Technical Report

# External Key Management in Element 11.7, 12.0, and Later

## Integration Guide

Erik Kemp, NetApp
June 2020 | TR-4843


## Abstract

This technical report describes how to use external key management solutions with NetApp® Element® 11.7 and later Element software. Topics include Element APIs and configuration of Key Management Interoperability Protocol (KMIP) key management servers and Element 11.7 or later.


NetApp®

TABLE OF CONTENTS





© 2020 NetApp, Inc. All Rights Reserved.

## LIST OF TABLES



## LIST OF FIGURES




© 2020 NetApp, Inc. All Rights Reserved.

# 1 Introduction

This document describes the configuration of NetApp® Element® 11.7 (or later) software for integration with external key management (EKM) solutions. EKM allows you to create and control encryption keys. Centralized key management for all infrastructure components is controlled through the Key Management Interoperability Protocol (KMIP), the industry-standard protocol.

- Keys used for encryption-at-rest features can be managed by an EKM service.
- Implementation supports industry-leading key management service (KMS) solutions.
- You can choose to allow Element software to manage the keys internally using current internal encryption at rest (EAR) implementation.
- You can convert from using keys managed by Element software to using externally managed keys.

## 1.1 Key Management Overview

When enabled, EAR prevents access to user data by locking self-encrypting drives (SEDs).

### Internal Key Management

Internal key management prevents access to user data if a single node is stolen.

EAR uses Shamir's Secret Sharing algorithm to create multiple cryptographically secure authentication key (AK) segments that are stored on individual nodes. When an AK is needed to unlock drives, multiple AK segments from multiple nodes are needed to re-create the AK.

### EKM

The AK can be acquired by the user of a key service. When the key service is used:

- Access to user data can be prevented if the entire cluster is stolen.
- EAR creates and stores the AKs by using external customer-owned KMIP key servers. These servers can be grouped in various ways. They are presented to EAR by the key service as a KMIP key provider that has one or more KMIP key servers assigned to it.
- If a theft occurs, access to user data can be prevented passively through control of access to the external key server, or EKS. For example, you can configure the EKS to accept only connections from specific data centers. You can also prevent access reactively by revoking the cluster's Transport Layer Security (TLS) certificate or by deleting the AK from the EKS before the thief can power on the stolen cluster.

# 2 Supported EKM Servers

The following EKM server platforms using the KMIP are supported.

## 2.1 Element 11.7 and Later

- Key manager: SafeNet Assured Technologies (AT) for Government
  - Key manager software: SafeNet AT OS 8.12.0
- Key manager: Gemalto SafeNet KeySecure; Gemalto SafeNet Virtual KeySecure
  - Key manager software: SafeNet OS 8.10.1
- Key manager: HyTrust Key Control
  - Key manager software: HyTrust Key Control OS 4.3


© 2020 NetApp, Inc. All Rights Reserved.

## 2.2 Element 12.0

- Key manager: Vormetric Data Security Manager (DSM)
  - Key manager software: Vormetric OS 6.3.0
- Key manager: IBM Security Key Lifecycle Manager (SKLM)
  - Key manager software: IBM OS 3.0.1.0

# 3 Hardware and Software Requirements

## 3.1 NetApp HCI Node/Drive Requirements

EKM can be used with any storage nodes that have SEDs. When you use EKM with NetApp HCI H610S-2F nodes, you can achieve a greater of security.

## 3.2 H610S-2F FIPS Node Details

You can achieve FIPS 140-2 L2 Data-at-Rest Encryption (DARE) with H610S-2F nodes, NetApp Element 11.7 (or later), EKM configuration, and a FIPS 140-2 L2 external key manager.

- H610S-2F (FIPS 140-2 certified node)
- KIOXIA CD5 Toshiba SSD (FIPS) KCD5FLUG1T92 1.92TB SSD

**Table 1) H610S-2F FIPS node details.**

| H610S-2F | |
|---|---|
| Drive capacity | 1.92TB |
| SED | FIPS SED |
| Drives | 12 |
| Drive type | NVMe U.2 SSD |
| 4K IOPS | 100,000 @ sub-ms latency |
| Network | 10/25GbE (SFP28) |
| Processor | 28 cores @ 2.2GHz |
| Memory | 384GB |
| Form factor | 1U, 31” deep |
| Effective capacity (4x) | 40TB |

To make sure the exact product and feature versions described in this document are supported for your environment, refer to the Interoperability Matrix Tool (IMT) on the NetApp Support site. The NetApp IMT defines the product components and versions that can be used to construct configurations that are supported by NetApp. Specific results depend on each customer’s installation in accordance with published specifications.

# 4 Gemalto SafeNet KeySecure

Gemalto SafeNet KeySecure offers robust capabilities for managing cryptographic keys across their entire lifecycle, including key generation, key import and export, and key rotation. With Gemalto SafeNet


© 2020 NetApp, Inc. All Rights Reserved.

KeySecure, all cryptographic keys are stored in a centralized, hardened appliance to simplify administration while ensuring tight security for the broadest array of data types.

## 4.1 Gemalto SafeNet KeySecure KMIP Server Configuration with Local Certificate Authority

First, configure a local certificate authority (CA) on SafeNet KeySecure. Second, add the local CA to the list of trusted CAs. Third, create a server certificate. Fourth, create a client certificate. Finally, configure the KMIP server settings.

### Prerequisites

1. When using a local CA, go to Security > Local CAs on SafeNet and create a self-signed root CA. Use whatever subject details you want.
2. The local CA must be added to the list of trusted CAs after it has been created. You can go to Security > Trusted CA Lists and edit the default list to include the newly created CA in the Local Certificate Authorities section.
3. Go to Security > SSL Certificates on SafeNet and fill out information under Create Certificate Request (CSR) with whatever you want. Click the certificate to get the certificate signing request (CSR), take that CSR to the CA, and get it signed as a server certificate. Take the response back to the SSL page, click the certificate, click Install Certificate, and paste the response.
4. Go to Device > Key Server > Key Server on SafeNet and add a KMIP protocol. Select Use SSL and pick the server certificate from the previous step.
5. Go to Security > Local Authentication > Local Users & Groups and create a user named SolidFire. The password can be whatever you want. The client CSR generated by NetApp Element has a default OU field of SolidFire, and by default SafeNet requires that a user with the same name be created on the system. This requirement can be changed in the settings on the Device > Key Server > Key Server page.

### Generate the CSR in Element

1. Call the `CreatePublicPrivateKeyPair` API to generate a private key internally. If you change the default organizational unit (OU), make sure that there is a user created on SafeNet with the same name as the one you have defined in the OU.
2. Call the `CreateClientCertificateSignRequest` API to generate a client CSR.

### Sign the CSR

Take that CSR from the previous API response to the Security > Local CAs on SafeNet, click your CA, and sign the CSR as a client certificate.

**Note:** To send certificates to Element by using an API, use a string and replace each newline with `\n`. When pasting certificates or CSRs in the SafeNet UI, replace each `\n` with a newline.

### Set Up the Key Provider and Key Server in Element

1. Call the `CreateKeyProviderKmip` API to create a provider.
2. Call the `CreateKeyServerKmip` API by using the client certificate you just got from SafeNet and the Local CA certificate.
3. Call the `AddKeyServerToProviderKmip` API to associate the server with the provider.
4. Call the `EnableEncryptionAtRest` API with the KeyProvider ID.


© 2020 NetApp, Inc. All Rights Reserved.

### 4.2 SafeNet KeySecure Capabilities

SafeNet KeySecure currently supports the following managed objects: certificates, private keys, public keys, templates, secret data, and symmetric keys.

Clients can submit the following requests:

- Activate
- AddAttribute
- Create
- CreateKeyPair
- DeleteAttribute
- Destroy
- DiscoverVersions
- Get
- GetAttributes
- GetAttributeList
- Locate
- ModifyAttribute
- Query
- Register
- Revoke

## 5 SafeNet Assured Technologies (AT) KeySecure for Government

The KMIP is used to transmit key management requests from clients to the SafeNet KeySecure system.

1. Verify that KeySecure is properly set up and licensed to support KMIP.
2. Create a trusted CA.
3. Create a TLS server certificate for the KMIP key server.
4. Create and configure a KMIP key server and bind the TLS server certificate to it.
5. Configure the KMIP client.
6. Create a local user in KeySecure for the KMIP client.

## 6 HyTrust KeyControl Server

The HyTrust KeyControl server is a software solution deployed from an Open Virtual Appliance (OVA) or ISO image. NetApp recommends that you read the HyTrust KeyControl Installation Overview to fully understand the KeyControl server deployment. To configure a KeyControl cluster (active-active configuration is recommended), as performed in the NetApp Element 11.7 integration validation, NetApp recommends using the OVA installation method for simplicity. This method is described in the HyTrust KeyControl OVA Installation instructions.

