Managing APIC Using the REST API

Adding Management Access

In-Band and Out-of-Band Management Access

The mgmt tenant provides a convenient means to configure access to fabric management functions. While fabric management functions are accessible through the APIC, they can also be accessed directly through in-band and out-of-band network policies.

Static and Dynamic Management Access

APIC supports both static and dynamic management access. For simple deployments where users manage the IP addresses of a few leaf and spine switches, configuring static in-band and out-of-band management connectivity is simpler. For more complex deployments, where you might have a large number of leaf and spine switches that require managing many IP addresses, static management access is not recommended. For detailed information about static management access, see *Cisco APIC and Static Management Access*.

About Static Management Access

Configuring static in-band and out-of-band management connectivity is simpler than configuring dynamic in-band and out-of-band management connectivity. When configuring in-band static management, you must specify the IP address for each node and make sure to assign unique IP addresses. For simple deployments where users manage the IP addresses of a few leaf and spine switches, it is easy to configure a static management access. For more complex deployments, where you might have a large number of leaf and spine switches that require managing many IP addresses, static management access is not recommended. We recommend that you configure a dynamic management access that automatically avoids the possible duplication of IP addresses.

Configuring In-Band Management Access Using the REST API

IPv4 and IPv6 addresses are supported for in-band management access. IPv6 configurations are supported using static configurations (for both in-band and out-of-band). IPv4 and IPv6 dual in-band and out-of-band configurations are supported only through static configuration. For more information, see the KB article,*Configuring Static Management Access in Cisco APIC*.

SUMMARY STEPS

1. Create a VLAN namespace.
2. Create a physical domain.
3. Create selectors for the in-band management.
4. Configure an in-band bridge domain and endpoint group (EPG).
5. Create an address pool.
6. Create management groups.

DETAILED STEPS

Step 1 Create a VLAN namespace.

Example:

```
POST
https://apic-ip-address/api/mo/uni.xml

<?xml version="1.0" encoding="UTF-8"?>
<!-- api/policymgr/mo/uni.xml -->
<polUni>
  <infraInfra>
    <!-- Static VLAN range -->
    <fvnsVlanInstP name="inband" allocMode="static">
      <fvnsEncapBlk name="encap" from="vlan-10" to="vlan-11"/>
    </fvnsVlanInstP>
  </infraInfra>
</polUni>
```

Step 2 Create a physical domain.

Example:

```
POST
https://apic-ip-address/api/mo/uni.xml

<?xml version="1.0" encoding="UTF-8"?>
<!-- api/policymgr/mo/uni.xml -->
<polUni>
  <physDomP name="inband">
    <infraRsVlanNs tDn="uni/infra/vlanns-inband-static"/>
  </physDomP>
</polUni>
```

Step 3 Create selectors for the in-band management.

Example:

```
POST
https://apic-ip-address/api/mo/uni.xml

<?xml version="1.0" encoding="UTF-8"?>
```

```
<!-- api/policymgr/mo/.xml -->
<polUni>
  <infraInfra>
    <infraNodeP name="vmmNodes">
      <infraLeafS name="leafS" type="range">
        <infraNodeBlk name="single0" from_="101" to_="101"/>
      </infraLeafS>
      <infraRsAccPortP tDn="uni/infra/accportprof-vmmPorts"/>
    </infraNodeP>

    <!-- Assumption is that VMM host is reachable via eth1/40. -->
    <infraAccPortP name="vmmPorts">
      <infraHPortS name="portS" type="range">
        <infraPortBlk name="block1"
                      fromCard="1" toCard="1"
                      fromPort="40" toPort="40"/>
        <infraRsAccBaseGrp tDn="uni/infra/funcprof/accportgrp-inband" />
      </infraHPortS>
    </infraAccPortP>


    <infraNodeP name="apicConnectedNodes">
      <infraLeafS name="leafS" type="range">
        <infraNodeBlk name="single0" from_="101" to_="102"/>
      </infraLeafS>
      <infraRsAccPortP tDn="uni/infra/accportprof-apicConnectedPorts"/>
    </infraNodeP>

    <!-- Assumption is that APIC is connected to eth1/1. -->
    <infraAccPortP name="apicConnectedPorts">
      <infraHPortS name="portS" type="range">
        <infraPortBlk name="block1"
                      fromCard="1" toCard="1"
                      fromPort="1" toPort="3"/>
        <infraRsAccBaseGrp tDn="uni/infra/funcprof/accportgrp-inband" />
      </infraHPortS>
    </infraAccPortP>

    <infraFuncP>
      <infraAccPortGrp name="inband">
        <infraRsAttEntP tDn="uni/infra/attentp-inband"/>
      </infraAccPortGrp>
    </infraFuncP>

    <infraAttEntityP name="inband">
      <infraRsDomP tDn="uni/phys-inband"/>
    </infraAttEntityP>
  </infraInfra>
</polUni>
```

Step 4 Configure an in-band bridge domain and endpoint group (EPG).

