Configureer NGMI en implementeer YANG in IOS XR

Inhoud

Inleiding

Dit document beschrijft een korte beschrijving van de GNMI in Cisco IOS® XR en hoe u PYANG kunt gebruiken en modelbomen kunt controleren.

Voorwaarden

Vereisten

Cisco raadt kennis van de volgende onderwerpen aan:

- Cisco IOS XR-platform.
- python.
- Netwerkbeheerprotocollen.

Gebruikte componenten

Dit document is niet beperkt tot specifieke hardwareversies en is van toepassing op 64-bits versie (eXR).

De informatie in dit document is gebaseerd op de apparaten in een specifieke laboratoriumomgeving. Alle apparaten die in dit document worden beschreven, hadden een opgeschoonde (standaard)configuratie. Als uw netwerk live is, moet u zorgen dat u de potentiële impact van elke opdracht begrijpt.

Achtergrondinformatie

gNMI-definitie

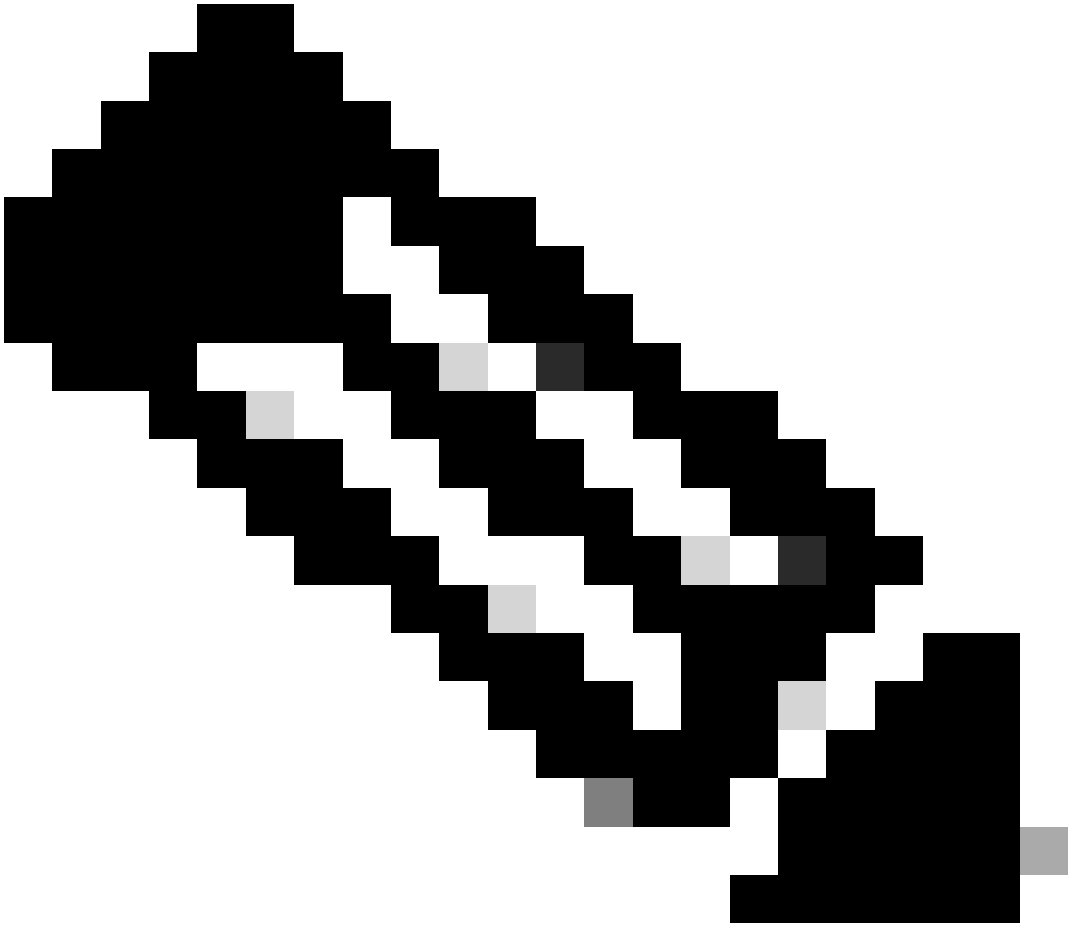
In het algemeen zijn er verschillende netwerkconfiguratieprotocollen, zoals NETCONF, RESTCONF, gNMI (Google Remote Procedure Calls (gRPC), gRPC Network Management Interface). Deze modellen worden gebruikt om te configureren om de netwerkapparaten te beheren en altijd te streven naar automatisering van processen die mechanisch kunnen zijn.