The KeyControl OVA must be deployed from the vCenter Server, and not from an ESXi host.

After the KeyControl server is deployed, configure the first KeyControl node as described in HyTrust's Configuring the First KeyControl Node installation instructions. After completing this procedure, add the second node as described in Adding a New KeyControl Node to an Existing Cluster (OVA Installation) to create the recommended active-active cluster.


© 2020 NetApp, Inc. All Rights Reserved.

**Note:** An active-active cluster is not a requirement, and a single KeyControl node can be deployed to perform the functions of KMIP, However, NetApp highly recommends deploying the solution with a minimum of two nodes for an active-active cluster solution that instantiates a highly available and robust architecture.

After the additional node is added, authorize the new KeyControl node as described in Authenticating New KeyControl Nodes.

**Note:** Your KeyControl license determines how many KeyControl nodes you can have in a cluster. For full information about the KeyControl licensing, see the HyTrust Managing the KeyControl License page.

## 6.1 Configuring the KeyControl Server

After the HyTrust KeyControl server is deployed and the initial installation is complete, you can configure the network settings, email server preferences, and certificate configuration. For these procedures, see the HyTrust KeyControl System Configuration documentation.

## 6.2 Configuring the KeyControl Server as a KMIP Server

To configure the KeyControl server as a KMIP server, see the HyTrust Configuring a KeyControl KMIP Server section of the administration guide.

**Note:** When EKM is used, as is the case in this solution, the KeyControl server is the KMIP server and Element is the KMIP client.

Certificates are required to facilitate the KMIP communications from the KeyControl server to Element and conversely. Although existing public key infrastructures (PKIs) can be used to import certificates for use by KeyControl and Element, the simplest solution is to use the built-in capabilities in the KeyControl server to create and publish the certificates. To perform this operation, create the certificate bundle as described in the Creating KMIP Client Certificate Bundles section of the HyTrust KeyControl administration guide.

### Verifying External Key Manager Communication on the HyTrust KeyControl Server

To verify that Element is communicating and requesting keys from the KeyControl server, use the Objects tab in the KeyControl UI. See the Managing KMIP Objects section of the HyTrust KeyControl administration guide.

**Note:** You might have to refresh the tab or page (by using the refresh list in the KeyControl UI) to view the updated requests. Also, if any changes are made to the certificates or KMIP configuration, you might need to restart the KMIP server, as outlined in the Restarting a KMIP Server section of the HyTrust KeyControl administration guide.

It is important to note that restarting the KMIP server does not restart the KeyControl server, just the KMIP service.

## 6.3 High-Availability Considerations

The HyTrust KeyControl solution uses an active-active deployment, which provides high-availability capability to manage encryption keys. NetApp highly recommends this deployment configuration. In an active-active cluster, changes made to any KeyControl node in the cluster are automatically reflected on all nodes in the cluster. For full information about the HyTrust KeyControl solution, see the HyTrust KeyControl product documentation.


© 2020 NetApp, Inc. All Rights Reserved.

# 7 Vormetric Data Security Manager

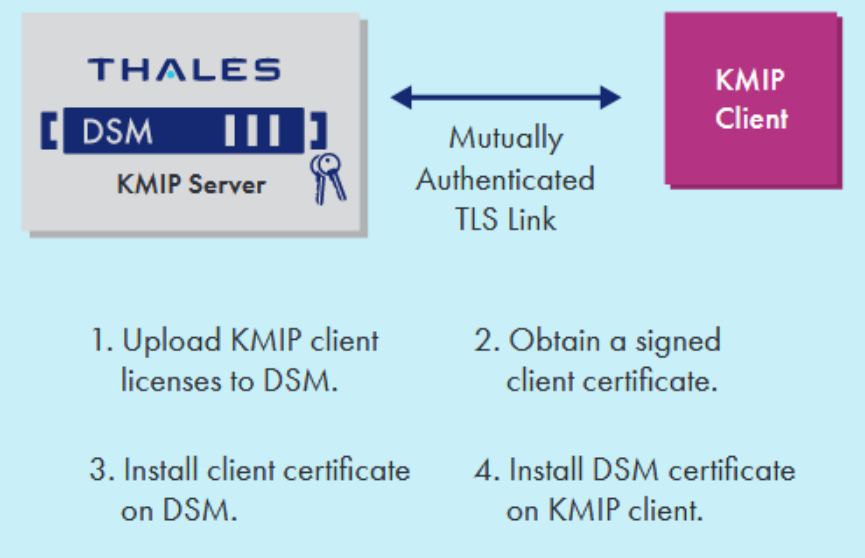
The Vormetric Data Security Manager (DSM) delivers centralized controls that enable consistent and repeatable management of encryption, access policies, and security intelligence for all your structured and unstructured data. The DSM is available as FIPS 140-2–certified and Common Criteria–certified virtual and physical appliances. The DSM is part of the Vormetric Data Security Platform, a solution that makes it easy and efficient to manage data-at-rest security across your entire organization. Built on an extensible infrastructure, the platform features multiple data security products that can be deployed individually or in combination to deliver advanced encryption, tokenization, and centralized key management. As a result, your security teams can address your data security policies, compliance mandates, and best practices, while reducing administration effort and TCO.

The DSM offers complete support for the KMIP standard. Effectively, any device or client software that is KMIP-enabled can communicate with the DSM to facilitate management of encrypted keys. Examples of KMIP clients include management systems in hyperconverged environments, SEDs in storage systems, and native encryption in next-generation databases and virtualized environments.

## 7.1 KMIP-Based Communications: How It Works

Following is a high-level overview of how the process works. The DSM needs to be prepared for KMIP support, which entails adding a KMIP client to the DSM KMIP client list. A KMIP client is registered with the DSM. Trust between the DSM and the KMIP client needs to be established. The KMIP protocol requires a mutually authenticated TLS connection between clients and servers. After the client is registered with the DSM for secure communications, a certificate for the KMIP client will be created. This certificate will be used by the KMIP client for TLS authentication (Figure 1).

**Figure 1) DSM KMIP overview.**




## 7.2 Vormetric Setup Instructions

Setting up a Vormetric EKS requires some additional steps.

1. Vormetric requires that the common name in the cluster's certificate match the host name of the Vormetric host.

   When calling `CreatePublicPrivateKeyPair`, be sure to provide the `commonName` parameter.

2. Vormetric does not act as a CA. Therefore, you must use a third-party CA to sign the cluster's CSR.

   For development purposes, you can use the following commands:


© 2020 NetApp, Inc. All Rights Reserved.

```
openssl genrsa -out ca.privkey.pem 2048 # Generate CA private key
openssl req -x509 -new -nodes -key ca.privkey.pem -sha256 -days 3650 -out ca.cert.pem # Generate
CA self-signed root certificate
openssl x509 -req -days 3600 -in cluster.csr.pem -CA ca.cert.pem -CAkey ca.privkey.pem -
CAcreateserial -out cluster.cert.pem -sha256 # Sign the cluster's CSR
openssl x509 -in cluster.cert.pem -text # View the cluster's new cert
```

3. Add the third-party CA root certificate as a trusted KMIP root certificate on the Vormetric server.
   a. Log in to Vormetric Data Security Manager as an administrator.
   b. Navigate to System > KMIP Trusted CA Certificates.
   c. Import/Update your CA's root certificate.
4. Add the cluster's certificate to the Vormetric host associated with your Vormetric domain.
   a. Log in to Vormetric Data Security Manager as the domain's administrator.
   b. Navigate to Hosts > Hosts.
   c. Click the Host Name of the desired host.
   d. Click Import KMIP Cert.
   e. Upload the cluster's certificate.

      Vormetric sometimes erroneously claims that valid certificates are invalid. You can try converting your certificate to DER if it's PEM-encoded (or to PEM if it's DER-encoded) and uploading the resulting certificate.
      Example command:

```
openssl x509 -inform PEM -in cert.pem -outform DER -out cert.der
```

   f. Click OK.

# 8 IBM Security Key Lifecycle Manager

For details, see Installing and migrating IBM Security Key Lifecycle Manager (SKLM).

This scenario shows how to set up a Secure Sockets Layer (SSL) handshake between IBM Security Key Lifecycle Manager server and the client device.

The SSL handshake enables IBM Security Key Lifecycle Manager server and client devices to establish the connection for secure communication. IBM Security Key Lifecycle Manager provides the Server Configuration wizard to configure server and the client devices for SSL handshakes.

You must complete the following steps in the wizard for the SSL/TLS handshake:

1. Create a self-signed SSL/KMIP server certificate.
2. Export the SSL/KMIP server certificate that is created in step 1 to a certificate file in an encoded format for use by the client device. You can also export an existing certificate.
3. Import the client communication certificate to the IBM Security Key Lifecycle Manager server.

