Use Cases with Data Models

In this section, certain uses cases with data models are described.

Request for AAA Access Details

In this use case, you use a Calvados model to view AAA access details.

Note If any user on XR is deleted, the local database checks whether there is a first user on Calvados VM.

- If there is a first user, no syncing occurs.
- If there is no first user, then the first user on XR (based on the order of creation) is synced to Calvados VM.

Prerequisites

- Ensure that the user is added to the Calvados environment. This is because even if the user is added to the XR environment and has `root-lr` permissions, access to Calvados models is denied.
- Establish a NETCONF or gRPC connection between the router and the client application.

Note The gRPC YANG path or JSON data is based on YANG module name and not YANG namespace.

1. Using standard YANG tools, send a request to the router from the client using the NETCONF <get> operation.

```
[ Request ]
<get>
  <filter type="subtree">
    <aaa xmlns="http://tail-f.com/ns/aaa/1.1">
      <privileged-access xmlns="http://www.cisco.com/calvados/aaa_show"/>
```

```
      </aaa>
    </filter>
  </get>
```

2. Verify the response sent by the router to the client.

```
[ Response ]
<?xml version="1.0" encoding="UTF-8"?><data
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <aaa xmlns="http://tail-f.com/ns/aaa/1.1">
   <privileged-access xmlns="http://www.cisco.com/calvados/aaa_show">
    <shell-access>None</shell-access>
    <first-user>root</first-user>
    <first-user-change>No</first-user-change>
    <current-disaster-recovery-user>root</current-disaster-recovery-user>
   </privileged-access>
  </aaa>
 </data>
```

Note To accomplish this task using gRPC GetOper request:

```
{
    "tailf-aaa:aaa": {
        "aaa_show:privileged-access": [
            null
        ]
    }
}
```

gRPC GetOper response:

```
{
 "tailf-aaa:aaa": {
  "aaa_show:privileged-access": {
   "shell-access": "None",
   "first-user": "root",
   "first-user-change": "No",
   "current-disaster-recovery-user": "root"
  }
 }
}
```

Use OC Model to Configure Static Route

In this use case, you configure a static route with next hop using OC network instance local routing (openconfig-ni-local-routing) data model.

Note The OC interface maps all IP configurations for parent interface under a VLAN with index 0. This restricts configuring a sub interface with tag 0.

1. Create a static route with next hop interface.

```
<edit-config>
<target>
<candidate/>
</target>
<config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
<network-instances xmlns="http://openconfig.net/yang/network-instance">
       <network-instance>
       <name>default</name>
        <protocols>
         <protocol>
           <identifier
xmlns:idx="http://openconfig.net/yang/policy-types">idx:STATIC</identifier>
            <name>DEFAULT</name>
          <config>
           <identifier
xmlns:idx="http://openconfig.net/yang/policy-types">idx:STATIC</identifier>
            <name>DEFAULT</name>
          </config>
<static-routes>
<static>
<prefix>5.5.5.0/24</prefix>
<config>
<prefix>5.5.5.0/24</prefix>
</config>
<next-hops>
<next-hop>
<index>link-1</index>
<config>
<index>link-1</index>
<next-hop>6.6.6.1</next-hop>
</config>
</next-hop>
</next-hops>
</static>
</static-routes>
    </protocol>
        </protocols>
       </network-instance>
      </network-instances>
</config>
</edit-config>
```

2. Verify the static route configuration.

```
<get-config>
<source>
<candidate/>
</source>
<filter>
<network-instances xmlns="http://openconfig.net/yang/network-instance">
       <network-instance>
       <name>default</name>
        <protocols>
        <protocol>
          <identifier
xmlns:idx="http://openconfig.net/yang/policy-types">idx:STATIC</identifier>
           <name>DEFAULT</name>
<static-routes>
<static>
<prefix>5.5.5.0/24</prefix>
</static>
</static-routes>
        </protocol>
        </protocols>
       </network-instance>
```

```
      </network-instances>
</filter>
</get-config>
```

Using NETCONF with Flexible CLI Configuration Groups

If you want to use NETCONF protocol with flexible CLI conifguration groups, you need to use the inherited configuration. To transition to NETCONF and YANG based configuration from a CLI configuration which includes flexible CLI configuration groups, use the following steps. Using these steps, you can retrieve all the configuration on the device which can be used in further NETCONF operations.

1. Send a NETCONF **get-config** request with source as `<running-inheritance/>` .

   ```
   <rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
       <get-config>
           <source>
             <running-inheritance xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-group-cfg"/>

           </source>
        </get-config>
   </rpc>
   ##
   ```

   This operation returns all the configuration present on the device (inherited or expanded) in the following format:

   ```
   <rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
       <data>
         ….
       </data>
   </rpc-reply
   ```

2. To apply the configuration to another device, send a NETCONF **edit-config** request in the following format:

   ```
   <rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
     <edit-config>
       <target>
          <candidate/>
       </target>
       <config>
       </config>
     </edit-config>
   </rpc>
   ```