Security API methods

Element Software

NetApp
March 05, 2025

Table of Contents

Security API methods

You can integrate Element software with external security-related services, such as an external key management server. These security-related methods enable you to configure Element security features such as external key management for Encryption at Rest.

- AddKeyServerToProviderKmip
- CreateKeyProviderKmip
- CreateKeyServerKmip
- CreatePublicPrivateKeyPair
- DeleteKeyProviderKmip
- DeleteKeyServerKmip
- DisableEncryptionAtRest
- EnableEncryptionAtRest
- GetClientCertificateSignRequest
- GetKeyProviderKmip
- GetKeyServerKmip
- ListKeyProvidersKmip
- ListKeyServersKmip
- ModifyKeyServerKmip
- RemoveKeyServerFromProviderKmip
- SignSshKeys
- TestKeyProviderKmip
- TestKeyServerKmip

Find more information

- SolidFire and Element Software Documentation
- Documentation for earlier versions of NetApp SolidFire and Element products

AddKeyServerToProviderKmip

You can use the `AddKeyServerToProviderKmip` method to assign a Key Management Interoperability Protocol (KMIP) key server to the specified key provider. During assignment, the server is contacted to verify functionality. If the specified key server is already assigned to the specified key provider, no action is taken and no error is returned. You can remove the assignment using the `RemoveKeyServerFromProviderKmip` method.

Parameters

This method has the following input parameters:

Name	Description	Type	Default value	Required
keyProviderID	The ID of the key provider to assign the key server to.	integer	None	Yes
keyServerID	The ID of the key server to assign.	integer	None	Yes

Return values

This method has no return value. The assignment is considered successful as long as there is no error returned.

Request example

Requests for this method are similar to the following example:

```
{
  "method": "AddKeyServerToProviderKmip",
  "params": {
    "keyProviderID": 1,
    "keyServerID": 15
    },
"id": 1
}
```

Response example

This method returns a response similar to the following example:

```
{
   "id": 1,
   "result":
      {}
    }
}
```

New since version

11.7

CreateKeyProviderKmip

You can use the `CreateKeyProviderKmip` method to create a Key Management Interoperability Protocol (KMIP) key provider with the specified name. A key provider defines a mechanism and location to retrieve authentication keys. When you create a new KMIP key provider, it does not have any KMIP key servers assigned to it. To create a KMIP key server, use the `CreateKeyServerKmip` method. To assign it to a provider, see `AddKeyServerToProviderKmip`.

Parameters

This method has the following input parameters:

Name	Description	Type	Default value	Required
keyProviderName	The name to associate with the created KMIP key provider. This name is only used for display purposes and does not need to be unique.	string	None	Yes

Return values

This method has the following return values:

Name	Description	Type
kmipKeyProvider	An object containing details about the newly created key provider.	KeyProviderKmip

Request example

Requests for this method are similar to the following example:

```
{
  "method": "CreateKeyProviderKmip",
  "params": {
    "keyProviderName": "ProviderName",
    },
"id": 1
}
```

Response example

This method returns a response similar to the following example:

```
{
   "id": 1,
   "result":
      {
    "kmipKeyProvider": {
        "keyProviderName": "ProviderName",
        "keyProviderIsActive": true,
        "kmipCapabilities": "SSL",
        "keyServerIDs": [
            15
        ],
        "keyProviderID": 1
    }
}
}
```

New since version

11.7

CreateKeyServerKmip

You can use the `CreateKeyServerKmip` method to create a Key Management Interoperability Protocol (KMIP) key server with the specified attributes. During creation, the server is not contacted; it does not need to exist before you use this method. For clustered key server configurations, you must provide the hostnames or IP addresses of all server nodes in the kmipKeyServerHostnames parameter. You can use the `TestKeyServerKmip` method to test a key server.