## 8.1 KMIP Client Configuration

The IBM Security Key Lifecycle Manager server supports KMIP communication with client devices. You can create and manage cryptographic objects by using a set of operations that IBM Security Key Lifecycle Manager provides.

You can use the IBM Security Key Lifecycle Manager UI for the following cryptographic object management activities:

- Registering client devices with the server
- Creating and configuring cryptographic objects for the registered client devices


© 2020 NetApp, Inc. All Rights Reserved.

- Viewing objects for the registered client devices
- Modifying client device information by adding more objects or associating a new certificate
- Deleting client devices and the associated objects from the server
- Searching for objects that are managed by the server

See Registering KMIP-compliant client devices for instructions on managing client devices and objects.

## 8.2 SKLM Installation

1. Obtain the SKLM tar files for installation and scp SKLM files to / directory of scs client created.

```
[root@scspa1677438001 /]# ls -lrt *.tar*
-rw-r--r-- 1 root root 3878256640 Oct 14 14:14 SKLM3.0.1_LIN64_1F2_ML_.tar
-rw-r--r-- 1 root root 3040748415 Oct 14 14:14 SKLM3.0.1_LIN64_2F2_ML_.tar.gz
[root@scspa1677438001 /]#
```

2. Untar both SKLM files, which will create `/disk1` and `/disk2` folders.

```
[root@scspa1677438001 /]# tar xvf SKLM3.0.1_LIN64_1F2_ML_.tar
disk1/
disk1/im/
disk1/im/con-disk-set-inst.sh
disk1/im/configuration/
disk1/im/configuration/org.eclipse.equinox.launcher/
disk1/im/configuration/org.eclipse.equinox.launcher/com.ibm.cic.agent.ui_1.4.24.v20160506_0238/
disk1/im/configuration/org.eclipse.equinox.launcher/com.ibm.cic.agent.ui_1.4.24.v20160506_0238/splash.bmp
disk1/im/configuration/org.eclipse.update/
disk1/im/configuration/org.eclipse.update/last.config.stamp
disk1/im/configuration/org.eclipse.update/platform.xml
```

```
root@scspa1677438001 /]# tar xvf SKLM3.0.1_LIN64_2F2_ML_.tar.gz
isk2/
isk2/ad/
isk2/ad/atoc/
isk2/ad/atoc/atoc.xml
isk2/ad/atoc/nq/
isk2/ad/atoc/nq/native_file.xml
isk2/ad/atoc/nq/native_zip.xml
isk2/ad/files/
isk2/ad/files/file000001
isk2/ad/files/file000002
```

3. Install the following libraries using `yum`:

```
yum -y install libstdc*
yum -y install libaio*
yum -y install libaio.i686
yum -y install libstdc++.i686
yum -y install glibc-2.12-1.166.el6_7.3.i686
yum -y install gtk2.i686
yum -y install libXtst.i686
yum -y install libpam.so.0
yum -y install libaio.so.1
yum -y install libstdc++.so.5
```

4. Change directory to `disk1`.

```
cd /disk1
```

5. Create an encrypted password (use Netapp123@) by using the `imcl` tool.


© 2020 NetApp, Inc. All Rights Reserved.

```
[root@scspa1677438001 disk1]# ./im/tools/imcl encryptstring Netapp123@
RAsViNK4dqNTYtaLp9gYyA==
[root@scspa1677438001 disk1]#
```

6. Modify all encrypted passwords (which have == at end) in `SKLM_Silent_Linux_Resp.xml` to the new encrypted password just created.

```
vim SKLM_Silent_Linux_Resp.xml
```

7. Run the following command:

```
./silent_install.sh SKLM_Silent_Linux_Resp.xml -acceptLicense
```

8. Log in to the SKLM key server—for example, https://10.234.232.15/ibm/SKLM/login.jsp.
9. Use the following login:

```
sklmadmin/Netapp123@
```

## 8.3 SKLM Configuration

To configure SKLM, you must create a certificate signing request (CSR), send it to the CA, obtain the certificate chain from the CA, and import the root certificate into the SKLM server.

1. Create a CSR with the SKLM CLI:
   a. On the SKLM server system, start the CLI to SKLM.

      ```
      /opt/IBM/WebSphere/AppServer/bin/wsadmin.sh -username SKLMAdmin -
      password Netapp123@ -lang jython
      ```

   b. In the SKLM CLI, enter the following command on one line:

      ```
      print AdminTask.tklmCertGenRequest('[-alias sklmCert -cn sklm -validity
      3650 -keyStoreName defaultKeyStore -fileName myCertRequest.crt -usage
      SSLSERVER]')
      ```

      This creates the following file.

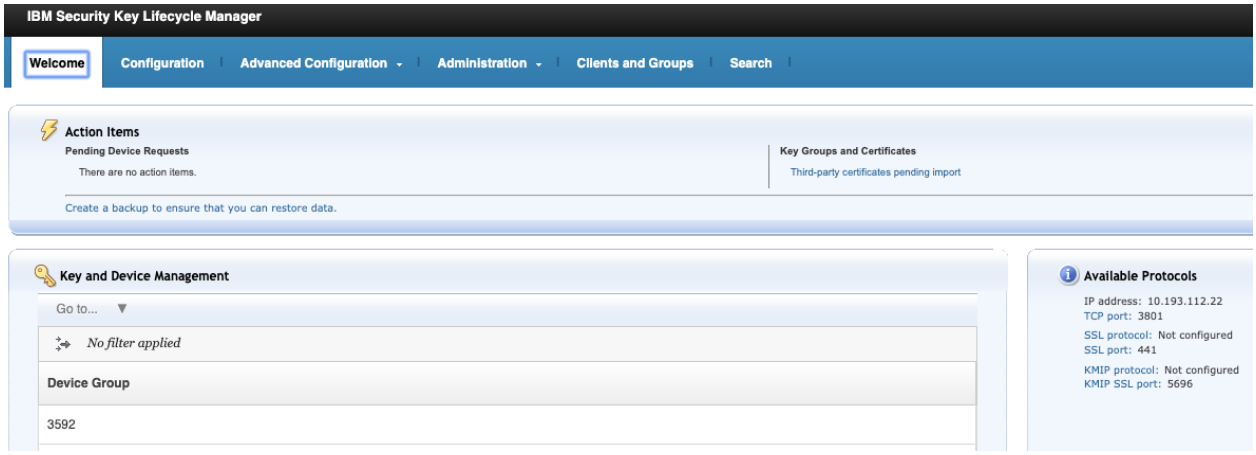
      ```
      /opt/IBM/WebSphere/AppServer/products/sklm/data/myCertRequest.crt
      ```

2. Send the CSR file from step 1 to the CA. For our testing, we are self-signing the certificate. Signing with the local CA is explained in the next step.
3. After receiving the signed certificate file from the CA, copy it to the SKLM system or copy the contents to the certificate file. Example:

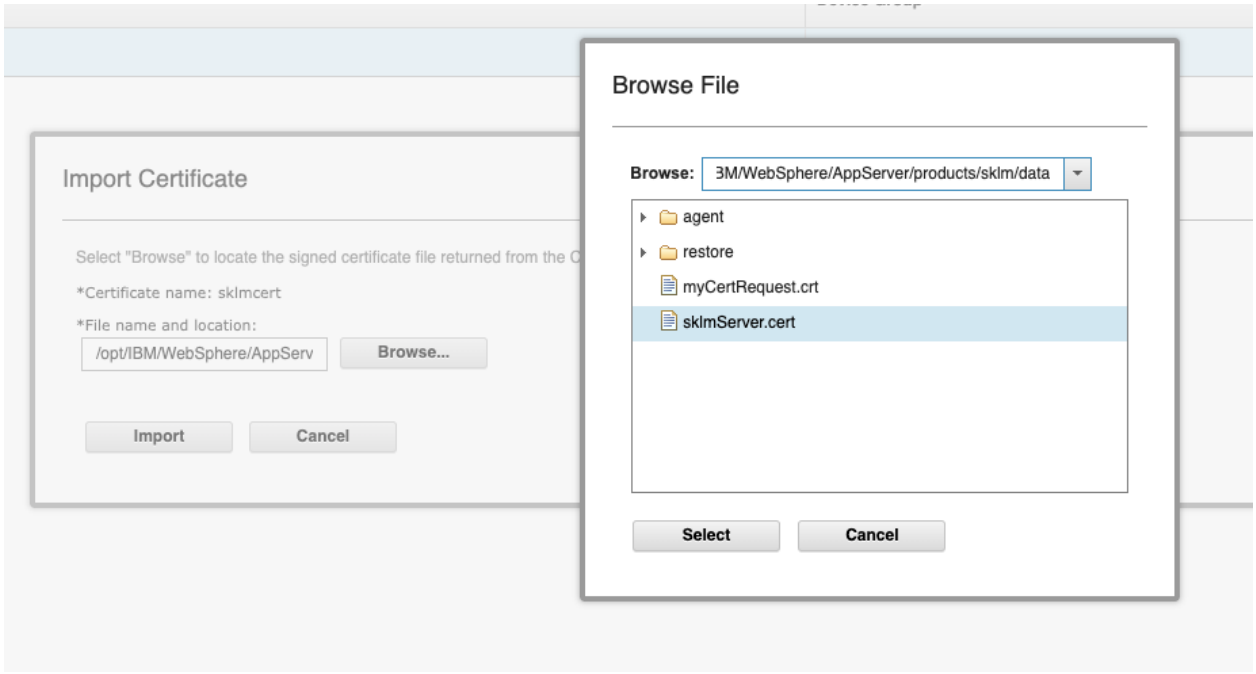
```
/opt/IBM/WebSphere/AppServer/products/sklm/data/sklmServer.cert
```

4. Import the root certificate into the SKLM server by using the SKLM UI:
   a. On the Welcome page, in the Action Items section, in the Key Groups and Certificates area, click You Have Pending Certificates.
   b. In the Pending Certificates table, select the certificate that you want to import, and click Import.
   c. In the File Name and Location field, type the path and name of the certificate file and click Import.


