters to perform the operation. This would be identical to the request body that would be provided to a request to the URL in the "`operation`" field.

For example, the below is a quorum approval request to encrypt plain text using an AES with Key Wrap (KW) cipher mode is:

```
POST https://{{server}}/sys/v1/approval_requests
```

Request body:

```
{
  "method":"POST",
  "operation":"/crypto/v1/keys/{{symm_key_uuid}}/encrypt",
  "body":{
    "alg": "AES",
    "mode": "KW",
    "plain": "VGhpcyBpcyBteSBzZWNyZXQ="
  }
}
```

Response:

```
{
    "acct_id": "b280ef77-0ba1-4880-8f34-3196ea12ba5e",
    "approvers": [],
    "body": {
        "alg": "AES",
        "mode": "KW",
        "plain": "VGhpcyBpcyBteSBzZWNyZXQ="
    },
    "created_at": "20191205T203648Z",
    "expiry": "20200104T203648Z",
    "method": "POST",
    "operation": "/crypto/v1/keys/d9cbf11b-a154-4aca-8598-
d6c569582b20/encrypt",
    "request_id": "e7a7cac9-d1d4-4082-b337-49e1483aa528",
    "requester": {
        "app": "17551f75-86bc-462c-a241-b63dea2449d7"
    },
    "reviewers": [
        {
```

```
            "user": "a8f31cfa-9b55-46eb-8c05-6961ebd62cc5"
        },
        {
            "user": "379cea5c-6c10-40c3-bc78-7fba347805f3"
        },
        {
            "user": "bb74bb5d-357b-47a8-9ba9-92b8f3e28b8f"
        },
        {
            "user": "dcfbd68b-4255-467d-886f-837c3cf5789d"
        }
    ],
    "status": "PENDING",
    "subjects": [
        {
            "sobject": "d9cbf11b-a154-4aca-8598-d6c569582b20"
        }
    ]
}
```


The response lists the UUIDs of all the users that are part of quorum groups. The status field is "`PENDING`" since the administrators need to approve the request.

After the approval request is created, users that are part of the quorum groups will be notified that there is a "pending approval". When the user logs into the account, the homepage of the Web UI will display a banner message.

Users can also click the "**Tasks**" tab in the "**Events**" window to review the list of "pending", "approved" and "archived" requests:

Users can also obtain a list of "approval requests" by sending an API request to the backend:

```
GET /sys/v1/approval_requests
```

Response:


```
[
```

```
    {
        "acct_id": "b280ef77-0ba1-4880-8f34-3196ea12ba5e",
        "approvers": [],
        "body": {
            "alg": "AES",
            "mode": "KW",
            "plain": "VGhpcyBpcyBteSBzZWNyZXQ="
        },
        "created_at": "20191205T203648Z",
        "expiry": "20200104T203648Z",
        "method": "POST",
        "operation": "/crypto/v1/keys/d9cbf11b-a154-4aca-8598-
d6c569582b20/encrypt",
        "request_id": "e7a7cac9-d1d4-4082-b337-49e1483aa528",
        "requester": {
            "app": "17551f75-86bc-462c-a241-b63dea2449d7"
        },
        "reviewers": [
            {
                "user": "379cea5c-6c10-40c3-bc78-7fba347805f3"
            },
            {
                "user": "dcfbd68b-4255-467d-886f-837c3cf5789d"
            },
            {
                "user": "bb74bb5d-357b-47a8-9ba9-92b8f3e28b8f"
            },
            {
                "user": "a8f31cfa-9b55-46eb-8c05-6961ebd62cc5"
            }
        ],
        "status": "PENDING",
        "subjects": [
            {
                "sobject": "d9cbf11b-a154-4aca-8598-d6c569582b20"
            }
        ]
    }
]
```


An App can get the result of this request by sending the request:

```
POST https://{{server}}/sys/v1/approval_requests/{{request_uuid}}/result
```

If the approval request is in "`pending`" state, the request will return HTTP error 400 - "Bad Request" and the message "request is pending".

The administrator can approve the request by clicking the "**Approve**" button.

Users can also approve or decline quorum requests with the following requests:

```
POST https://{{server}}/sys/v1/approval_requests/{{request_uuid}}/approve
POST https://{{server}}/sys/v1/approval_requests/{{request_uuid}}/deny
```

Once (administrator 1 or administrator 2) or (administrator 3 and administrator 4) approve the request, the request will transition to "APPROVED" state in the Web UI and the users that approved the quorum request are displayed.