Parameters

This method has the following input parameters:

Name	Description	Type	Default value	Required
kmipCaCertificate	The public key certificate of the external key server's root CA. This will be used to verify the certificate presented by external key server in the TLS communication. For key server clusters where individual servers use different CAs, provide a concatenated string containing the root certificates of all the CAs.	string	None	Yes
kmipClientCertificate	A PEM format Base64 encoded PKCS#10 X.509 certificate used by the Solidfire KMIP client.	string	None	Yes
kmipKeyServerHostnames	Array of the hostnames or IP addresses associated with this KMIP key server. Multiple hostnames or IP addresses must only be provided if the key servers are in a clustered configuration.	string array	None	Yes
kmipKeyServerName	The name of the KMIP key server. This name is only used for display purposes and does not need to be unique.	string	None	Yes
kmipKeyServerPort	The port number associated with this KMIP key server (typically 5696).	integer	None	No

Return values

This method has the following return values:

Name	Description	Type
kmipKeyServer	An object containing details about the newly created key server.	KeyServerKmip

Request example

Requests for this method are similar to the following example:

```
{
  "method": "CreateKeyServerKmip",
  "params": {
    "kmipCaCertificate": "MIICPDCCAaUCEDyRMcsf9tAbDpq40ES/E...",
    "kmipClientCertificate": "dKkkirWmnWXbj9T/UWZYB2oK0z5...",
    "kmipKeyServerHostnames" : ["server1.hostname.com",
"server2.hostname.com"],
    "kmipKeyServerName" : "keyserverName",
    "kmipKeyServerPort" : 5696
  },
"id": 1
}
```

Response example

This method returns a response similar to the following example:

```
{
   "id": 1,
   "result":
      {
        "kmipKeyServer": {
            "kmipCaCertificate":"MIICPDCCAaUCEDyRMcsf9tAbDpq40ES/E...",
            "kmipKeyServerHostnames":[
                "server1.hostname.com", "server2.hostname.com"
            ],
            "keyProviderID":1,
            "kmipKeyServerName":"keyserverName",
            "keyServerID":1
            "kmipKeyServerPort":1,
            "kmipClientCertificate":"dKkkirWmnWXbj9T/UWZYB2oK0z5...",
            "kmipAssignedProviderIsActive":true
        }
     }
}
```


New since version

11.7

CreatePublicPrivateKeyPair

You can use the `CreatePublicPrivateKeyPair` method to create public and private SSL keys. You can use these keys to generate certificate signing requests. There can only be one key pair in use for each storage cluster. Before using this method to replacing existing keys, ensure the keys are no longer in use by any providers.

Parameters

This method has the following input parameters:

Name	Description	Type	Default value	Required
commonName	The X.509 distinguished name **Common Name** field (CN).	string	None	No
country	The X.509 distinguished name **Country** field ©.	string	None	No

Name	Description	Type	Default value	Required
emailAddress	The X.509 distinguished name **Email Address** field (MAIL).	string	None	No
locality	The X.509 distinguished name **Locality Name** field (L).	string	None	No
organization	The X.509 distinguished name **Organization Name** field (O).	string	None	No
organizationalUnit	The X.509 distinguished name **Organizational Unit Name** field (OU).	string	None	No
state	The X.509 distinguished name **State** or **Province Name** field (ST or SP or S).	string	None	No

Return values

This method has no return values. If there is no error, key creation is considered successful.

Request example

Requests for this method are similar to the following example:

```
{
  "method": "CreatePublicPrivateKeyPair",
  "params": {
    "commonName": "Name",
    "country": "US",
    "emailAddress" : "email@domain.com"
   },
"id": 1
}
```

Response example

This method returns a response similar to the following example:

```
{
   "id": 1,
   "result":
      {}
}
```

New since version

11.7

DeleteKeyProviderKmip

You can use the `DeleteKeyProviderKmip` method to delete the specified inactive Key Management Interoperability Protocol (KMIP) key provider.

Parameters

This method has the following input parameters:

Name	Description	Type	Default value	Required
keyProviderID	The ID of the key provider to delete.	integer	None	Yes

Return values

This method has no return values. The delete operation is considered successful as long is there is no error.