Example:

```
POST https://apic-ip-address/api/mo/uni.xml

<?xml version="1.0" encoding="UTF-8"?>
<!-- api/policymgr/mo/.xml -->
<polUni>
  <fvTenant name="mgmt">
    <!-- Configure the in-band management gateway address on the
         in-band BD. -->
    <fvBD name="inb">
      <fvSubnet ip="10.13.1.254/24"/>
    </fvBD>
```

```
    <mgmtMgmtP name="default">
      <!-- Configure the encap on which APICs will communicate on the
           in-band network. -->
      <mgmtInB name="default" encap="vlan-10">
        <fvRsProv tnVzBrCPName="default"/>
      </mgmtInB>
    </mgmtMgmtP>
  </fvTenant>
</polUni>
```

Step 5 Create an address pool.

Example:

```
POST
https://apic-ip-address/api/mo/uni.xml

<?xml version="1.0" encoding="UTF-8"?>
<!-- api/policymgr/mo/.xml -->
<polUni>
  <fvTenant name="mgmt">
    <!-- Adresses for APIC in-band management network -->
    <fvnsAddrInst name="apicInb" addr="10.13.1.254/24">
      <fvnsUcastAddrBlk from="10.13.1.1" to="10.13.1.10"/>
    </fvnsAddrInst>

    <!-- Adresses for switch in-band management network -->
    <fvnsAddrInst name="switchInb" addr="10.13.1.254/24">
      <fvnsUcastAddrBlk from="10.13.1.101" to="10.13.1.120"/>
    </fvnsAddrInst>
  </fvTenant>
</polUni>
```

Note Dynamic address pools for IPv6 is not supported.

Step 6 Create management groups.

Example:

```
POST
https://apic-ip-address/api/mo/uni.xml

<?xml version="1.0" encoding="UTF-8"?>
<!-- api/policymgr/mo/.xml -->
<polUni>
  <infraInfra>
    <!-- Management node group for APICs -->
    <mgmtNodeGrp name="apic">
      <infraNodeBlk name="all" from_="1" to_="3"/>
      <mgmtRsGrp tDn="uni/infra/funcprof/grp-apic"/>
    </mgmtNodeGrp>

    <!-- Management node group for switches-->
    <mgmtNodeGrp name="switch">
      <infraNodeBlk name="all" from_="101" to_="104"/>
      <mgmtRsGrp tDn="uni/infra/funcprof/grp-switch"/>
    </mgmtNodeGrp>

    <!-- Functional profile -->
    <infraFuncP>
      <!-- Management group for APICs -->
      <mgmtGrp name="apic">
        <!-- In-band management zone -->
        <mgmtInBZone name="default">
```

```
        <mgmtRsInbEpg tDn="uni/tn-mgmt/mgmtp-default/inb-default"/>
        <mgmtRsAddrInst tDn="uni/tn-mgmt/addrinst-apicInb"/>
      </mgmtInBZone>
    </mgmtGrp>

    <!-- Management group for switches -->
    <mgmtGrp name="switch">
      <!-- In-band management zone -->
      <mgmtInBZone name="default">
        <mgmtRsInbEpg tDn="uni/tn-mgmt/mgmtp-default/inb-default"/>
        <mgmtRsAddrInst tDn="uni/tn-mgmt/addrinst-switchInb"/>
      </mgmtInBZone>
    </mgmtGrp>
  </infraFuncP>
 </infraInfra>
</polUni>
```

Note Dynamic address pools for IPv6 is not supported.

Configuring Static In-Band Management Access Using the REST API

Step 1 Create a VLAN namespace.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- api/policymgr/mo/uni.xml -->
<polUni>
  <infraInfra>
    <!-- Static VLAN range -->
    <fvnsVlanInstP name="inband" allocMode="static">
      <fvnsEncapBlk name="encap" from="vlan-10" to="vlan-11"/>
    </fvnsVlanInstP>
  </infraInfra>
</polUni>
```

Step 2 Create a physical domain.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- api/policymgr/mo/uni.xml -->
<polUni>
  <physDomP name="inband">
    <infraRsVlanNs tDn="uni/infra/vlanns-inband-static"/>
  </physDomP>
</polUni>
```

Step 3 Create selectors for the in-band management.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- api/policymgr/mo/.xml -->
<polUni>
  <infraInfra>
    <infraNodeP name="vmmNodes">
      <infraLeafS name="leafS" type="range">
        <infraNodeBlk name="single0" from_="101" to_="101"/>
```

```
    </infraLeafS>
    <infraRsAccPortP tDn="uni/infra/accportprof-vmmPorts"/>
  </infraNodeP>

  <!-- Assumption is that VMM host is reachable via eth1/40. -->
  <infraAccPortP name="vmmPorts">
    <infraHPortS name="portS" type="range">
      <infraPortBlk name="block1"
                    fromCard="1" toCard="1"
                    fromPort="40" toPort="40"/>
      <infraRsAccBaseGrp tDn="uni/infra/funcprof/accportgrp-inband" />
    </infraHPortS>
  </infraAccPortP>


  <infraNodeP name="apicConnectedNodes">
    <infraLeafS name="leafS" type="range">
      <infraNodeBlk name="single0" from_="101" to_="102"/>
    </infraLeafS>
    <infraRsAccPortP tDn="uni/infra/accportprof-apicConnectedPorts"/>
  </infraNodeP>