Deze protocollen maken gebruik van verschillende gegevensmodellen om de gebruikers te laten begrijpen wat het netwerkapparaat verwerkt, met andere woorden, het is een gestructureerde informatie, een schema, dat informatie normaliseert en hoe het wordt verbruikt door het apparaat, in dit geval de router.

GNMI houdt toezicht op de gegevensverwerking en levert RPC (Remote Procedure Calls) om de verschillende apparaten in het netwerk te bedienen.

gNMI heeft vier functies:

- Mogelijkheden: gNMI vraag de router de modellen die in de router worden geïnstalleerd, wordt dit verder verklaard in dit document.
- Ontvang: Elke bladcomponent in de gegevensboom kan aan de router worden gevraagd, deze verrichting verzoekt de gevraagde informatie.
- Set: De bladeren worden beschouwd als variabelen, wat hen van de veranderingsmogelijkheden voorziet, de vastgestelde verrichting helpt op dit toestaan dat de gebruiker een waarde in het gegevensmodel bijwerkt.
- Abonneren: Gebruikt in Telemetrie, deze functie helpt bij het trekken van gegevens van een bepaalde module in het model.



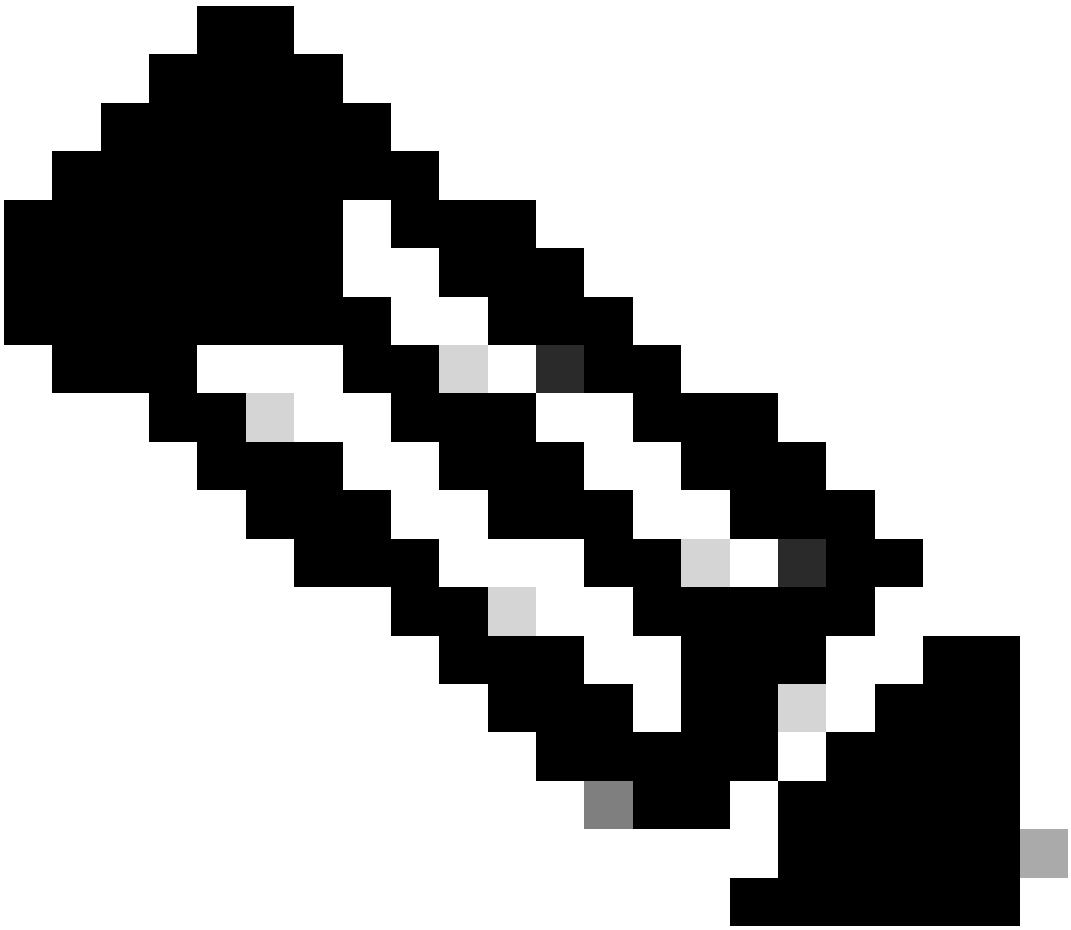
Opmerking: Cisco heeft veel informatie over dit onderwerp gedeeld. Klik op de volgende link: xrdocs blog - OpenConfig gNMI