© 2020 NetApp, Inc. All Rights Reserved.




5. Click Third-Party Certificates Pending Import and browse to the certificate.

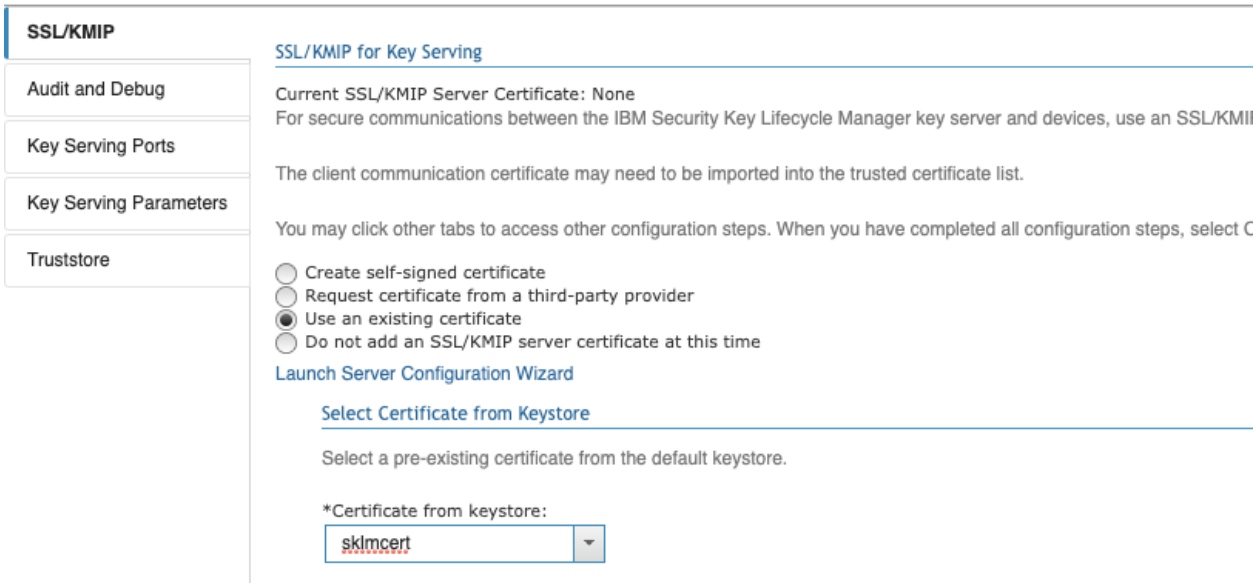



6. From the Welcome page, click KMIP Protocol.

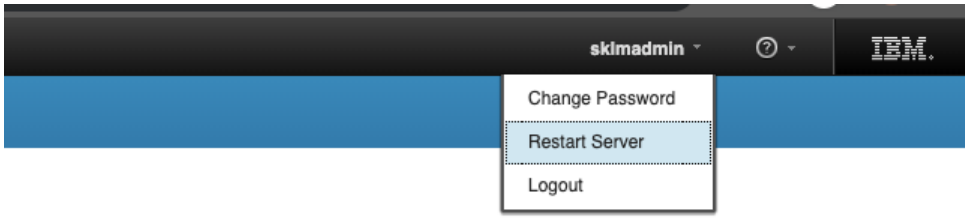


© 2020 NetApp, Inc. All Rights Reserved.