  <!-- Assumption is that APIC is connected to eth1/1. -->
  <infraAccPortP name="apicConnectedPorts">
    <infraHPortS name="portS" type="range">
      <infraPortBlk name="block1"
                    fromCard="1" toCard="1"
                    fromPort="1" toPort="3"/>
      <infraRsAccBaseGrp tDn="uni/infra/funcprof/accportgrp-inband" />
    </infraHPortS>
  </infraAccPortP>

  <infraFuncP>
    <infraAccPortGrp name="inband">
      <infraRsAttEntP tDn="uni/infra/attentp-inband"/>
    </infraAccPortGrp>
  </infraFuncP>

  <infraAttEntityP name="inband">
    <infraRsDomP tDn="uni/phys-inband"/>
  </infraAttEntityP>
 </infraInfra>
</polUni>
```

Step 4 Configure an in-band bridge domain and endpoint group (EPG).

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- api/policymgr/mo/.xml -->
<polUni>
  <fvTenant name="mgmt">
    <!-- Configure the in-band management gateway address on the
         in-band BD. -->
    <fvBD name="inb">
      <fvSubnet ip="<subnet_ip_address>"/>
    </fvBD>

    <mgmtMgmtP name="default">
      <!-- Configure the encap on which APICs will communicate on the
           in-band network. -->
      <mgmtInB name="default" encap="vlan-10">
        <fvRsProv tnVzBrCPName="default"/>
      </mgmtInB>
    </mgmtMgmtP>
```

```
  </fvTenant>
</polUni>
```

Step 5 Create static in-band management IP addresses and assign them to node IDs.

Example:

```
<polUni>
  <fvTenant name="mgmt">
    <mgmtMgmtP name="default">
      <mgmtInB name="default">
        <mgmtRsInBStNode tDn="topology/pod-1/node-101"
                         addr="<ip_address_1>"
                         gw="<gw_address>”
    v6Addr = “<ip6_address_1>”
    v6Gw = “<ip6_gw_address>"/>
        <mgmtRsInBStNode tDn="topology/pod-1/node-102"
                         addr="<ip_address_2>"
                         gw="<gw_address>”
    v6Addr = “<ip6_address_2>"
    v6Gw = “<ip6_gw_address>"/>
        <mgmtRsInBStNode tDn="topology/pod-1/node-103"
                         addr="<ip_address_3>"
                         gw="<gw_address>”
    v6Addr = “<ip6_address_3>"
    v6Gw = “<ip6_gw_address>"/>
        <mgmtRsInBStNode tDn="topology/pod-1/node-104"
                         addr="<ip_address_4>"
                         gw="<gw_address>”
    v6Addr = “<ip6_address_4>"
    v6Gw = “<ip6_gw_address>"/>

        <mgmtRsInBStNode tDn="topology/pod-1/node-105"
                         addr="<ip_address_5>"
                         gw="<gw_address>”
    v6Addr = “<ip6_address_5>"
    v6Gw = “<ip6_gw_address>"/>

      </mgmtInB>
    </mgmtMgmtP>
  </fvTenant>
</polUni>
```

Configuring Out-of-Band Management Access Using the REST API

IPv4 and IPv6 addresses are supported for out-of-band management access.

Before you begin

The APIC out-of-band management connection link must be 1 Gbps.

SUMMARY STEPS

1. Create an out-of-band contract.
2. Associate the out-of-band contract with an out-of-band EPG.
3. Associate the out-of-band contract with an external management EPG.
4. Create a management address pool.

5. Create node management groups.

DETAILED STEPS

Step 1 Create an out-of-band contract.

Example:

```
POST https://apic-ip-address/api/mo/uni.xml

<polUni>
    <fvTenant name="mgmt">
        <!-- Contract -->
        <vzOOBBrCP name="oob-default">
            <vzSubj name="oob-default">
                <vzRsSubjFiltAtt tnVzFilterName="default" />
            </vzSubj>
        </vzOOBBrCP>
    </fvTenant>
</polUni>
```

Step 2 Associate the out-of-band contract with an out-of-band EPG.

Example:

```
POST https://apic-ip-address/api/mo/uni.xml

<polUni>
    <fvTenant name="mgmt">
        <mgmtMgmtP name="default">
            <mgmtOoB name="default">
                <mgmtRsOoBProv tnVzOOBBrCPName="oob-default" />
            </mgmtOoB>
        </mgmtMgmtP>
    </fvTenant>
</polUni>
```

Step 3 Associate the out-of-band contract with an external management EPG.

Example:

```
POST https://apic-ip-address/api/mo/uni.xml

<polUni>
    <fvTenant name="mgmt">
        <mgmtExtMgmtEntity name="default">
            <mgmtInstP name="oob-mgmt-ext">
                <mgmtRsOoBCons tnVzOOBBrCPName="oob-default" />
                <!-- SUBNET from where switches are managed -->
                <mgmtSubnet ip="10.0.0.0/8" />
            </mgmtInstP>
        </mgmtExtMgmtEntity>
    </fvTenant>
</polUni>
```

Step 4 Create a management address pool.