Request example

Requests for this method are similar to the following example:

```
{
  "method": "DeleteKeyProviderKmip",
  "params": {
    "keyProviderID": "1"
    },
"id": 1
}
```

Response example

This method returns a response similar to the following example:

```
{
   "id": 1,
   "result":
      {}
}
```

New since version

11.7

DeleteKeyServerKmip

You can use the `DeleteKeyServerKmip` method to delete an existing Key Management Interoperability Protocol (KMIP) key server. You can delete a key server unless it is the last one assigned to its provider, and that provider is providing keys which are currently in use.

Parameters

This method has the following input parameters:

Name	Description	Type	Default value	Required
keyServerID	The ID of the KMIP key server to delete.	integer	None	Yes

Return values

This method has the no return values. The delete operation is considered successful if there are no errors.

Request example

Requests for this method are similar to the following example:

```
{
  "method": "DeleteKeyServerKmip",
  "params": {
     "keyServerID": 15
  },
"id": 1
}
```

Response example

This method returns a response similar to the following example:

```
{
   "id": 1,
   "result":
      {}
}
```

New since version

11.7

DisableEncryptionAtRest

You can use the `DisableEncryptionAtRest` method to remove the encryption that was previously applied to the cluster using the `EnableEncryptionAtRest` method. This disable method is asynchronous and returns a response before encryption is disabled. You can use the `GetClusterInfo` method to poll the system to see when the process has completed.

To see the current status of encryption at rest and/or software encryption at rest on the cluster, use the get cluster info method. You can use the `GetSoftwareEncryptionAtRestInfo` method to get information the cluster uses to encrypt data at rest.

You cannot use this method to disable software encryption at rest. To disable software encryption at rest, you need to create a new cluster with software encryption at rest disabled.

Parameters

This method has no input parameters.

Return values

This method has no return values.

Request example

Requests for this method are similar to the following example:

```
{
   "method": "DisableEncryptionAtRest",
   "params": {},
   "id": 1
}
```

Response example

This method returns a response similar to the following example:

```
{
   "id" : 1,
  "result" : {}
}
```

New since version

9.6

Find more information

- GetClusterInfo
- SolidFire and Element Software Documentation
- Documentation for earlier versions of NetApp SolidFire and Element products

EnableEncryptionAtRest

You can use the `EnableEncryptionAtRest` method to enable the Advanced Encryption Standard (AES) 256-bit encryption at rest on the cluster so that the cluster can manage the encryption key used for the drives on each node. This feature is not enabled by default.

To see the current status of encryption at rest and/or software encryption at rest on the cluster, use the get cluster info method. You can use the `GetSoftwareEncryptionAtRestInfo` method to get information the cluster uses to encrypt data at rest.

This method does not enable software encryption at rest. This can only be done using the create cluster method with `enableSoftwareEncryptionAtRest` set to `true`.

When you enable encryption at rest, the cluster automatically manages encryption keys internally for the drives on each node in the cluster.

If a keyProviderID is specified, the password is generated and retrieved according to the type of key provider. This is usually done using a Key Management Interoperability Protocol (KMIP) key server in the case of a KMIP key provider. After this operation, the specified provider is considered active and cannot be deleted until

Encryption at Rest is disabled using the `DisableEncryptionAtRest` method.

If you have a node type with a model number ending in "-NE", the `EnableEncryptionAtRest` method call will fail with a response of "Encryption not allowed. Cluster detected non-encryptable node".

You should only enable or disable encryption when the cluster is running and in a healthy state. You can enable or disable encryption at your discretion and as often as you need.

This process is asynchronous and returns a response before encryption is enabled. You can use the `GetClusterInfo` method to poll the system to see when the process has completed.

Parameters

This method has the following input parameters:

Name	Description	Type	Default value	Required
keyProviderID	The ID of a KMIP key provider to use.	integer	None	No

Return values

This method has no return values.