7. Now select Use an Existing Certificate.




8. Restart the SKLM.



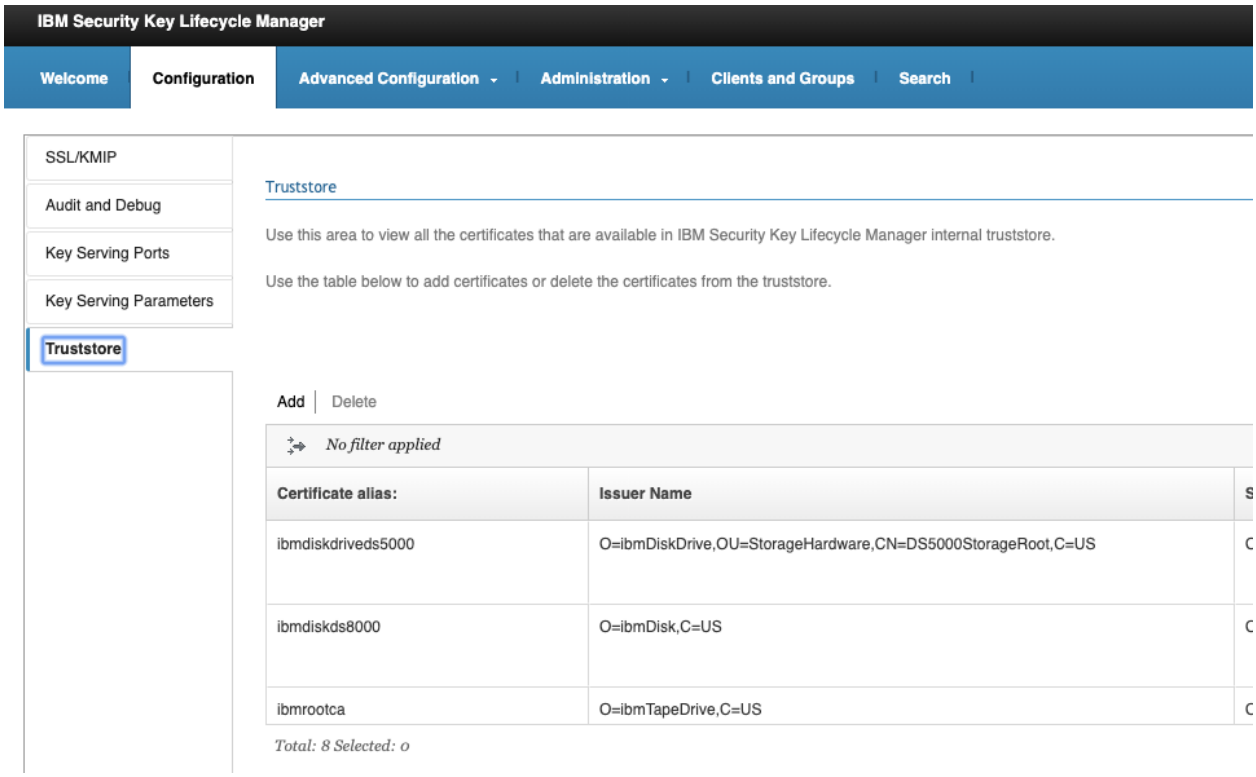

## 8.4 Configuring SKLM to Trust Element by Using a Shared CA Certificate

1. Copy the CA public key certificate (`ca.cert.pem`) to:

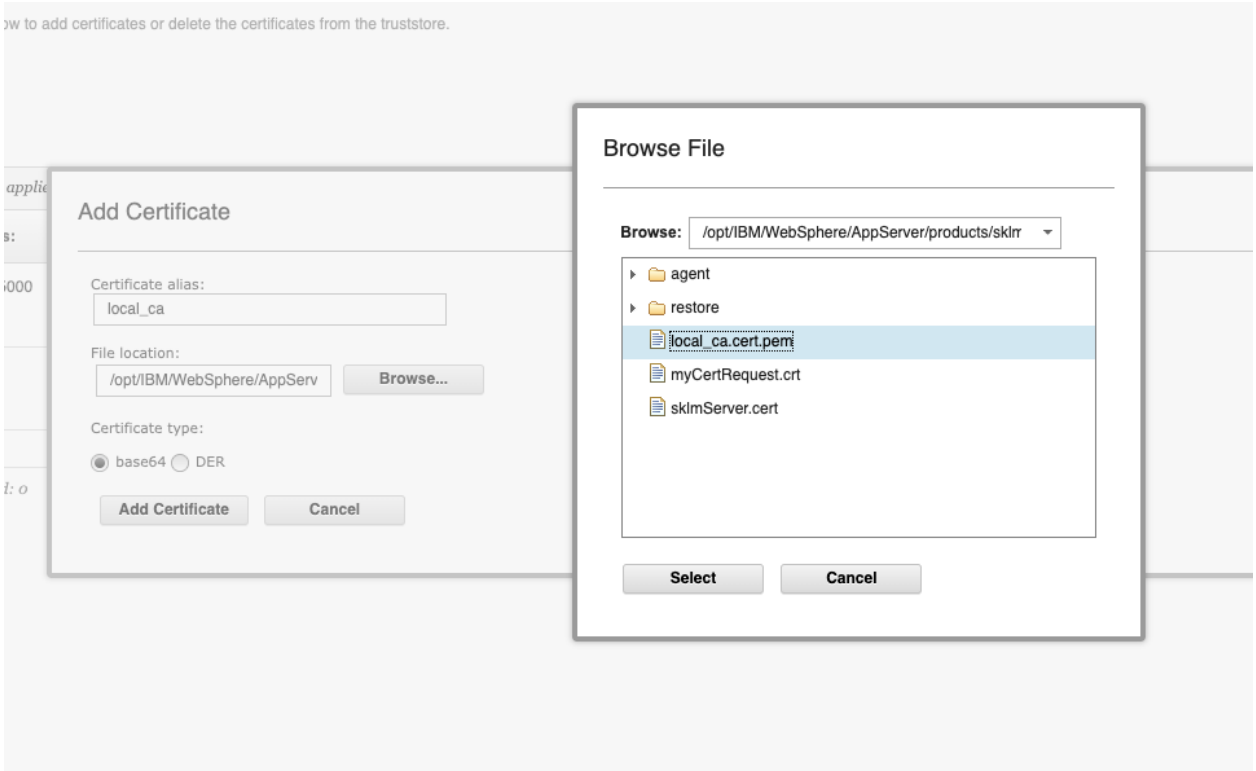
```
/opt/IBM/WebSphere/AppServer/products/sklm/data/local_ca.cert.pem
```

2. Add the certificate to the Truststore.


© 2020 NetApp, Inc. All Rights Reserved.




3. Browse to the certificate to add to the Truststore.




4. Configure Element by creating the CSR, provider, and so on.

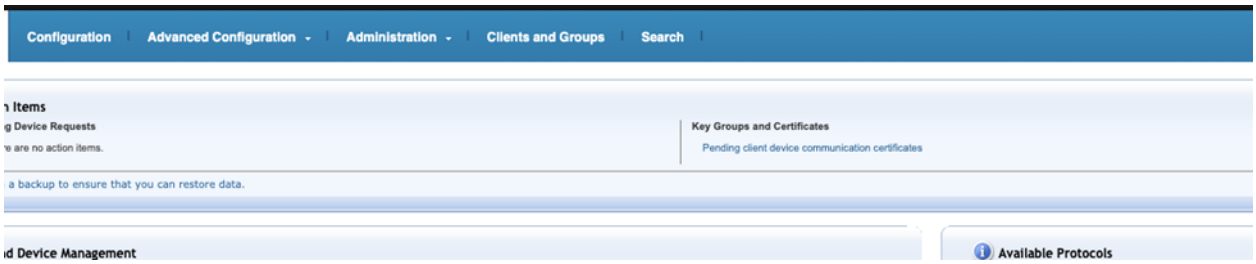

© 2020 NetApp, Inc. All Rights Reserved.

Upon executing `TestKeyServerKmip` or `AddKeyServerToProviderKmip`, you receive the following error.

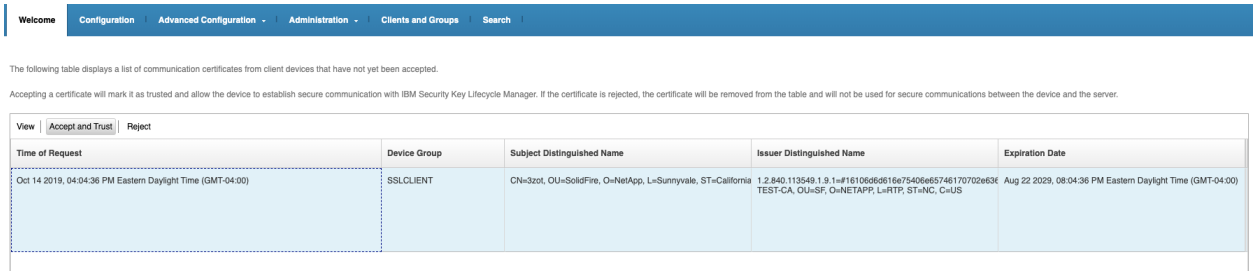
```
KMIP error: Response status: OPERATION_FAILED Reason: INVALID_MESSAGE Message: CTGKS0070E Server does not trust the client certificate.
```

5. From the SKLM UI, click Pending Client Device Communication Certificates to add the client certificate to the server.

   You can also copy the signed client certificate to the SKLM server.

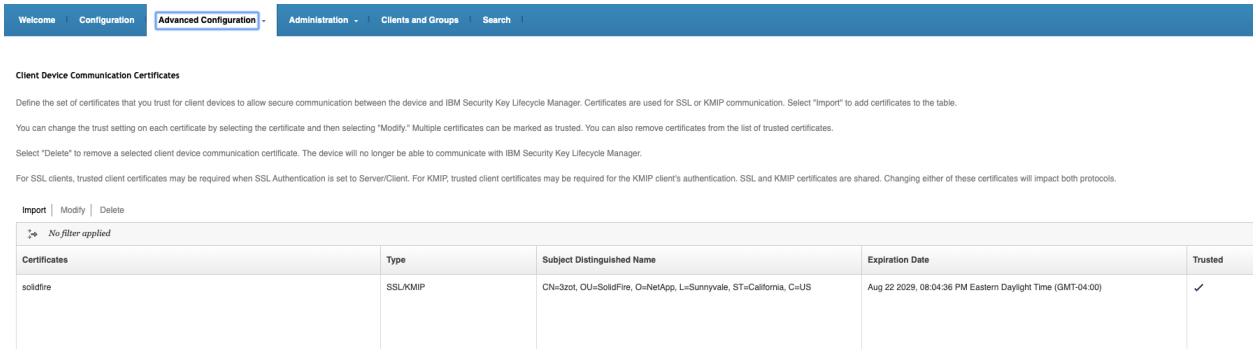



6. Click Accept and Trust.



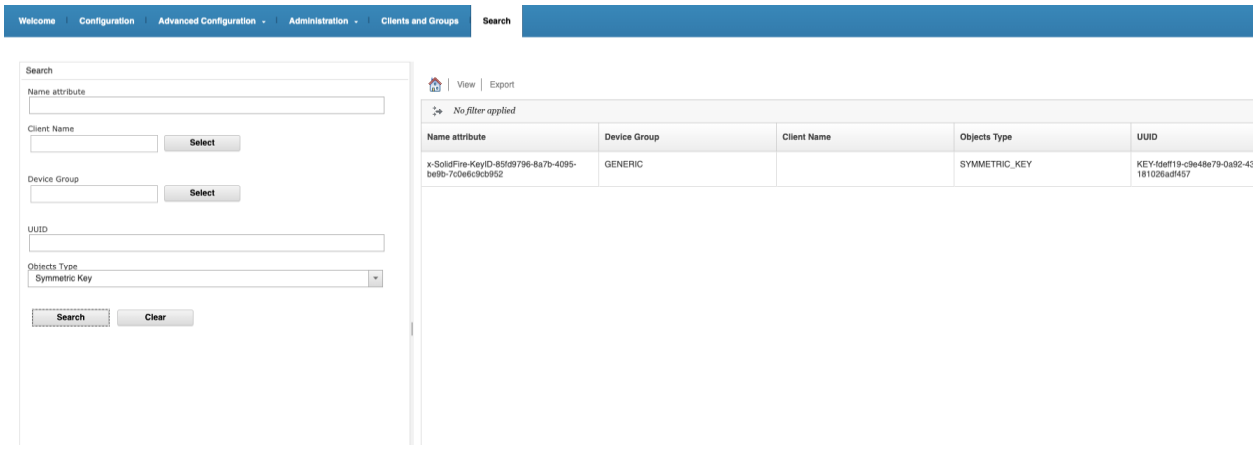

**Note:** If you're copying the client certificate manually, you can import it from the following UI screen.

7. Click import to import it.




8. You can view all the Element created keys from the following UI.


© 2020 NetApp, Inc. All Rights Reserved.