Example:

```
POST https://apic-ip-address/api/mo/uni.xml
```

```
<polUni>
    <fvTenant name="mgmt">
        <fvnsAddrInst name="switchOoboobaddr" addr="172.23.48.1/21">
            <fvnsUcastAddrBlk from="172.23.49.240" to="172.23.49.244"/>
        </fvnsAddrInst>
    </fvTenant>
</polUni>
```

Step 5 Create node management groups.

Example:

```
POST https://apic-ip-address/api/mo/uni.xml

<polUni>
    <infraInfra>
        <infraFuncP>
            <mgmtGrp name="switchOob">
                <mgmtOoBZone name="default">
                    <mgmtRsAddrInst tDn="uni/tn-mgmt/addrinst-switchOoboobaddr" />
                    <mgmtRsOobEpg tDn="uni/tn-mgmt/mgmtp-default/oob-default" />
                </mgmtOoBZone>
            </mgmtGrp>
        </infraFuncP>
        <mgmtNodeGrp name="switchOob">
            <mgmtRsGrp tDn="uni/infra/funcprof/grp-switchOob" />
            <infraNodeBlk name="default" from_="101" to_="103" />
        </mgmtNodeGrp>
    </infraInfra>
</polUni>
```

Note You can configure the APIC server to use out-of-band management connectivity as the default connectivity mode.

```
POST https://apic-ip-address/api/node/mo/.xml
<polUni>
<fabricInst>
    <mgmtConnectivityPrefs interfacePref="ooband"/>
</fabricInst>
</polUni>
```

Configuring Static Out-of-Band Management Access Using the REST API

Before you begin

The APIC out-of-band management connection link must be 1 Gbps.

Step 1 Create an out-of-band contract.

Example:

```
<polUni>
    <fvTenant name="mgmt">
        <!-- Contract -->
        <vzOOBBrCP name="oob-default">
            <vzSubj name="oob-default">
```

```
                <vzRsSubjFiltAtt tnVzFilterName="default" />
            </vzSubj>
        </vzOOBBrCP>
    </fvTenant>
</polUni>
```

Step 2 Associate the out-of-band contract with an out-of-band EPG.

Example:

```
<polUni>
    <fvTenant name="mgmt">
        <mgmtMgmtP name="default">
            <mgmtOoB name="default">
                <mgmtRsOoBProv tnVzOOBBrCPName="oob-default" />
            </mgmtOoB>
        </mgmtMgmtP>
    </fvTenant>
</polUni>
```

Step 3 Associate the out-of-band contract with an external management EPG.

Example:

```
<polUni>
    <fvTenant name="mgmt">
        <mgmtExtMgmtEntity name="default">
            <mgmtInstP name="oob-mgmt-ext">
                <mgmtRsOoBCons tnVzOOBBrCPName="oob-default" />
                <!-- SUBNET from where switches are managed -->
                <mgmtSubnet ip="<mgmt_subnet_ip_address>" />
            </mgmtInstP>
        </mgmtExtMgmtEntity>
    </fvTenant>
</polUni>
```

Step 4 Create static out-of-band management IP addresses and assign them to node IDs.

CHECK IP Addresses

Example:

```
<polUni>
  <fvTenant name="mgmt">
    <mgmtMgmtP name="default">
      <mgmtOoB name="default">
        <mgmtRsOoBStNode tDn="topology/pod-1/node-101"
                         addr="<ip_address_1>"
                         gw="<gw_address>"/>
        <mgmtRsOoBStNode tDn="topology/pod-1/node-102"
                         addr="<ip_address_2>"
                         gw="<gw_address>"/>
        <mgmtRsOoBStNode tDn="topology/pod-1/node-103"
                         addr="<ip_address_3>"
                         gw="<gw_address>"/>
      </mgmtOoB>
    </mgmtMgmtP>
  </fvTenant>
</polUni>
```

Managing Configuration Files

Overview

This topic provides information on:

- How to use configuration Import and Export to recover configuration states to the last known good state using the Cisco APIC
- How to encrypt secure properties of Cisco APIC configuration files

You can do both scheduled and on-demand backups of user configuration. Recovering configuration states (also known as "roll-back") allows you to go back to a known state that was good before. The option for that is called an Atomic Replace. The configuration import policy (configImportP) supports atomic + replace (importMode=atomic, importType=replace). When set to these values, the imported configuration overwrites the existing configuration, and any existing configuration that is not present in the imported file is deleted. As long as you do periodic configuration backups and exports, or explicitly trigger export with a known good configuration, then you can later restore back to this configuration using the following procedures for the CLI, REST API, and GUI.

For more detailed conceptual information about recovering configuration states using the Cisco APIC, please refer to the *Cisco Application Centric Infrastructure Fundamentals Guide.*

The following section provides conceptual information about encrypting secure properties of configuration files:

Backing Up, Restoring, and Rolling Back Configuration Files Workflow

This section describes the workflow of the features for backing up, restoring, and rolling back configuration files. All of the features described in this document follow the same workflow pattern. Once the corresponding policy is configured, **admintSt** must be set to **triggered** in order to trigger the job.