Request example

Requests for this method are similar to the following example:

```
{
   "method": "EnableEncryptionAtRest",
   "params": {},
   "id": 1
}
```

Response examples

This method returns a response similar to the following example from the EnableEncryptionAtRest method. There is no result to report.

```
{
   "id": 1,
   "result": {}
}
```

While Encryption At Rest is being enabled on a cluster, GetClusterInfo returns a result describing the state of

Encryption at Rest ("encryptionAtRestState") as "enabling". After Encryption at Rest is fully enabled, the returned state changes to "enabled".


```
{
   "id": 1,
      "result": {
         "clusterInfo": {
            "attributes": { },
               "encryptionAtRestState": "enabling",
            "ensemble": [
               "10.10.5.94",
               "10.10.5.107",
               "10.10.5.108"
            ],
            "mvip": "192.168.138.209",
            "mvipNodeID": 1,
            "name": "Marshall",
            "repCount": 2,
            "svip": "10.10.7.209",
            "svipNodeID": 1,
            "uniqueID": "91dt"
      }
   }
}
```


New since version

9.6

Find more information

- SecureEraseDrives
- GetClusterInfo
- SolidFire and Element Software Documentation
- Documentation for earlier versions of NetApp SolidFire and Element products

GetClientCertificateSignRequest

You can use the `GetClientCertificateSignRequest` method to generate a certificate signing request that can be signed by a certificate authority to generate a client certificate for the cluster. Signed certificates are needed to establish a trust relationship for interacting with external services.

Parameters

This method has no input parameters.

Return values

This method has the following return values:

Name	Description	Type
clientCertificateSignRequest	A PEM format Base64 encoded PKCS#10 X.509 client certificate sign request.	string

Request example

Requests for this method are similar to the following example:

```
{
  "method": "GetClientCertificateSignRequest",
  "params": {
  },
"id": 1
}
```

Response example

This method returns a response similar to the following example:

```
{
   "id": 1,
   "result":
      {
        "clientCertificateSignRequest":
"MIIByjCCATMCAQAwgYkxCzAJBgNVBAYTAlVTMRMwEQYDVQQIEwpDYWxpZm9ybm..."
   }
}
```

New since version

11.7

GetKeyProviderKmip

You can use the `GetKeyProviderKmip` method to retrieve information about the specified Key Management Interoperability Protocol (KMIP) key provider.

Parameters

This method has the following input parameters:

Name	Description	Type	Default value	Required
keyProviderID	The ID of the KMIP key provider object to return.	integer	None	Yes

Return values

This method has the following return values:

Name	Description	Type
kmipKeyProvider	An object containing details about the requested key provider.	KeyProviderKmip

Request example

Requests for this method are similar to the following example:

```
{
  "method": "GetKeyProviderKmip",
  "params": {
    "keyProviderID": 15
    },
"id": 1
}
```

Response example

This method returns a response similar to the following example:

```
{
   "id": 1,
   "result":
    {
    "kmipKeyProvider": {
        "keyProviderID": 15,
        "kmipCapabilities": "SSL",
        "keyProviderIsActive": true,
        "keyServerIDs": [
            1
        ],
        "keyProviderName": "ProviderName"
    }
   }
}
```

New since version

11.7

GetKeyServerKmip

You can use the `GetKeyServerKmip` method to return information about the specified Key Management Interoperability Protocol (KMIP) key server.