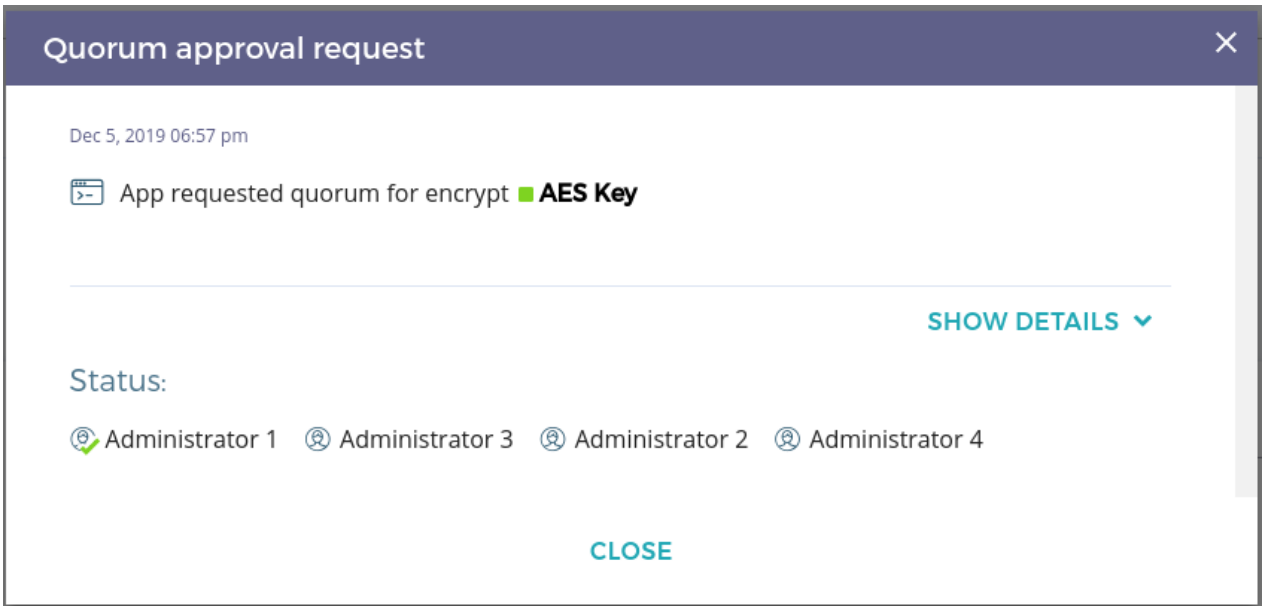


FIGURE 2: USERS WHO APPROVED QUORUM REQUESTS

Following is the request to get the approval request:

```
GET https://{{server}}/sys/v1/approval_requests
```

Response:


```
[
    {
        "acct_id": "b280ef77-0ba1-4880-8f34-3196ea12ba5e",
        "approvers": [
            {
                "user": "379cea5c-6c10-40c3-bc78-7fba347805f3"
            }
        ],
        "body": {
            "alg": "AES",
            "mode": "KW",
            "plain": "k7hjmmsHZGrEnJs0mu1y1f/Y/l93bs6jT1a0kS4SDRs="
        },
        "created_at": "20191206T025707Z",
        "expiry": "20200105T025732Z",
        "method": "POST",
        "operation": "/crypto/v1/keys/d9cbf11b-a154-4aca-8598-
d6c569582b20/encrypt",
        "request_id": "595e09a2-5987-4ad6-af92-56cdedba71e9",
        "requester": {
            "app": "17551f75-86bc-462c-a241-b63dea2449d7"
        },
        "reviewers": [
            {
                "user": "379cea5c-6c10-40c3-bc78-7fba347805f3"
            },
            {
                "user": "bb74bb5d-357b-47a8-9ba9-92b8f3e28b8f"
            },
            {
```

```
                "user": "dcfbd68b-4255-467d-886f-837c3cf5789d"
            },
            {
                "user": "a8f31cfa-9b55-46eb-8c05-6961ebd62cc5"
            }
        ],
        "status": "APPROVED",
        "subjects": [
            {
                "sobject": "d9cbf11b-a154-4aca-8598-d6c569582b20"
            }
        ]
    }
]
```

The status of the approval request is "APPROVED".

To get the result of the operation, the request is:

```
GET https://{{server}}/sys/v1/approval_requests/{{request_id}}/result
```

Response:

```
    {
      "status": 200,
      "body": {
        "cipher":
"vcybpJfTd/gC094q5WrtfySrqesr7fySJ8TLbVSzUIA2BdVeDYNnQA==",
        "kid": "d9cbf11b-a154-4aca-8598-d6c569582b20"
       }
    }
```

6.13 LEGACY APPLICATIONS

Applications that are created before Fortanix DSM version 3.16 are called Legacy applications. For backward compatibility, these legacy applications will be marked with a special icon that denotes that they are legacy applications.

Fortanix has applied new security restrictions which will be applicable for applications created in Fortanix DSM version 3.16 and above.

 These new security restrictions will not be enforced on applications that are marked “legacy”.

7.0 DOCUMENT INFORMATION

7.1 DOCUMENT LOCATION

The latest published version of this document is located at the URL:

https://support.fortanix.com/hc/en-us/articles/360038354592-User-s-Guide-Fortanix-Data-Security-Manager-Key-Lifecycle-Management

7.2 DOCUMENT UPDATES

This document will typically be updated on a periodic review and update cycle.

For any urgent document updates, please send an email to: support@fortanix.com