Once triggered, an object of type **configJob** (representing that run) is created under a container object of type **configJobCont**. (The naming property value is set to the policy DN.) The container's **lastJobName** field can be used to determine the last job that was triggered for that policy.

Note Up to five **configJob** objects are kept under a single job container at a time, with each new job triggered. The oldest job is removed to ensure this.

The **configJob** object contains the following information:

- Execution time
- Name of the file being processed/generated
- Status, as follows:
  - Pending
  - Running

- Failed
- Fail-no-data
- Success
- Success-with-warnings

- Details string (failure messages and warnings)
- Progress percentage = 100 * lastStepIndex/totalStepCount
- Field lastStepDescr indicating what was being done last

About Configuration Export to Controllers

Configuration export extracts user-configurable managed object (MO) trees from all 32 shards in the cluster, writes them into separate files, then compresses them into a tar gzip file. The configuration export then uploads the tar gzip file to a preconfigured remote location (configured through **configRsRemotePath** pointing to a **fileRemotePath** object) or stores it as a **snapshot** on the controller(s).

Note See the Snapshots section for more details.

The **configExportP** policy is configured as follows:

- **name**—Policy name.
- **format**—Format in which the data is stored inside the exported archive (xml or json).
- **targetDn**—The domain name (DN) of the specific object you want to export. (Empty means everything.)
- **snapshot**—When true, the file is stored on the controller; no remote location configuration is needed.
- **includeSecureFields**—Set to true by default, this indicates whether the encrypted fields (passwords, etc.) should be included in the export archive.

Note The **configSnapshot** object is created holding the information about this snapshot. (See the Snapshots section.)

Scheduling Exports

An export policy can be linked with a scheduler, which triggers the export automatically based on a preconfigured schedule. This is done through the **configRsExportScheduler** relation from the policy to a **trigSchedP** object. (See the Sample Configuration section.)

Note A scheduler is optional. A policy can be triggered at any time by setting the adminSt to **triggered**.

About Configuration Import to Controller

Configuration import downloads, extracts, parses, analyzes, and applies the specified, previously exported archive one shard at a time in the following order: infra, fabric, tn-common, then everything else. The fileRemotePath configuration is performed the same way as for export (through configRsRemotePath). Importing snapshots is also supported.

The **configImportP** policy is configured as follows:

- **name**—Policy name
- **fileName**—Name of the archive file (not the path file) to be imported
- **importMode**
  - Best-effort mode: Each MO is applied individually, and errors only cause the invalid MOs to be skipped.

Note If the object is not present on the controller, none of the children of the object get configured. Best-effort mode attempts to configure the children of the object.

  - Atomic mode: configuration is applied by whole shards. A single error causes the whole shard to be rolled back to its original state.

- **importType**
  - Replace—Current system configuration is replaced with the contents or the archive being imported. (Only atomic mode is supported.)
  - Merge—Nothing is deleted, and archive content is applied on top the existing system configuration.
- **snapshot**—When true, the file is taken from the controller and no remote location configuration is needed.
- **failOnDecryptErrors**—(true by default) The file fails to import if the archive was encrypted with a different key than the one that is currently set up in the system.

Troubleshooting

The following scenarios may need troubleshooting:

- If the generated archive could not be downloaded from the remote location, refer to the Connectivity Issues section.
- If the import succeeded with warnings, check the details.
- If a file could not be parsed, refer to the following scenarios:
  - If the file is not a valid XML or JSON file, check whether the files from the exported archive were manually modified.
  - If an object property has an unknown property or property value, it may be because:
    - The property was removed or an unknown property value was manually entered.
    - The model type range was modified (non-backward compatible model change).

- The naming property list was modified.

- If an MO could not be configured, note the following:
  - Best-effort mode logs the error and skips the MO.
  - Atomic mode logs the error and skips the shard.

Configuration File Encryption

As of release 1.1(2), the secure properties of APIC configuration files can be encrypted by enabling AES-256 encryption. AES encryption is a global configuration option; all secure properties conform to the AES configuration setting. It is not possible to export a subset of the ACI fabric configuration such as a tenant configuration with AES encryption while not encrypting the remainder of the fabric configuration. See the *Cisco Application Centric Infrastructure Fundamentals*, "Secure Properties" chapter for the list of secure properties.

The APIC uses a 16 to 32 character passphrase to generate the AES-256 keys. The APIC GUI displays a hash of the AES passphrase. This hash can be used to see if the same passphrases was used on two ACI fabrics. This hash can be copied to a client computer where it can be compared to the passphrase hash of another ACI fabric to see if they were generated with the same passphrase. The hash cannot be used to reconstruct the original passphrase or the AES-256 keys.

Observe the following guidelines when working with encrypted configuration files:

- Backward compatibility is supported for importing old ACI configurations into ACI fabrics that use the AES encryption configuration option.

Note Reverse compatibility is not supported; configurations exported from ACI fabrics that have enabled AES encryption cannot be imported into older versions of the APIC software.