Parameters

This method has the following input parameters:

Name	Description	Type	Default value	Required
keyServerID	The ID of the KMIP key server to return information about.	integer	None	Yes

Return values

This method has the following return values:

Name	Description	Type
kmipKeyServer	An object containing details about the requested key server.	KeyServerKmip

Request example

Requests for this method are similar to the following example:

```
{
  "method": "GetKeyServerKmip",
  "params": {
     "keyServerID": 15
  },
"id": 1
}
```

Response example

This method returns a response similar to the following example:

```
{
   "id": 1,
   "result":
      {
        "kmipKeyServer": {
            "kmipCaCertificate":"MIICPDCCAaUCEDyRMcsf9tAbDpq40ES/E...",
            "kmipKeyServerHostnames":[
                "server1.hostname.com", "server2.hostname.com"
            ],
            "keyProviderID":1,
            "kmipKeyServerName":"keyserverName",
            "keyServerID":15
            "kmipKeyServerPort":1,
            "kmipClientCertificate":"dKkkirWmnWXbj9T/UWZYB2oK0z5...",
            "kmipAssignedProviderIsActive":true
        }
    }
}
```

New since version

11.7

GetSoftwareEncryptionAtRestInfo

You can use the `GetSoftwareEncryptionAtRestInfo` method to get software encryption-at-rest information the cluster uses to encrypt data at rest.

Parameters

This method has no input parameters.

Return values

This method has the following return values:

Parameter	Description	Type	Optional
masterKeyInfo	Information about the current software encryption-at-rest master key.	EncryptionKeyInfo	True
rekeyMasterKeyAsyncResultID	The async result ID of the current or most recent rekey operation (if any), if it has not been deleted yet. `GetAsyncResult` output will include a `newKey` field that contains information about the new master key and a `keyToDecommission` field that contains information about the old key.	integer	True
state	The current software encryption-at-rest state. Possible values are `disabled` or `enabled`.	string	False
version	A version number that is incremented each time software encryption at rest is enabled.	integer	False

Request example

Requests for this method are similar to the following example:

```
{
 "method": "getsoftwareencryptionatrestinfo"
}
```

Response example

This method returns a response similar to the following example:

```
{
    "id": 1,
    "result": {
        "masterKeyInfo": {
            "keyCreatedTime": "2021-09-20T23:15:56Z",
            "keyID": "4d80a629-a11b-40ab-8b30-d66dd5647cfd",
            "keyManagementType": "internal"
        },
        "state": "enabled",
        "version": 1
    }
}
```


New since version

12.3

Find more information

- SolidFire and Element Software Documentation
- Documentation for earlier versions of NetApp SolidFire and Element products

ListKeyProvidersKmip

You can use the `ListKeyProvidersKmip` method to retrieve a list of all existing Key Management Interoperability Protocol (KMIP) key providers. You can filter the list by specifying additional parameters.

Parameters

This method has the following input parameters:

Name	Description	Type	Default value	Required
keyProviderIsActive	Filters returned KMIP key server objects based on whether they are active. Possible values: • true: Returns only KMIP key providers which are active (providing keys which are currently in use). • false: Returns only KMIP key providers which are inactive (not providing any keys and able to be deleted). If omitted, returned KMIP key providers are not filtered based on whether they are active.	boolean	None	No

Name	Description	Type	Default value	Required
kmipKeyProviderHasServerAssigned	Filters returned KMIP key providers based on whether they have a KMIP key server assigned. Possible values: • true: Returns only KMIP key providers which have a KMIP key server assigned. • false: Returns only KMIP key providers which do not have a KMIP key server assigned. If omitted, returned KMIP key providers are not filtered based on whether they have a KMIP key server assigned.	boolean	None	No

Return values

This method has the following return values:

Name	Description	Type
kmipKeyProviders	A list of KMIP key providers that have been created.	KeyProviderKmip array

Request example

Requests for this method are similar to the following example:

```
{
  "method": "ListKeyProvidersKmip",
  "params": {},
"id": 1
}
```

Response example

This method returns a response similar to the following example:

```
{
   "id": 1,
   "result":
    {
    "kmipKeyProviders": [
        {
            "keyProviderID": 15,
            "kmipCapabilities": "SSL",
            "keyProviderIsActive": true,
            "keyServerIDs": [
                1
            ],
            "keyProviderName": "KeyProvider1"
        }
    ]
}
}
```

New since version

11.7

ListKeyServersKmip

You can use the `ListKeyServersKmip` method to list all Key Management Interoperability Protocol (KMIP) key servers that have been created. You can filter the results by specifying additional parameters.