GNMI-functies

Netwerkbeheerprotocol	GNMI
Gebruikt transport	HTTP 2/3
Ondersteund door	Neutraal voor leveranciers
Codering	Proto Buff

Proto Buff is de taal-neutrale, platform-neutrale methode van de-serialiseren en serialiseren van

gegevens tussen twee apparaten, waarin elk verzoek een antwoord heeft.



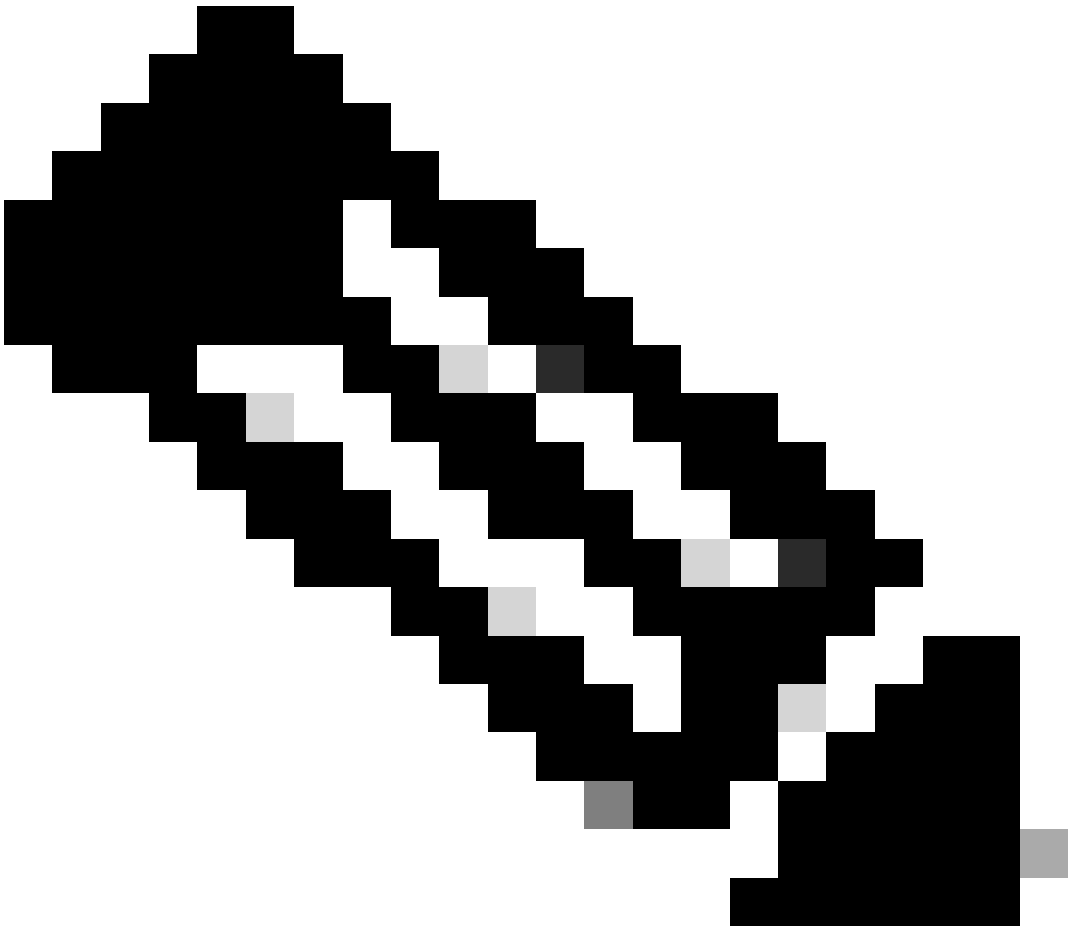
> Opmerking: Voor meer informatie in gRCP en Proto Buff klik op de volgende link: grpc Guide.