- Always enable AES encryption when performing fabric backup configuration exports. Doing so will assure that all the secure properties of the configuration will be successfully imported when restoring the fabric.

Note If a fabric backup configuration is exported without AES encryption enabled, none of the secure properties will be included in the export. Since such an unencrypted backup would not include any of the secure properties, it is possible that importing such a file to restore a system could result in the administrator along with all users of the fabric being locked out of the system.

- The AES passphrase that generates the encryption keys cannot be recovered or read by an ACI administrator or any other user. The AES passphrase is not stored. The APIC uses the AES passphrase to generate the AES keys, then discards the passphrase. The AES keys are not exported. The AES keys cannot be recovered since they are not exported and cannot be retrieved via the REST API.

- The same AES-256 passphrase always generates the same AES-256 keys. Configuration export files can be imported into other ACI fabrics that use the same AES passphrase.
- For troubleshooting purposes, export a configuration file that does not contain the encrypted data of the secure properties. Temporarily turning off encryption before performing the configuration export removes the values of all secure properties from the exported configuration. To import such a configuration file that has all secure properties removed, use the import merge mode; do not use the import replace mode. Using the import merge mode will preserve the existing secure properties in the ACI fabric.
- By default, the APIC rejects configuration imports of files that contain fields that cannot be decrypted. Use caution when turning off this setting. Performing a configuration import inappropriately when this default setting is turned off could result in all the passwords of the ACI fabric to be removed upon the import of a configuration file that does not match the AES encryption settings of the fabric.

Note Failure to observe this guideline could result in all users, including fabric administrations, being locked out of the system.

About the fileRemotePath Object

The fileRemotePath object holds the following remote location-path parameters:

- Hostname or IP
- Port
- Protocol: FTP, SCP, and others
- Remote directory (not file path)
- Username
- Password

Note The password must be resubmitted every time changes are made.

Sample Configuration

The following is a sample configuration:

Under **fabricInst** (uni/fabric), enter:

```
<fileRemotePath name="path-name" host="host name or ip" protocol="scp"
remotePath="path/to/some/folder" userName="user-name" userpasswd="password" />
```

Configuring a Remote Location Using the REST API

This procedure explains how to create a remote location using the REST API.

```
<fileRemotePath name="local" host="host or ip" protocol="ftp|scp|sftp" remotePath="path to
 folder" userName="uname" userPasswd="pwd" />
```

Configuring Configuration File Export to Controller Using the REST API

Before you begin

Create a remote path and scheduling policy.

Note When providing a remote location, if you set the snapshot to `True`, the backup ignores the remote path and stores the file on the controller.

SUMMARY STEPS

1. Create a configuration export policy by sending a POST request with XML such as the following example.

DETALED STEPS

Create a configuration export policy by sending a POST request with XML such as the following example.

Example:

```
<configExportP name="policy-name" format="xml" targetDn="/some/dn or empty which means everything"
snapshot="false" adminSt="triggered">
<configRsRemotePath tnFileRemotePathName="some remote path name" />
<configRsExportScheduler tnTrigSchedPName="some scheduler name" />
</configExportP>
```

Configuring a Configuration File Import Policy Using the REST API

SUMMARY STEPS

1. Configure a configuration file import policy, send a post with XML such as the following example:

DETALED STEPS

Configure a configuration file import policy, send a post with XML such as the following example:

Example:

```
<configImportP name="policy-name" fileName="someexportfile.tgz" importMode="atomic"
     importType="replace" snapshot="false" adminSt="triggered">
<configRsRemotePath tnFileRemotePathName="some remote path name" />
</configImportP>
```

Encrypting Configuration Files Using the REST API

SUMMARY STEPS

1. To encrypt a configuration file using the REST API, send a post with XML such as the following example:

DETAILED STEPS

To encrypt a configuration file using the REST API, send a post with XML such as the following example:

Example:

```
https://apic-ip-address/api/mo/uni/fabric.xml
<pkiExportEncryptionKey passphrase="abcdefghijklmnopqrstuvwxyz" strongEncryptionEnabled="true"/>
```

Snapshots and Rollbacks

Snapshots

Snapshots are configuration backup archives, stored (and replicated) in a controller managed folder. To create one, an export can be performed with the **snapshot** property set to true. In this case, no remote path configuration is needed. An object of **configSnapshot** type is created to expose the snapshot to the user.

You can create recurring snapshots, which are saved to **Admin** > **Import/Export** > **Export Policies** > **Configuration** > **defaultAuto**.

configSnapshot objects provide the following:

- file name
- file size
- creation date
- root DN indicating what the snapshot is of (fabric, infra, specific tenant, and so on)
- ability to remove a snapshot (by setting the retire field to true)

To import a snapshot, first create an import policy. Navigate to **Admin** > **Import/Export** and click **Import Policies**. Right click and choose **Create Configuration Import Policy** to set the import policy attributes.

About Rollbacks

The **configRollbackP** policy is used to undo the changes made between two snapshots. Managed Objects (MOs) are processed as follows:

- Deleted MOs are recreated.
- Created MOs are deleted.

- Modified MOs are reverted.