Parameters

This method has the following input parameters:

Name	Description	Type	Default value	Required
keyProviderID	When specified, the method only returns KMIP key servers that are assigned to the specified KMIP key provider. If omitted, returned KMIP key servers will not be filtered based on whether they are assigned to the specified KMIP Key Provider.	integer	None	No
kmipAssignedProviderIsActive	Filters returned KMIP key server objects based on whether they are active. Possible values: • true: Returns only KMIP key servers which are active (providing keys which are currently in use). • false: Returns only KMIP key servers which are inactive (not providing any keys and able to be deleted). If omitted, returned KMIP key servers are not filtered based on whether they are active.	boolean	None	No

Name	Description	Type	Default value	Required
kmipHasProviderAssigned	Filters returned KMIP key servers based on whether they have a KMIP key provider assigned. Possible values: • true: Returns only KMIP key servers which have a KMIP key provider assigned. • false: Returns only KMIP key servers which do not have a KMIP key provider assigned. If omitted, returned KMIP key servers are not filtered based on whether they have a KMIP key provider assigned.	boolean	None	No

Return values

This method has the following return values:

Name	Description	Type
kmipKeyServers	The complete list of KMIP key servers which have been created.	KeyServerKmip array

Request example

Requests for this method are similar to the following example:

```
{
  "method": "ListKeyServersKmip",
  "params": {},
"id": 1
}
```

Response example

This method returns a response similar to the following example:

```
{
    "kmipKeyServers": [
        {
            "kmipKeyServerName": "keyserverName",
            "kmipClientCertificate": "dKkkirWmnWXbj9T/UWZYB2oK0z5...",
            "keyServerID": 15,
            "kmipAssignedProviderIsActive": true,
            "kmipKeyServerPort": 5696,
            "kmipCaCertificate": "MIICPDCCAaUCEDyRMcsf9tAbDpq40ES/E...",
            "kmipKeyServerHostnames": [
                "server1.hostname.com", "server2.hostname.com"
            ],
            "keyProviderID": 1
        }
    ]
}
```

New since version

11.7

ModifyKeyServerKmip

You can use the `ModifyKeyServerKmip` method to modify an existing Key Management Interoperability Protocol (KMIP) key server to the specified attributes. Although the only required parameter is the keyServerID, a request containing only the keyServerID will take no action and return no error. Any other parameters you specify will replace the existing values for the key server with the specified keyServerID. The key server is contacted during the operation to ensure that it is functional. You can provide multiple hostnames or IP addresses with the kmipKeyServerHostnames parameter, but only if the key servers are in a clustered configuration.

Parameters

This method has the following input parameters:

Name	Description	Type	Default value	Required
keyServerID	The ID of the KMIP Key Server to modify.	integer	None	Yes

kmipCaCertificate	The public key certificate of the external key server's root CA. This will be used to verify the certificate presented by external key server in the TLS communication. For key server clusters where individual servers use different CAs, provide a concatenated string containing the root certificates of all the CAs.	string	None	No
kmipClientCertificate	A PEM format Base64 encoded PKCS#10 X.509 certificate used by the Solidfire KMIP client.	string	None	No
kmipKeyServerHostnames	Array of the hostnames or IP addresses associated with this KMIP key server. Multiple hostnames or IP addresses must only be provided if the key servers are in a clustered configuration.	string array	None	No
kmipKeyServerName	The name of the KMIP key server. This name is only used for display purposes and does not need to be unique.	string	None	No
kmipKeyServerPort	The port number associated with this KMIP key server (typically 5696).	integer	None	No

Return values

This method has the following return values:

Name	Description	Type
kmipKeyServer	An object containing details about the newly modified key server.	KeyServerKmip