NMI-basisconfiguratie in Cisco IOS XR

Het volgende is de basisconfiguratie voor de router:

```
RP/0/RSP0/CPU0:XR(config)#grpc
RP/0/RSP0/CPU0:XR(config-grpc)#address-family ipv4
RP/0/RSP0/CPU0:XR(config-grpc)#max-request-total 256
RP/0/RSP0/CPU0:XR(config-grpc)#max-request-per-user 32
```

```
grpc
 address-family ipv4
 max-request-total 256
 max-request-per-user 32
```



Opmerking: een poort kan worden geconfigureerd op basis van de installatie, de standaard, zonder TLS te gebruiken is 57400, voor meer informatie klik: github - grpc

pYANG als validator

pYANG is een YANG validator geschreven in python. Deze bibliotheek voor python assisteert bij het controleren van de YANG modellen en ook, het kennen van hen.

Om deze functie uit te voeren zoals in de documentatie (pYANG-documentatie) wordt voorgesteld om een virtuele omgeving in de computer te creëren.

Documentatie voor virtuele omgeving venv uitvoeren

De applicatie moet werken:

```
python -m venv <name of the directory>
```

Bijvoorbeeld (in MacOS terminal):

```
% mkdir test
% cd test
% python3 -m venv virtual_env
% ls
virtual_env
```

Zo installeert u pYANG in deze virtuele omgeving cd in de directory en plakt u het volgende:

```
% cd virual_env
% git clone https://github.com/mbj4668/pyang.git
% cd pyang
% pip install -e .
```

Voor deze demonstratie werd python3 pip gebruikt, zodra de pip installeert -e wordt uitgegeven, activeert de venv: source <virtual environment directory>/bin/activation (voor MacOS).

```
% source virtual_env/bin/activate
```

```
% python3 -m pip install pyang
Collecting pyang
  Downloading pyang-2.6.0-py2.py3-none-any.whl (594 kB)
     |████████████████████████████████| 594 kB 819 kB/s
Collecting lxml
  Downloading lxml-5.1.0-cp39-cp39-macosx_11_0_arm64.whl (4.5 MB)
     |████████████████████████████████| 4.5 MB 14.2 MB/s
Installing collected packages: lxml, pyang
Successfully installed lxml-5.1.0 pyang-2.6.0
```