Note The rollback feature operates only on snapshots. Remote archives are not supported. If you want to use the data in a remote archive, use the snapshot manager to create a snapshot from from the data for the rollback. The policy does not require a remote path configuration.

Rollback Workflow

The policy snapshotOneDN and snapshotTwoDn fields must be set and the first snapshot (S1) must precede snapshot two (S2). Once triggered, snapshots are extracted and analyzed, and the difference between them is calculated and applied.

MOs are located that are:

- Present in S1 but not present in S2—These MOs are deleted and rollback re-creates them.
- Not present in S1 but not present in S2—These MOs are created after S1 and rollback deletes them if:
  - These MOs are not modified after S2 is taken.
  - None of the MO descendants are created or modified after S2 is taken.
- Present in both S1 and S2, but with different property values—These MO properties are reverted to S1, unless the property was modified to a different value after S2 is taken. In this case, it is left as is.

  The rollback feature also generates a diff file that contains the configuration generated as a result of these calculations. Applying this configuration is the last step of the rollback process. The content of this file can be retrieved through a special REST API called readiff: apichost/mqapi2/snapshots.readiff.xml?jobdn=SNAPSHOT_JOB_DN.

  Rollback (which is difficult to predict) also has a preview mode (set preview to true), which prevents rollback from making any actual changes. It calculates and generates the diff file, allowing you to preview what exactly is going to happen once the rollback is actually performed.

Diff Tool

Another special REST API is available, which provides diff functionality between two snapshots: apichost/mqapi2/snapshots.diff.xml?s1dn=SNAPSHOT_ONE_DN&s2dn=SNAPSHOT_TWO_DN.

Uploading and Downloading Snapshots Using the REST API

The **configSnapshotManagerP** policy allows you to create snapshots from remotely stored export archives. You can attach a remote path to the policy, provide the file name (same as with configImportP), set the mode to download, and trigger. The manager downloads the file, analyzes it to make sure that the archive is valid, stores it on the controller, and creates the corresponding configSnapshot object. The snapshot manager also allow you to upload a snapshot archive to a remote location. In this case, the mode must be set to upload.

Before you begin

Set up remotely stored archives.

SUMMARY STEPS

1. To download or upload a snapshot policy, send a POST request with XML such as the following:

DETAILED STEPS

To download or upload a snapshot policy, send a POST request with XML such as the following:

Example:

```
<configSnapshotManagerP name="policy-name" fileName="someexportfile.tgz"
    mode="upload|download" adminSt="triggered">
<configRsRemotePath tnFileRemotePathName="some remote path name" />
</configSnapshotManagerP>
```

Configuring and Executing a Configuration Rollback Using the REST API

Before you begin

Create a rollback policy and a snapshot.

SUMMARY STEPS

1. To configure and execute a rollback, send a POST request with XML such as the following:

DETAILED STEPS

To configure and execute a rollback, send a POST request with XML such as the following:

Example:

```
<configRollbackP name="policy-name" snapshotOneDn="dn/of/snapshot/one"
snapshotOneDn="dn/of/snapshot/two" preview="false" adminSt="triggered" />
```

Using Configuration Zones

Configuration Zones

Configuration zones divide the ACI fabric into different zones that can be updated with configuration changes at different times. This limits the risk of deploying a faulty fabric-wide configuration that might disrupt traffic or even bring the fabric down. An administrator can deploy a configuration to a non-critical zone, and then deploy it to critical zones when satisfied that it is suitable.

The following policies specify configuration zone actions:

- `infrazone:ZoneP` is automatically created upon system upgrade. It cannot be deleted or modified.

- `infrazone:Zone` contains one or more pod groups (`PodGrp`) or one or more node groups (`NodeGrp`).

Note You can only choose `PodGrp` or `NodeGrp`; both cannot be chosen.

A node can be part of only one zone (`infrazone:Zone`). `NodeGrp` has two properties: name, and deployment mode. The deployment mode property can be:

- `enabled` - Pending updates are sent immediately.
- `disabled` - New updates are postponed.

Note

- Do not upgrade, downgrade, commission, or decommission nodes in a disabled configuration zone.
- Do not do a clean reload or an uplink/downlink port conversion reload of nodes in a disabled configuration zone.

- `triggered` - pending updates are sent immediately, and the deployment mode is automatically reset to the value it had before the change to `triggered`.

When a policy on a given set of nodes is created, modified, or deleted, updates are sent to each node where the policy is deployed. Based on policy class and `infrazone` configuration the following happens:.

- For policies that do not follow `infrazone` configuration, the APIC sends updates immediately to all the fabric nodes.
- For policies that follow `infrazone` configuration, the update proceeds according to the `infrazone` configuration:
  - If a node is part of an `infrazone:Zone`, the update is sent immediately if the deployment mode of the zone is set to enabled; otherwise the update is postponed.
  - If a node is not part of an`infrazone:Zone`, the update is done immediately, which is the ACI fabric default behavior.