Request example

Requests for this method are similar to the following example:

```
{
  "method": "ModifyKeyServerKmip",
  "params": {
    "keyServerID": 15
    "kmipCaCertificate": "CPDCCAaUCEDyRMcsf9tAbDpq40ES/E...",
    "kmipClientCertificate": "kirWmnWXbj9T/UWZYB2oK0z5...",
    "kmipKeyServerHostnames" : ["server1.hostname.com",
"server2.hostname.com"],
    "kmipKeyServerName" : "keyserverName",
    "kmipKeyServerPort" : 5696
  },
"id": 1
}
```

Response example

This method returns a response similar to the following example:

```
{
   "id": 1,
   "result":
      {
        "kmipKeyServer": {
            "kmipCaCertificate":"CPDCCAaUCEDyRMcsf9tAbDpq40ES/E...",
            "kmipKeyServerHostnames":[
                "server1.hostname.com", "server2.hostname.com"
            ],
            "keyProviderID":1,
            "kmipKeyServerName":"keyserverName",
            "keyServerID":1
            "kmipKeyServerPort":1,
            "kmipClientCertificate":"kirWmnWXbj9T/UWZYB2oK0z5...",
            "kmipAssignedProviderIsActive":true
        }
     }
}
```


New since version

11.7

RekeySoftwareEncryptionAtRestMasterKey

You can use the `RekeySoftwareEncryptionAtRestMasterKey` method to rekey the software encryption-at-rest master key used to encrypt DEKs (Data Encryption Keys). During cluster creation, software encryption at rest is configured to use Internal Key Management (IKM). This rekey method can be used after cluster creation to use either IKM or External Key Management (EKM).

Parameters

This method has the following input parameters. If the `keyManagementType` parameter is not specified, the rekey operation is performed using the existing key management configuration. If the `keyManagementType` is specified and the key provider is external, the `keyProviderID` parameter must also be used.

Parameter	Description	Type	Optional
keyManagementType	The type of key management used to manage the master key. Possible values are: `Internal`: Rekey using internal key management. `External`: Rekey using external key management. If this parameter is not specified, the rekey operation is performed using the existing key management configuration.	string	True
keyProviderID	The ID of the key provider to use. This is a unique value returned as part of one of the `CreateKeyProvider` methods. The ID is only required when `keyManagementType` is `External` and is otherwise invalid.	integer	True

Return values

This method has the following return values:

Parameter	Description	Type	Optional
asyncHandle	Determine the status of the rekey operation using this `asyncHandle` value with `GetAsyncResult`. `GetAsyncResult` output will include a `newKey` field that contains information about the new master key and a `keyToDecommission` field that contains information about the old key.	integer	False

Request example

Requests for this method are similar to the following example:

```
{
 "method": "rekeysoftwareencryptionatrestmasterkey",
 "params": {
   "keyManagementType": "external",
   "keyProviderID": "<ID number>"
 }
}
```

Response example

This method returns a response similar to the following example:

```
{
    "asyncHandle": 1
}
```

New since version

12.3

Find more information

- SolidFire and Element Software Documentation
- Documentation for earlier versions of NetApp SolidFire and Element products

RemoveKeyServerFromProviderKmip

You can use the `RemoveKeyServerFromProviderKmip` method to unassign the specified Key Management Interoperability Protocol (KMIP) key server from the provider it was assigned to. You can unassign a key server from its provider unless it is the last one and its provider is active (providing keys which are currently in use). If the specified key server is not assigned to a provider, no action is taken and no error is returned.

Parameters

This method has the following input parameters:

Name	Description	Type	Default value	Required
keyServerID	The ID of the KMIP key server to unassign.	integer	None	Yes

Return values

This method has no return values. The removal is considered successful as long as no error is returned.

Request example

Requests for this method are similar to the following example:

```
{
  "method": "RemoveKeyServerFromProviderKmip",
  "params": {
    "keyServerID": 1
   },
"id": 1
}
```

Response example

This method returns a response similar to the following example:

```
{
   "id": 1,
   "result":
      {}
    }
}
```

New since version

11.7

SignSshKeys

After SSH is enabled on the cluster using the EnableSSH method, you can use the `SignSshKeys` method to gain access to a shell on a node.