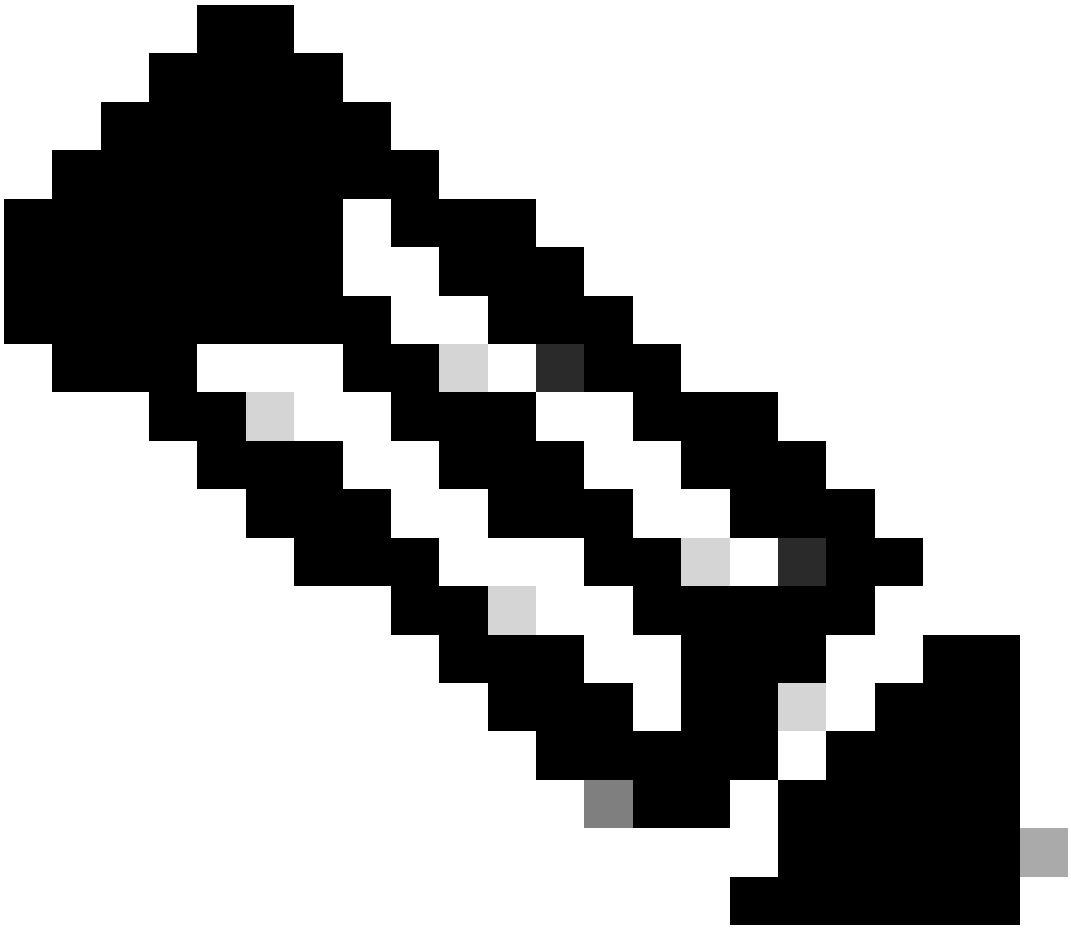
```
% pyang -h
Usage: pyang [options] [<filename>...]

Validates the YANG module in <filename> (or stdin), and all its dependencies.

Options:
  -h, --help            Show this help message and exit
  -v, --version         Show version number and exit
<snip>
```

Met geïnstalleerd en werkend PYANG, ga met modellen verder downloaden.

In de volgende link zijn er alle modellen die Cisco IOS XR gebruikt: Cisco IOS XR-modellen.

Aanbevolen wordt om deze modellen in de venv-directory te klonen met de volgende codelink: https://github.com/YangModels/yang.git

