Using Data Models

Using data models involves three tasks:

Obtain Data Models

The data models are available in the mgbl pie software package. Installing a package on the router installs specific features that are part of that package. Cisco IOS XR software is divided into various software packages to select the features to run on the router. Each package contains components that perform a specific set of router functions, such as routing, security, and so on.

Pre-requisites:

Ensure that the mgbl pie software image is loaded in the router.

For installation instructions, see *Perform System Upgrade and Install Feature Packages* chapter in the System Setup and Software Installation Guide for Cisco NCS 5500 Series Routers.

1. Verify that the data models are available using `netconf-monitoring` request.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
 <get>
   <filter type="subtree">
     <netconf-state xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring">
     <schemas/>
     </netconf-state>
   </filter>
 </get>
</rpc>
```

   All IOS XR and System Admin YANG models are displayed.

The YANG models can be retrieved from the router without logging into the router using **get-schema** command:

Get Schema List (data will be used in step 2).

```
<get>
<filter type="subtree">
<netconf-state xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring">
```

```
<schemas/>
</netconf-state>
</filter>
</get>
</rpc>
```

All the models on the router are displayed.

```
TRACE: 2016/06/13 11:11:42 transport.go:104: Reading from connection
TRACE: 2016/06/13 11:11:42 gnc_main.go:587: Session established (Id: 1009461378)
TRACE: 2016/06/13 11:11:42 session.go:93: Request:
<rpc message-id="16a79f87-1d47-4f7a-a16a-9405e6d865b9"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"><get><filter type="subtree"><netconf-state

xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring"><schemas/></netconf-state></filter></get></rpc>
TRACE: 2016/06/13 11:11:42 transport.go:104: Reading from connection
TRACE: 2016/06/13 11:11:42 session.go:117:
Response:
#143589
<rpc-reply message-id="16a79f87-1d47-4f7a-a16a-9405e6d865b9"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<data>
<netconf-state xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring">
<schemas>
<schema>
<identifier>Cisco-IOS-XR-crypto-sam-oper</identifier>
<version>2015-01-07</version>
<format>yang</format>
<namespace>http://cisco.com/ns/yang/Cisco-IOS-XR-crypto-sam-oper</namespace>
<location>NETCONF</location>
</schema>
<schema>
<identifier>Cisco-IOS-XR-crypto-sam-oper-sub1</identifier>
<version>2015-01-07</version>
<format>yang</format>
<namespace>http://cisco.com/ns/yang/Cisco-IOS-XR-crypto-sam-oper</namespace>
<location>NETCONF</location>
</schema>
<schema>
<identifier>Cisco-IOS-XR-snmp-agent-oper</identifier>
<version>2015-10-08</version>
<format>yang</format>
<namespace>http://cisco.com/ns/yang/Cisco-IOS-XR-snmp-agent-oper</namespace>
<location>NETCONF</location>
</schema>
------------<truncated>--------------
```

For more information about structure of data models, see YANG Module.

What To Do Next:

Enable the protocol to establish connection between the router and the client application.

Enable Protocol

The router communicates with the client application using protocols. On the router and client application, enable a communication protocol based on the requirement:

- NETCONF
- gRPC

Note Only the first root-lr user created on XR is synchronized as the first root-system user on System Admin, while the consecutive users are not synchronized. The consecutive users created on XR do not exist in the System Admin. Hence any operations through NETCONF or gRPC that requires sysadmin access performed by the consecutive users fails. To overcome this limitation, create the user with the same name in System Admin and grant permission by assigning them to the appropriate group.

For more information about protocols, see Communication Protocols.

Enable NETCONF over SSH Protocol

NETCONF is an XML-based protocol used over Secure Shell (SSH) transport to configure a network. The client applications use this protocol to request information from the router, and make configuration changes to the router.

For more information about NETCONF, see NETCONF Protocol.

Pre-requisites:

- Software package k9sec pie is installed on the router.
- Software package mgbl pie is installed on the router.
- Crypto keys are generated.

To enable the NETCONF protocol, complete these steps:

1. Enable NETCONF protocol over an SSH connection.

```
ssh server v2
ssh server netconf
netconf agent tty
netconf-yang agent ssh
```

   The default port number of 830 is used. A different port within the range of 1 to 65535 can be specified if required.

2. Set the session parameters.

```
router (config)# netconf-yang agent session {  limit value |  absolute-timeout value |
 idle-timeout value }
```

   where:

   - **limit value:** sets the maximum count for concurrent netconf-yang sessions. The range is from 1 to 1024.
   - **absolute-timeout value:** sets the absolute session lifetime, in minutes. The range is from 1 to 1440.
   - **idle-timeout value:** sets the idle session lifetime, in minutes. The range is from 1 to 1440.

3. Verify configuration settings for statistics and clients.

```
router (config)# do show netconf-yang statistics
router (config)# do show netconf-yang clients
```

Enable NETCONF

```
config
netconf-yang agent ssh
ssh server netconf port 830
!
```

Verify Configuration Using Statistics

After the NETCONF request is sent, use **do show netconf-yang statistics** command to verify the configuration.