Beginning with Element 12.5, `sfreadonly` is a new system account allows for basic troubleshooting on a node. This API enables SSH access using the `sfreadonly` system account across all nodes in the cluster.

> Unless advised by NetApp Support, any alterations to the system are unsupported, voiding your support contract, and may result in instability or inaccessibility of data.

After you use the method, you must copy the keychain from the response, save it to the system that will be initiating the SSH connection, then run the following command:

```
ssh -i <identity_file> sfreadonly@<node_ip>
```

`identity_file` is a file from which the identity (private key) for public key authentication is read and `node_ip` is the IP address of the node. For more information on `identity_file`, see the SSH man page.

Parameters

This method has the following input parameters:

Name	Description	Type	Default value	Required
duration	Integer from 1 to 24 reflecting number of hours for signed key to be valid. If duration is not specified, the default is used.	integer	1	No
publicKey	If provided, this parameter will only return the signed_public_key instead of creating a full keychain to the user. Public keys submitted using the URL bar in a browser with + are interpreted as spaced and breaking signing.	string	Null	No

Name	Description	Type	Default value	Required
sfadmin	Allows access to the sfadmin shell account when you make the API call with supportAdmin cluster access, or when the node is not in a cluster.	boolean	False	No

Return values

This method has the following return values:

Name	Description	Type
keygen_status	Contains the identity in the signed key, the principals allowed, and the valid start and end dates for the key.	string
private_key	A private SSH key value is only returned if the API is generating a complete keychain for the end user.  The value is Base64 encoded; you must decode the value when it is written to a file to ensure that it is read as a valid private key.	string
public_key	A public SSH key value is only returned if the API is generating a complete keychain for the end user.  When you pass a public_key parameter to the API method, only the `signed_public_key` value is returned in the response.	string
signed_public_key	The SSH public key that results from signing the public key, whether this was user provided or generated by API.	string

Request example

Requests for this method are similar to the following example:

```
{
    "method": "SignSshKeys",
    "params": {
        "duration": 2,
        "publicKey":<string>
    },
    "id": 1
}
```

Response example

This method returns a response similar to the following example:

```
{
  "id": null,
  "result": {
    "signedKeys": {
      "keygen_status": <keygen_status>,
      "signed_public_key": <signed_public_key>
    }
  }
}
```

In this example, a public key is signed and returned that is valid for the duration (1-24 hours).

New since version

12.5

TestKeyProviderKmip

You can use the `TestKeyProviderKmip` method to test whether the specified Key Management Interoperability Protocol (KMIP) key provider is reachable and functioning normally.

Parameters

This method has the following input parameters:

Name	Description	Type	Default value	Required
keyProviderID	The ID of the key provider to test.	integer	None	Yes

Return values

This method has no return values. The test is considered successful as long as no error is returned.

Request example

Requests for this method are similar to the following example:

```
{
  "method": "TestKeyProviderKmip",
  "params": {
    "keyProviderID": 15
  },
"id": 1
}
```

Response example

This method returns a response similar to the following example:

```
{
   "id": 1,
   "result":
      {}
    }
}
```

New since version

11.7

TestKeyServerKmip

You can use the `TestKeyServerKmip` method to test whether the specified Key Management Interoperability Protocol (KMIP) key server is reachable and functioning normally.

Parameters

This method has the following input parameters:

Name	Description	Type	Default value	Required
keyServerID	The ID of the KMIP key server to test.	integer	None	Yes

Return values

This method has no return values. The test is considered successful if there are no errors returned.

Request example

Requests for this method are similar to the following example:

```
{
  "method": "TestKeyServerKmip",
  "params": {
    "keyServerID": 15
  },
"id": 1
}
```

Response example

This method returns a response similar to the following example:

```
{
   "id": 1,
   "result":
      {}
    }
```

New since version

11.7

Copyright information

Trademark information