# 9 Getting Started with EKM

EKM provides secure AK management with an off-cluster external key server (EKS). The EKS provides secure generation and storage of the AKs.

The AKs are used to lock and unlock SEDs when EAR is enabled on the cluster. The cluster uses the KMIP, an OASIS-defined standard protocol, to communicate with the EKS.

## 9.1 Setting Up EKM

You can follow these basic steps by using the NetApp Element API to set up your EKM feature. Details of each API method can be found in the Element API Reference Guide.

1. Establish a trust relationship with the EKS.
   a. To create a public/private key pair for the Element cluster that is used to establish a trust relationship with the key server, call the `CreatePublicPrivateKeyPair` API method.
   b. Get the CSR that the CA needs to sign. The CSR enables the key server to verify that the Element cluster that will access the keys is authenticated as the Element cluster. Call the `GetClientCertificateSignRequest` API method.
   c. Use the EKS/CA to sign the retrieved CSR. See third-party documentation for more information.
2. Create a server and provider on the cluster to communicate with the EKS. A key provider defines where a key should be obtained, and a server defines the specific attributes of the target EKS.
   a. To create a key provider in which the key server details will reside, call the `CreateKeyProviderKmip` API method.
   b. To create a key server that provides the signed certificate and the public key of the CA, call the `CreateKeyServerKmip` and `TestKeyServerKmip` API methods.

      If the test fails, verify your server connectivity and configuration. Then repeat the test.
   c. To add the key server into the key provider container, call the `AddKeyServerToProviderKmip` and `TestKeyProviderKmip` API methods.

   If the test fails, verify your server connectivity and configuration. Then repeat the test.
3. Enable encryption at rest (EAR).
   a. To enable EAR, provide the ID of the key provider that contains the key server used for storing the keys. Call the `EnableEncryptionAtRest` API method.

   **Note:** To enable EAR using an EKM configuration, you must use the API. Using the existing Element UI button causes the system to revert to using internally generated keys.


© 2020 NetApp, Inc. All Rights Reserved.

## 9.2 Recovering Inaccessible or Invalid AKs

Occasionally, an error can occur and requires intervention. If an error occurs, a cluster fault code is generated. The two most likely cases are described here.

- The cluster is unable to unlock the drives because of a `KmipServerFault` cluster fault. This fault can occur when the cluster first boots up and the key server is inaccessible, or the required key is unavailable. To recover, follow the recovery steps in the cluster fault codes (if any).
- A sliceServiceUnhealthy fault might be set because the metadata drives have been marked as failed and placed into the "Available" state. To recover:
  - Re-add the drives.
  - After 3 to 4 minutes, check that the `sliceServiceUnhealthy` fault has cleared.

## 9.3 EKM API Commands

This section lists all the APIs available for managing and configuring EKM.

The following APIs are used for establishing a trust relationship between the cluster and external customer-owned servers:

- `CreatePublicPrivateKeyPair`

  Creates SSL public and private keys. These keys can be used to generate CSRs.

  There can be only one key pair in use for the cluster. To replace the existing keys, make sure that they are not being used by any providers before invoking this API.

- `GetClientCertificateSignRequest`

  Used for defining the specific details of external customer-owned servers.

- `CreateKeyServerKmip`

  Creates a KMIP key server with the specified attributes. The server is not contacted as part of this operation, so it need not already exist or be configured.

  For clustered key server configurations, all server nodes' host names or IP addresses must be provided in the `kmipKeyServerHostnames` parameter.

- `ModifyKeyServerKmip`

  Modifies a KMIP key server to the specified attributes. The only required parameter is `keyServerID`. A request that contains only the `keyServerID` parameter is a no-op and no error will be returned. Any other parameters that are specified will replace the existing values on the KMIP key server with the specified key server ID. Because this server might be part of an active provider, this replacement will result in contacting the server to verify that it is functional. Multiple host names or IP addresses must only be provided to the `kmipKeyServerHostnames` parameter if the key servers are in a clustered configuration.

- `DeleteKeyServerKmip`

  Delete the specified KMIP key server. A KMIP key server can be deleted unless it is the last one assigned to its provider, and that provider is active (providing keys that are currently in use).

- `GetKeyServerKmip`

  Returns the specified KMIP key server object.

- `ListKeyServersKmip`

  Returns the list of KMIP key servers that have been created through `CreateKeyServerKmip`. You can filter the list by specifying more parameters.

- `TestKeyServerKmip`


© 2020 NetApp, Inc. All Rights Reserved.

Tests whether the specified KMIP key server is functioning normally.

The following APIs are used for creating and maintaining key providers that manage external key servers:

- `CreateKeyProviderKmip`

  Creates a KMIP key provider with the specified name. A key provider defines a mechanism and location to retrieve AKs. A KMIP key provider represents a collection of one or more KMIP key servers. A newly created KMIP key provider will not have any KMIP key servers assigned to it. To create a KMIP key server, use `CreateKeyServerKmip`; to assign it to a provider created with this method, use `AddKeyServerToProviderKmip`.

- `DeleteKeyProviderKmip`

  Deletes the specified inactive key provider.

- `AddKeyServerToProviderKmip`

  Adds (assigns) the specified KMIP key server to the specified key provider. This action results in contacting the server to verify that it is functional and to synchronize keys if there are multiple key servers assigned to the provider. This synchronization can result in conflicts that could cause this method to fail. If the specified KMIP key server is already assigned to the specified key provider, this is a no-op and no error will be returned. You can remove the assignment (unassign the key server) by using `RemoveKeyServerFromProviderKmip`.

- `RemoveKeyServerFromProviderKmip`

  Removes (unassigns) the specified KMIP key server from the provider it was assigned to through `AddKeyServerToProviderKmip`. A KMIP key server (if any) can be unassigned from its provider unless it is the last one and that provider is active (providing keys that are currently in use). If the specified KMIP key server is not assigned to a provider, this is a no-op and no error will be returned.

- `GetKeyProviderKmip`

  Returns the specified KMIP key provider object.

- `ListKeyProvidersKmip`

  Returns the list of KMIP key providers that have been created through `CreateKeyProviderKmip`. You can filter the list by specifying more parameters.

- `TestKeyProviderKmip`

  Tests whether the specified key provider is functioning normally.

The following are additional APIs:

- `EnableEncryptionAtRest`

  Initiates the process of setting a password on SEDs within the cluster. This feature is not enabled by default but can be toggled on and off as needed.

  If a password is set on a SED that is removed from the cluster, the password remains set and the drive is not secure-erased. Data can be secure-erased through the `SecureEraseDrives` method.

  If no parameters are specified, the password is generated internally and at random (the only option for endpoints before Element 12.0). This generated password is distributed across the nodes by means of Shamir's Secret Sharing algorithm such that at least two nodes are required to reconstruct the password. The complete password to unlock the drives is not stored on any single node and is never sent across the network in its entirety. This approach protects against the theft of any number of drives or a single node.

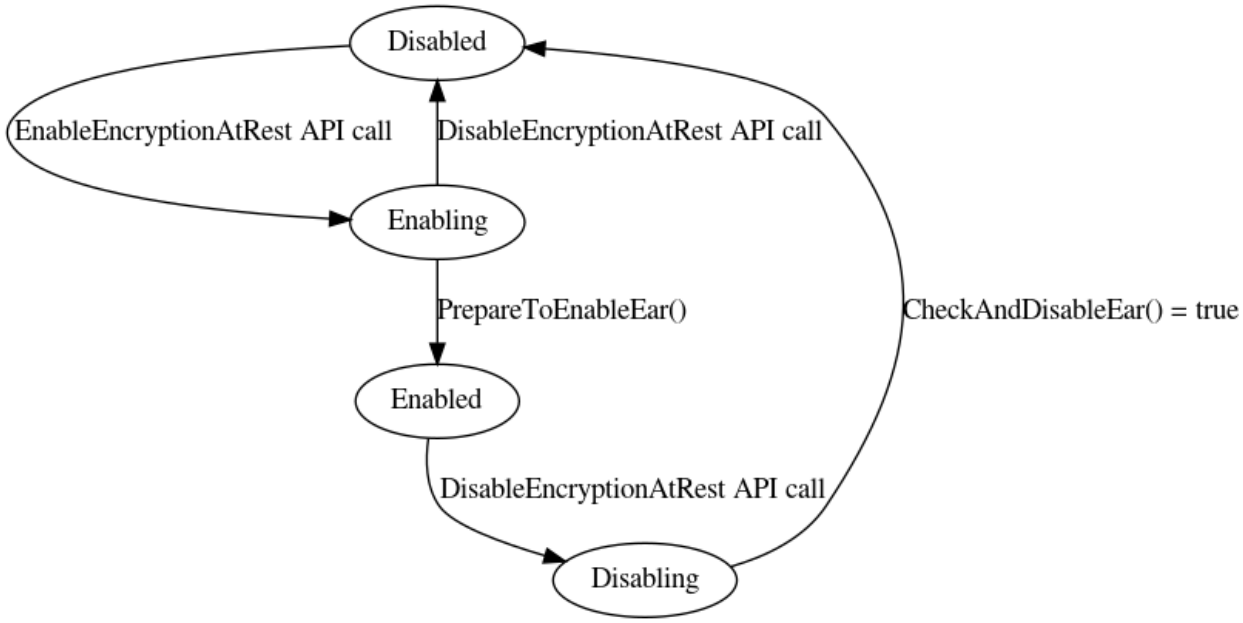
  If a `keyProviderID` parameter is specified, the password is generated/retrieved as appropriate for the type of provider. For a KMIP key provider, it would commonly be generated/retrieved through a KMIP key server. (See `CreateKeyProviderKmip`.) After this operation, the specified provider is considered active and cannot be deleted until `DisableEncryptionAtRest` is called.