---



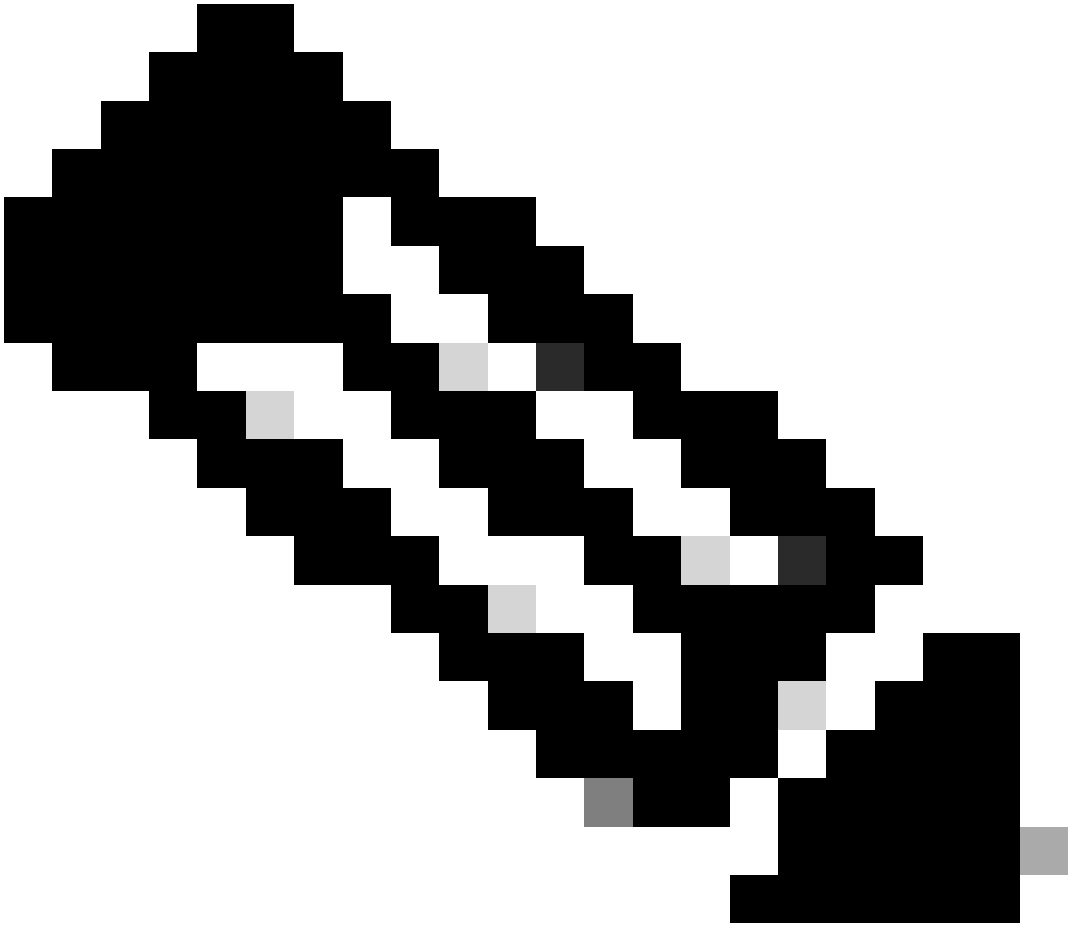
Opmerking: dit gebeurt niet wanneer de virtuele omgeving is geactiveerd.

```
% git clone https://github.com/YangModels/yang.git
Cloning into 'yang'...
remote: Enumerating objects: 54289, done.
remote: Counting objects: 100% (1910/1910), done.
remote: Compressing objects: 100% (323/323), done.
remote: Total 54289 (delta 1643), reused 1684 (delta 1586), pack-reused 52379
Receiving objects: 100% (54289/54289), 116.64 MiB | 8.98 MiB/s, done.
Resolving deltas: 100% (42908/42908), done.
Updating files: 100% (112197/112197), done.
```

Activeer de virtuele omgeving opnieuw en test de volgende query: pyang -f tree

yang/vendor/cisco/xr/711/Cisco-IOS-XR-ifmgr-cfg.yang.

```
(virtual_env) % pyang -f tree yang/vendor/cisco/xr/711/Cisco-IOS-XR-ifmgr-cfg.yang
yang/vendor/cisco/xr/711/Cisco-IOS-XR-ifmgr-cfg.yang:5: error: module "Cisco-IOS-XR-types" not found in
yang/vendor/cisco/xr/711/Cisco-IOS-XR-ifmgr-cfg.yang:8: error: module "cisco-semver" not found in searc
module: Cisco-IOS-XR-ifmgr-cfg
  +--rw global-interface-configuration
  |  +--rw link-status?   Link-status-enum
  +--rw interface-configurations
     +--rw interface-configuration* [active interface-name]
        +--rw dampening
        |  +--rw args?                 enumeration
        |  +--rw half-life?            uint32
        |  +--rw reuse-threshold?      uint32
        |  +--rw suppress-threshold?   uint32
        |  +--rw suppress-time?        uint32
        |  +--rw restart-penalty?      uint32
        +--rw mtus
        |  +--rw mtu* [owner]
        |     +--rw owner    xr:Cisco-ios-xr-string
        |     +--rw mtu      uint32
        +--rw encapsulation
        |  +--rw encapsulation?         string
        |  +--rw capsulation-options?   uint32
        +--rw shutdown?                      empty
        +--rw interface-virtual?             empty
        +--rw secondary-admin-state?         Secondary-admin-state-enum
        +--rw interface-mode-non-physical?   Interface-mode-enum
        +--rw bandwidth?                     uint32
        +--rw link-status?                   empty
        +--rw description?                   string
        +--rw active                         Interface-active
        +--rw interface-name                 xr:Interface-name
```



Opmerking: Merk op dat bladeren een gegevensindeling hebben zoals String, uint32, enzovoort; terwijl roots deze informatie niet weergeven. Operaties zoals GET en SET zijn gewijd aan pull/update deze waarden.