```
show netconf-yang statistics
Summary statistics        requests|             total time|   min time per request|   max
time per request|   avg time per request|
other                              0|       0h  0m  0s   0ms|       0h  0m  0s   0ms|
 0h  0m  0s   0ms|       0h  0m  0s   0ms|
close-session                      4|       0h  0m  0s   3ms|       0h  0m  0s   0ms|
 0h  0m  0s   1ms|       0h  0m  0s   0ms|
kill-session                       0|       0h  0m  0s   0ms|       0h  0m  0s   0ms|
 0h  0m  0s   0ms|       0h  0m  0s   0ms|
get-schema                         0|       0h  0m  0s   0ms|       0h  0m  0s   0ms|
 0h  0m  0s   0ms|       0h  0m  0s   0ms|
get                                0|       0h  0m  0s   0ms|       0h  0m  0s   0ms|
 0h  0m  0s   0ms|       0h  0m  0s
get-config                         1|       0h  0m  0s   1ms|       0h  0m  0s   1ms|
 0h  0m  0s   1ms|       0h  0m  0s   1ms|
edit-config                        3|       0h  0m  0s   2ms|       0h  0m  0s   0ms|
 0h  0m  0s   1ms|       0h  0m  0s   0ms|
commit                             0|       0h  0m  0s   0ms|       0h  0m  0s   0ms|
 0h  0m  0s   0ms|       0h  0m  0s   0ms|
cancel-commit                      0|       0h  0m  0s   0ms|       0h  0m  0s   0ms|
 0h  0m  0s   0ms|       0h  0m  0s   0ms|
lock                               0|       0h  0m  0s   0ms|       0h  0m  0s   0ms|
 0h  0m  0s   0ms|       0h  0m  0s   0ms|
unlock                             0|       0h  0m  0s   0ms|       0h  0m  0s   0ms|
 0h  0m  0s   0ms|       0h  0m  0s   0ms|
discard-changes                    0|       0h  0m  0s   0ms|       0h  0m  0s   0ms|
 0h  0m  0s   0ms|       0h  0m  0s   0ms|
validate                           0|       0h  0m  0s   0ms|       0h  0m  0s   0ms|
 0h  0m  0s   0ms|       0h  0m  0s   0ms|
```

Verify Configuration Using Clients

```
show netconf-yang clients
client session ID|   NC version|    client connect time|        last OP time|        last
OP type|    <lock>|
22969|                  1.1|         0d  0h  0m  2s|             11:11:24|
close-session|     No|
```

What To Do Next:

After NETCONF is enabled, use the YANG data models to manage the relevant configurations.

Enable gRPC over HTTP/2 Protocol

Google-defined remote procedure call (gRPC) is an open-source RPC framework. gRPC supports IPv4 and v6 address families.

For more information about gRPC, see gRPC Protocol.

Pre-requisite:

- Configure TLS.

Note It is recommended to configure TLS. Enabling gRPC protocol uses the default HTTP/2 transport with no TLS enabled on TCP. gRPC mandates AAA authentication and authorization for all gRPC requests. If TLS is not configured, the authentication credentials are transferred over the network unencrypted. Enabling TLS ensures that the credentials are secure and encrypted. Non-TLS mode can only be used in secure internal network.

- Software package mgbl pie is installed on the router.

To enable the gRPC protocol, complete these steps:

1. Enable gRPC over an HTTP/2 connection.

```
Router# configure
Router (config)# grpc
```

2. Enable access to a specified port number.

```
Router (config-grpc)# port <port-number>
```

The <port-number> range is from 57344 to 57999. If a port number is unavailable, an error is displayed.

3. In the configuration mode, set the session parameters.

```
Router (config)# grpc{ address-family | dscp | max-request-per-user | max-request-total
 | max-streams | max-streams-per-user | no-tls | service-layer | tls-cipher | tls-mutual
 | tls-trustpoint | vrf }
```

where:

- **address-family:** set the address family identifier type
- **dscp:** set QoS marking DSCP on transmitted gRPC
- **max-request-per-user:** set the maximum concurrent requests per user
- **max-request-total:** set the maximum concurrent requests in total
- **max-streams:** set the maximum number of concurrent gRPC requests. The maximum subscription limit is 128 requests. The default is 32 requests
- **max-streams-per-user:** set the maximum concurrent gRPC requests for each user. The maximum subscription limit is 128 requests. The default is 32 requests
- **no-tls:** disable transport layer security (TLS). The TLS is enabled by default.

- **service-layer:** enable the grpc service layer configuration
- **tls-cipher:** enable the gRPC TLS cipher suites
- **tls-mutual:** set the mutual authentication
- **tls-trustpoint:** configure trustpoint
- **server-vrf:** enable server vrf

What To Do Next:

After gRPC is enabled, use the YANG data models to manage the relevant configurations.

Manage Configurations using Data Model

From the client application, use data models to manage the configurations of the router.

Prerequisites

- Software packages k9sec pie and mgbl are installed on the router.
- NETCONF or gRPC protocols enabled on the client and the router.

To manage configurations using data models, complete these steps:

1. Use a YANG tool to import the data model on the client application.
2. Configure the router by modifying the values of the data model using the YANG tool.

For more information on the values of the data models that can be configured, see Structure of YANG Models.

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.

Example: Configure CDP

In this example, you use the data model for CDP and configure CDP with the values as shown in the table:

CDP parameter	Description	Desired value for parameter
CDP Version	Specifies the version used to communicate with the neighboring devices	v1
Hold time	Specifies the duration for which the receiving device to hold the CDP packet	200 ms
Timer	Specifies how often the software sends CDP updates	80 ms

CDP parameter	Description	Desired value for parameter
Log Adjacency Table	Logs changes in the adjacency table. When CDP adjacency table logging is enabled, a syslog is generated each time a CDP neighbor is added or removed	enable

1. Download the configuration YANG data model for CDP `Cisco-IOS-XR-cdp-cfg.yang` from the router. To download the data model, see Obtain Data Models, on page 1.
2. Import the data model to the client application using any YANG tool.
3. Modify the leaf nodes of the data model:
   - enable (to enable cdp)
   - holdtime
   - timer
   - advertise v1 only
   - log adjacency

Configure CDP Using NETCONF

In this example, you use the data model for CDP and configure CDP using NETCONF RPC request:

```
<edit-config>
  <target>
   <candidate/>
  </target>
  <config xmlns:xc="urn:ietf:params:xml:n:netconf:base:1.0">
   <cdp xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-cdp-cfg">
   <timer>80</timer>
   <enable>true</enable>
   <log-adjacency></log-adjacency>
   <hold-time>200</holdtime>
   <advertise-v1-only></advertise-v1-only>
  </cdp>
 </config>
</edit-config>
```

Note CDP can also be configured under the interface configuration by augmenting the interface manager. Use the `Cisco-IOS-XR-ifmgr-cfg` YANG model to configure CDP under the interface configuration.

Configure CDP Using gRPC

In this example, you use the data model for CDP and configure CDP using gRPC MergeConfig RPC request:

```
{
 "Cisco-IOS-XR-cdp-cfg:cdp": {
  "timer": 50,
  "enable": true,
  "log-adjacency": [
   null
  ],
  "hold-time": 180,
  "advertise-v1-only": [
   null
  ]
 }
}
```

Note CDP can also be configured under the interface configuration by augmenting the interface manager. Use the `Cisco-IOS-XR-ifmgr-cfg` YANG model to configure CDP under the interface configuration.

Commit Configuration

Commit the configuration to set the new values in the current running configuration.

The configuration can also be committed through a `confimed-commit` operation. NETCONF and gRPC supports confirmed-commit RPC. This RPC requires an explicit confirmation from the user before the configuration takes effect on the router. This feature helps in verifying that the change in the configuration works correctly and does not cause fluctuation in the management connectivity. If the configuration change causes loss of management connectivity, the configuration is automatically rolled back to the previous committed configuration after the default `confirm-timeout` period of 600 seconds.

To commit a configuration, use </commit> RPC:

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

To confirm-commit a configuration:

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <commit>
        <confirmed/>
    </commit>
</rpc>
```

The confirmed-commit capability supports the <cancel-commit> operation and the <confirmed>, <confirm-timeout>, <persist>, and <persist-id> parameters for the <commit> operation.

```
<rpc message-id="101"
          xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
       <commit>
         <confirmed/>
         <persist>IQ,d4668</persist>
         <confirm-timeout>120</confirm-timeout>
       </commit>
</rpc>
```

A confirmed-commit request will fail with `Datastore Locked` error if:

- another operation is performed between a confirmed-commit and a confirming-commit operation
- another session has an active confirmed-commit request and a persist ID was not provided
- a persist ID was provided but did not match the persist ID of the active confirmed-commit session

gRPC Confirmed-Commit for Merging Configuration

gRPC confirmed-commit request can be issued for exisiting merge-config and cli-config operations. In this example, the request is made for merge-cli operation.

```
manageability/ems/client/client -oper merge-config -server_addr="<address>" -json_in_file
<directory-path>/<file>.json
-confirmed=yes -confirm_timeout=400
enter PID:14917:main.main
emsMergeConfig: Received ReqId 14917, Response '
----------------- gRPC Summary ----------------------
Operation: merge-config
Number of iterations: 1
Total bytes transferred: 126
Number of bytes per second: 374
Round trip throughputs Mbps: 0.002999
Ave elapsed time in seconds: 0.336079
Min elapsed time in seconds: 0.336079
Max elapsed time in seconds: 0.336079
--------------- End gRPC Summary ----------------------
The confirmed commit request should be followed by a confirming commit to make the
configuration permanent:
manageability/ems/client/client -oper commit -server_addr="<address>"
enter PID:14917:main.main
emsCommitConfig: Received ReqId 14917, Response '
----------------- gRPC Summary ----------------------
Operation: commit
Number of iterations: 1
Total bytes transferred: 126
Number of bytes per second: 374
Round trip throughputs Mbps: 0.002999
Ave elapsed time in seconds: 0.336079
Min elapsed time in seconds: 0.336079
Max elapsed time in seconds: 0.336079
--------------- End gRPC Summary ----------------------
```