- `DisableEncryptionAtRest`


© 2020 NetApp, Inc. All Rights Reserved.

Initiates the process of removing the password from SEDs within the cluster (Figure 2).

**Figure 2) EKM state machine.**




# 10 EKM Cluster Faults, Errors, Events, and Logging

The following issues can arise in systems associated with EKM.

## 10.1 Key Service Cluster Faults

EKM might raise one or more `KmipServerFault` faults. The fault message contains details, and usually gives information about how to find the root cause or resolve the issue.

Usually, these faults are caused by the EKS being unreachable, or by key/certificate issues (expiration, key/certificate mismatches, untrusted certificates, and so on).

If the fault message is “Key server <id> generic failure,” an unanticipated error has occurred. There might be more information in the logs, particularly if the fault was caused by an OpenSSL error.

If the fault message is “Key server <id> unknown failure,” EKM has reached code that should be unreachable, which indicates a bug in error handling.

### KmipCertificateFault

The following KMIP certificate faults can be generated:

- **Root Certification Authority (CA) certificate is nearing expiration.**

**Table 2) KMIP certificate faults for RCA certificate expiration.**

| Status | Fault Description |
|---|---|
| Warning | Certificate expires within 30 days. |
| Error | Certificate expires within 7 days. |
| Critical | Certificate expires within 3 days. |


© 2020 NetApp, Inc. All Rights Reserved.

**Corrective action**: The fault details show how many days are left until the certificate expiration date. Before the certificate expires, update the expiring certificate. Acquire a new certificate from the root CA with an expiration date at least 30 days out, and use the `ModifyKeyServerKmip` API to provide the updated root CA certificate.

- **Client certificate is nearing expiration**.

**Table 3) KMIP certificate faults for client certificate expiration.**

| Status | Fault Description |
|---|---|
| Warning | Certificate expires within 30 days. |
| Error | Certificate expires within 7 days. |
| Critical | Certificate expires within 3 days. |

**Corrective action**: The fault details show how many days are left until the certificate expiration date. Create a new CSR by using the `GetClientCertificateSigningRequest` API. Have it signed and make sure that the new expiration date is at least 30 days out. Use the `ModifyKeyServerKmip` API to replace the expiring KMIP client certificate with the new certificate.

- **Root Certification Authority (CA) certificate expired**.

  This fault is logged with “Critical” severity.

  **Corrective action**: Acquire a new certificate from the root CA with an expiration date at least 30 days out, and use the `ModifyKeyServerKmip` API to provide the updated root CA certificate.

- **Client certificate expired**.

  This fault is logged with “Critical” severity.

  **Corrective action**: Create a new CSR using the `GetClientCertificateSigningRequest` API. Have it signed and make sure that the new expiration date is at least 30 days out. Use the `ModifyKeyServerKmip` API to replace the expired KMIP client certificate with the new certificate.

- **Invalid Root Certification Authority (CA) certificate**.

  This fault is logged with “Critical” severity.

  **Corrective action**: Double-check to see if the correct certificate was provided, and, if needed, reacquire the certificate from the root CA. Use `ModifyKeyServerKmip` API to install the correct KMIP client certificate.

- **Invalid Client certificate**.

  This fault is logged with “Critical” severity.

  **Corrective action**: Check to see if the correct KMIP client certificate is installed. The root CA of the client certificate should be installed on the EKM server. If you need to update the KMIP client certificate, use `ModifyKeyServerKmip` to install the correct KMIP client certificate.

## KmipServerFault

All KMIP server faults are logged with “Critical” severity. The following KMIP server faults can be generated:

- **Connection Failure**.

  **Corrective action**: If none of the nodes can access the EKM server, the key server ID is provided in the fault details. Check to see whether the server is alive and reachable by means of the management network. If only some nodes are unable to access the EKM server, the nodes that are unable to reach the key server are listed in the fault details. Perform troubleshooting at the network or specific node level to determine why only some of the nodes can access the EKM server.

- **Authentication Failure**.


© 2020 NetApp, Inc. All Rights Reserved.

**Corrective action**: Check to see if the correct root CA and KMIP client certificates are being used. If you need to update any of the certificates, use `ModifyKeyServerKmip` to install the correct certificate.

- **Server Error**.

  **Corrective action**: The fault gives details about the error. Depending on the error, troubleshooting on the EKM server might be necessary.

## 10.2 Key Service API Errors

Aside from generic/low-level SF exceptions, EKM key service API failures can involve these exceptions:

- `xCsrGenerationFailed`: This exception occurs during `GetClientCertificateSignRequest`, but probably indicates that a bad value was previously passed to `CreatePublicPrivateKeyPair`.
- `xEkmGenericError`: This exception shouldn't ever be returned from an API (though it might be seen in logs). It probably indicates either a bug in the EKM code or a problematic change in the OpenSSL/Cryptsoft libraries (unless seen as the cause of some other exception or fault). If it appears in the logs, this exception might indicate that a Cryptsoft KMIP error code is not correctly handled by `KmipHelper`.
- `xKeyPairInUse`: An attempt was made to regenerate the EKM TLS client key pair while it is still in use. Ensure that the keys are not being used (for example, by disabling EAR) and retry.
- `xKeyProviderActive`: An attempt was made to delete an active key provider. Ensure that the provider is not being used (for example, by disabling EAR) and retry.
- `xKeyProviderConfigurationError`: A test of the key provider failed because of one of the following:
  - `xKeyServerAuthFailed`
  - `xKeyServerInvalidAuth`
  - `xOpenSslError`
  - `xEkmGenericError`
- `xKeyProviderFailed`: A test of the key provider failed because of an `xKeyServerFailed` result.
- `xKeyProviderUnavailable`: A test of the key provider failed because of either `xKeyServerUnavailable` or `xKeyServerSslError` result.
- `xKeyServerAuthFailed`: An attempt to connect to the EKS failed because of an authorization failure. Probable causes:
  - Wrong EKS server. Check the host name/port of the server, correct with `ModifyKeyServerKmip`, and retry.
  - Incorrect client-side configuration. Check the TLS client private key, client certificate, and CA root certificate and retry.
  - Incorrect server-side configuration. Check that the EKS is configured to trust the client certificate (or the client certificate's CA root certificate).
- `xKeyServerSslError`: A TLS error occurred while the system was connecting to the EKS. (The EKS is reachable.) This exception is most likely caused by an incorrect key, certificate, or CA root certificate.
- `xKeyServerFailed`: Some operation interacting with the EKS failed. The most likely causes are:
  - Attempting to use an unsupported EKS type
  - Failure to connect to, authenticate to, or verify the EKS.
- `xKeyServerFailedKeyAlreadyExists`: The user is trying to create a key server that already exists. Try checking the output of `ListKeyServersKmip`.


© 2020 NetApp, Inc. All Rights Reserved.

- `xKeyServerHasProviderAssigned`: The user attempted to delete a server that is "in use" (is assigned to a provider). Remove the server from the provider with `RemoveKeyServerFromProviderKmip`; then retry.
- `xKeyServerInvalidAuth`: The TLS client key pair, client certificate, or EKS CA root certificate is invalid (or some combination thereof).
  - This exception probably means that the user provided improperly formatted data. Make sure that the key/certificate input is provided in the correct format (`CreateKeyServerKmip`, `ModifyKeyServerKmip`) and retry.
  - This exception could indicate a bug in EKM code or a corruption of ZooKeeper data.
- `xKeyServerLastForActiveProvider`: The user attempted to remove the only server or the last server from an in-use provider. Either add another server with `AddKeyServerToProviderKmip` first, or delete the provider with `DeleteKeyProviderKmip`, or stop using the key provider (for example, by disabling EAR). Then retry.
- `xKeyServerUnavailable`: The EKS cannot be reached. Check the network route and configured host name and port. Then retry. (`ModifyKeyServerKmip` can be useful.)