Configuration Zone Supported Policies

The following policies are supported for configuration zones:

```
analytics:CfgSrv
bgp:InstPol
callhome:Group
callhome:InvP
callhome:QueryGroup
cdp:IfPol
cdp:InstPol
comm:Pol
comp:DomP
coop:Pol
datetime:Pol
dbgexp:CoreP
dbgexp:TechSupP
```

```
dhcp:NodeGrp
dhcp:PodGrp
edr:ErrDisRecoverPol
ep:ControlP
ep:LoopProtectP
eqptdiagp:TsOdFabP
eqptdiagp:TsOdLeafP
fabric:AutoGEp
fabric:ExplicitGEp
fabric:FuncP
fabric:HIfPol
fabric:L1IfPol
fabric:L2IfPol
fabric:L2InstPol
fabric:L2PortSecurityPol
fabric:LeCardP
fabric:LeCardPGrp
fabric:LeCardS
fabric:LeNodePGrp
fabric:LePortP
fabric:LePortPGrp
fabric:LFPortS
fabric:NodeControl
fabric:OLeafS
fabric:OSpineS
fabric:PodPGrp
fabric:PortBlk
fabric:ProtGEp
fabric:ProtPol
fabric:SFPortS
fabric:SpCardP
fabric:SpCardPGrp
fabric:SpCardS
fabric:SpNodePGrp
fabric:SpPortP
fabric:SpPortPGrp
fc:DomP
fc:FabricPol
fc:IfPol
fc:InstPol
file:RemotePath
fvns:McastAddrInstP
fvns:VlanInstP
fvns:VsanInstP
fvns:VxlanInstP
infra:AccBaseGrp
infra:AccBndlGrp
infra:AccBndlPolGrp
infra:AccBndlSubgrp
infra:AccCardP
infra:AccCardPGrp
infra:AccNodePGrp
infra:AccPortGrp
infra:AccPortP
infra:AttEntityP
infra:CardS
infra:ConnFexBlk
infra:ConnFexS
infra:ConnNodeS
infra:DomP
infra:FexBlk
infra:FexBndlGrp
infra:FexGrp
infra:FexP
```

```
infra:FuncP
infra:HConnPortS
infra:HPathS
infra:HPortS
infra:LeafS
infra:NodeBlk
infra:NodeGrp
infra:NodeP
infra:OLeafS
infra:OSpineS
infra:PodBlk
infra:PodGrp
infra:PodP
infra:PodS
infra:PolGrp
infra:PortBlk
infra:PortP
infra:PortS
infra:PortTrackPol
infra:Profile
infra:SHPathS
infra:SHPortS
infra:SpAccGrp
infra:SpAccPortGrp
infra:SpAccPortP
infra:SpineP
infra:SpineS
isis:DomPol
l2ext:DomP
l2:IfPol
l2:InstPol
l2:PortSecurityPol
l3ext:DomP
lacp:IfPol
lacp:LagPol
lldp:IfPol
lldp:InstPol
mcp:IfPol
mcp:InstPol
mgmt:NodeGrp
mgmt:PodGrp
mon:FabricPol
mon:InfraPol
phys:DomP
psu:InstPol
qos:DppPol
snmp:Pol
span:Dest
span:DestGrp
span:SpanProv
span:SrcGrp
span:SrcTargetShadow
span:SrcTargetShadowBD
span:SrcTargetShadowCtx
span:TaskParam
span:VDest
span:VDestGrp
span:VSpanProv
span:VSrcGrp
stormctrl:IfPol
stp:IfPol
stp:InstPol
stp:MstDomPol
stp:MstRegionPol
```

```
trig:SchedP
vmm:DomP
vpc:InstPol
vpc:KAPol
```

Creating Configuration Zones Using the REST API

Before you begin

This procedure explains how to create a configuration zone using the REST API.

Create a configuration zone using the REST API leaf switch or pod examples below.

Example:

Creating a Config Zone with Leaf Switches

```
<infraInfra>
<infrazoneZoneP name="default">
<infrazoneZone name="Group1" deplMode="disabled">
<infrazoneNodeGrp name="nodeGroup">
<infraNodeBlk name="nodeblk1" from_=101 to_=101/>
<infraNodeBlk name="nodeblk2" from_=103 to_=103/>
</infrazoneNodeGrp>
</infrazoneZone>
<infrazoneZone name="Group2" deplMode="enabled">
<infrazoneNodeGrp name="nodeGroup2">
<infraNodeBlk name="nodeblk" from_=102 to_=102/>
</infrazoneNodeGrp>
</infrazoneZone>
</infrazoneZoneP>
</infraInfra>
```

Example:

Creating a Config Zone with Pods

```
<infraInfra>
    <infrazoneZoneP name="default">
        <infrazoneZone name="testZone" descr="testZone-Description" deplMode="enabled">
            <infrazonePodGrp name="podGroup1">
                <infraPodBlk name="group1" from_=101 to_=101/>
                <infraPodBlk name="group2" from_=103 to_=103/>
            </infrazonePodGrp>
            <infrazonePodGrp name="podGroup2">
                <infraPodBlk name="group" from_=102 to_=102/>
            </infrazonePodGrp>
        </infrazoneZone>
    </infrazoneZoneP>
</infraInfra>
```