  These exceptions might appear intermittently and be automatically cleaned up (for example, if there is a somewhat unreliable network connection.)
- `xNoOnlineKeyProviderWithThatID`: An attempt was made to retrieve a non-existent key provider with `GetKeyProviderKmip`. Make sure that the correct provider ID was provided and retry.
- `xSslKeysDoNotExist`: `GetClientCertificateRequest` was called without a call to `CreatePublicPrivateKeyPair`. First call the latter, then retry the former.

## 10.3 Key Service Logging Information

Default logs are stored in `sf-master.info` and `sf-master.error`. `KS` is the logging component. `OpenSSL` is also a relevant logging component.

By default, Cryptsoft verbose KMIP logging is disabled. To enable it, set the `cKmipCmdExtraFlags` API constant to 161. (161 is the bitwise `OR` of `KMIP_CMD_FLAGS_LOG` (0x1), `KMIP_CMD_FLAGS_DEBUG` (0x20), and `KMIP_CMD_FLAGS_VERBOSE` (0x80). This additional logging appears in `/var/log/sf-kmip.info`.

### Useful Logging Strings

Grepping for these strings in the log files can produce useful information in debugging EAR or key service issues.

- `KMIP_ERROR_`
  - The Cryptsoft KMIP library indicated an error, possibly caused by OpenSSL errors. (If it was cases by OpenSSL errors, they should appear on subsequent log lines. But they will be [OpenSSL] logs, not [KS] logs.)
  - `KMIP_ERROR_IO` is a generic Cryptsoft error that appears for several issues, but generally indicates that there was an error in communication between SF and the EKS.
- `OpenSSL error stack`
  - This error is logged during or before logging of the stack of OpenSSL errors. KMIP errors often occur because of OpenSSL errors.
- `SSL_shutdown:shutdown while in init`
  - This error is like `KMIP_ERROR_IO` in that it is a generic error that OpenSSL might give when connecting to the EKS. It is often a symptom of a more specific OpenSSL error that is also logged (for example, `bad hostname lookup`).


© 2020 NetApp, Inc. All Rights Reserved.

- `BIO_get_host_ip: bad hostname lookup`
  - This message indicates that the node could not connect to the EKS. Make sure that the host name/port is correct, and that the key server is reachable.
- `ssl3_read_bytes:tlsv1 alert unknown ca`
  - This message indicates that the EKS does not trust the CA used to sign the SF client certificate.
  - More likely: SF might be configured with the correct wrong certificate.
  - Less likely: The EKS is not configured to trust the correct CAs.
  - `ssl3_get_server_certificate:certificate verify failed` occurs for different low-level reasons, but has the same implications. This message indicates that the configured TLS client key does not match the configured client certificate. It is highly unlikely that this message will appear.

## 10.4 Key Server Configuration

When you're debugging communication issues between the cluster and the EKS, it can be useful to run `TestKeyServerKmip` or `openssl s_client` commands. To do so, you need the TLS client private key, client certificate, and EKS CA root certificate. You can extract them from Zookeeper by using the `ListDatabaseChildren` API or `zkcli` CLI tool (the certificates also appear in logs). The TLS client key pair is located at `DBPaths::KeyPairDefault` (in the `/keymanagement/keypairs/default` directory).

```
openssl s_client -tls1_2 -connect <key server address:port> -cert client.crt -key client.key -
CAfile CA.pem -nbio -state
```

**Note:** You are copying the secret client key on the Linux system. The client key **must** be deleted after the debugging. Possession of this key combined with access to the cluster allows access to the cluster's data.

## 10.5 Certificate Formatting

When you provide certificates as API parameters, a particular format is required. Certificates should start with `-----BEGIN CERTIFICATE-----` and end with `-----END CERTIFICATE-----`, each on its own line. The Base64-encoded text in between should include only:

- Uppercase and lowercase Roman alphabet characters (A - Z, a - z)
- Numerals (0 through 9)
- New lines
- These symbols: + / =

## 10.6 EKS Debugging Tips

- Ping the key server to check that the EKS is alive and reachable through at least one of the configured IP addresses or host names.
- Follow the corrective action in the cluster faults (if any).
- Check `sf-master.info` and `sf-master.error` log files for any error messages and KMIP communication failures. If there are invalid client or CA certificates, they will be captured in the logs to allow double-checking.
- Check the logs and configuration on the EKS itself.

# Appendix A: Terminology

**AK**: Authentication key. The key used to unlock the encryption/decryption on a drive. The AK, which is associated with a unique key ID, is stored on an EKS.


© 2020 NetApp, Inc. All Rights Reserved.

**CA**: Certificate authority. An entity that issues digital certificates.

**CSR**: Certificate signing request.

**Data encryption key (DEK)**: A 256-bit AES key identified by a universal unique identifier, or UUID (`dataEncryptionKeyID`) that is suitable as a cryptographically secure encryption key for software encryption.

**EAR**: Encryption at rest. This feature uses SEDs to protect customer data stored on the SSDs of a SolidFire node. “At rest” refers to the fact that the data is encrypted only when the drive is powered off or locked (the default state when it is turned on). Customer data in flight within the system and in memory is not protected by this feature.

**EKM**: External key management.

**EKS**: External key server.

**FIPS**: Federal Information Processing Standard.

**Key provider**: An abstraction to support dramatically different mechanisms for generating, storing, and retrieving AKs and DEKs.

**Key service**: The key service provides secure generation, storage, and retrieval of generic AKs. A common component within the cluster, this service was developed to support encryption at rest with EKM.

**KMIP**: Key Management Interoperability Protocol, an OASIS-defined standard protocol used for the communication between clients and servers. The protocol lets you perform certain management operations on objects stored and maintained by a key management system, primarily symmetric and asymmetric cryptographic keys and digital certificates.

**KMS**: Key management service.

**NIST**: National Institute of Standards and Technology.

**SED**: Self-encrypting drive, a drive that automatically encrypts all data written and decrypts all data read. This encryption is always occurring, has no additional performance impact, and is completely transparent to the user. To protect the data, the user provides a password. This password is used by the drive to encrypt and decrypt the media encryption key itself.

## Where to Find Additional Information

To learn more about the information that is described in this document, review the following documents and websites:

- NetApp Element 12.0 Setup Guide
  http://docs.netapp.com/sfe-120/topic/com.netapp.doc.sfe-sg/home.html
- NetApp Element 12.0 API Reference Guide
  http://docs.netapp.com/sfe-120/index.jsp?topic=%2Fcom.netapp.doc.sfe-api%2Fhome.html
- Key Management Interoperability Protocol Specification Version 2.0
  https://docs.oasis-open.org/kmip/kmip-spec/v2.0/csd01/kmip-spec-v2.0-csd01.html

## Version History

| Version | Date | Document Version History |
|---|---|---|
| Version 1.0 | June 2020 | Initial release |


© 2020 NetApp, Inc. All Rights Reserved.

Refer to the Interoperability Matrix Tool (IMT) on the NetApp Support site to validate that the exact product and feature versions described in this document are supported for your specific environment. The NetApp IMT defines the product components and versions that can be used to construct configurations that are supported by NetApp. Specific results depend on each customer's installation in accordance with published specifications.

**Copyright Information**

Copyright © 2020 NetApp, Inc. All Rights Reserved. Printed in the U.S. No part of this document covered by copyright may be reproduced in any form or by any means—graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an electronic retrieval system—without prior written permission of the copyright owner.

Software derived from copyrighted NetApp material is subject to the following license and disclaimer:

THIS SOFTWARE IS PROVIDED BY NETAPP "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WHICH ARE HEREBY DISCLAIMED. IN NO EVENT SHALL NETAPP BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

NetApp reserves the right to change any products described herein at any time, and without notice. NetApp assumes no responsibility or liability arising from the use of products described herein, except as expressly agreed to in writing by NetApp. The use or purchase of this product does not convey a license under any patent rights, trademark rights, or any other intellectual property rights of NetApp.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

Data contained herein pertains to a commercial item (as defined in FAR 2.101) and is proprietary to NetApp, Inc. The U.S. Government has a non-exclusive, non-transferrable, non-sublicensable, worldwide, limited irrevocable license to use the Data only in connection with and in support of the U.S. Government contract under which the Data was delivered. Except as provided herein, the Data may not be used, disclosed, reproduced, modified, performed, or displayed without the prior written approval of NetApp, Inc. United States Government license rights for the Department of Defense are limited to those rights identified in DFARS clause 252.227-7015(b).

**Trademark Information**

NETAPP, the NETAPP logo, and the marks listed at http://www.netapp.com/TM are trademarks of NetApp, Inc. Other company and product names may be trademarks of their respective owners.

TR-4843-0620