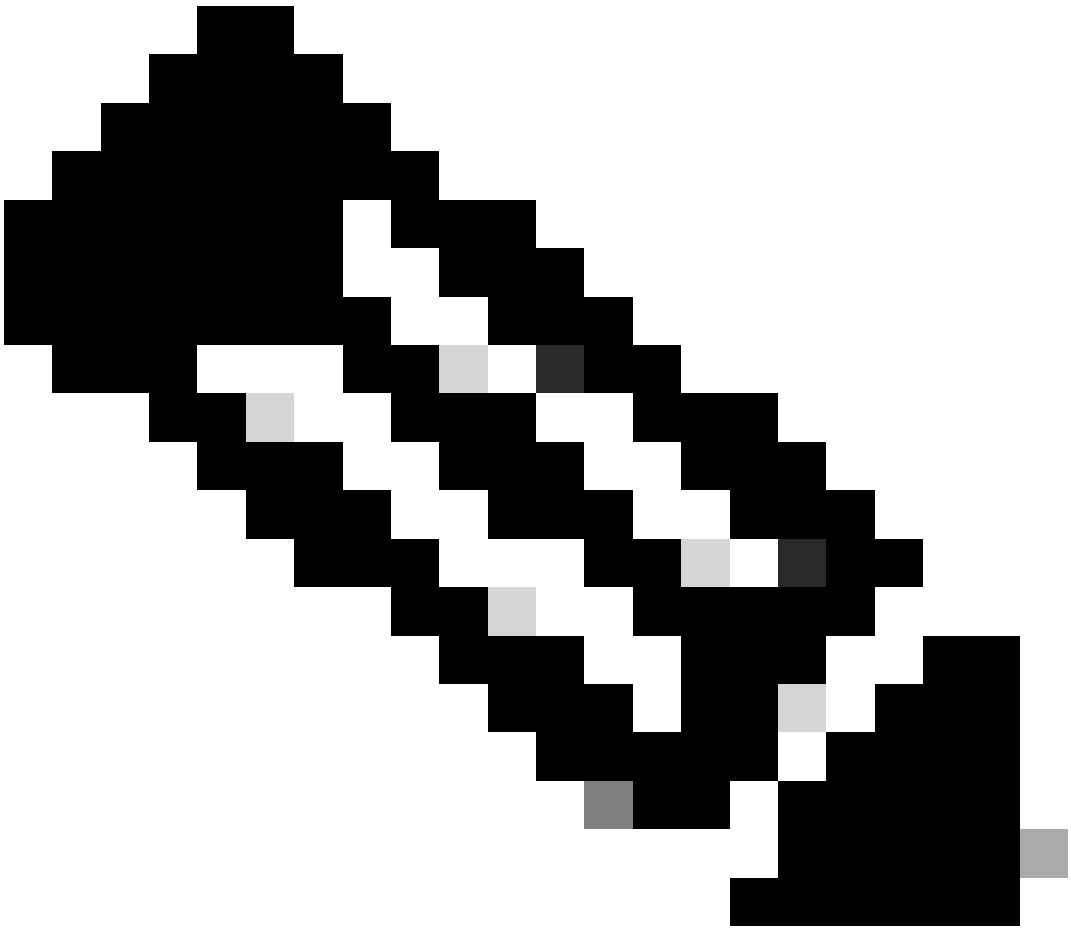
Een andere opmerking is dat de meeste modellen moeten worden uitgebreid om de volledige configuratie te hebben, in de CLI-output is er de basisconfiguratie voor interfacebeheer, voor het geval dat IPv4 moet worden weergegeven, gebruik de volgende:

```
% pyang -f tree yan2/vendor/cisco/xr/711/Cisco-IOS-XR-ifmgr-cfg.yang  yan2/vendor/cisco/xr/711/Cisco-IO
module: Cisco-IOS-XR-ifmgr-cfg
  +--rw global-interface-configuration
  |  +--rw link-status?   Link-status-enum
  +--rw interface-configurations
     +--rw interface-configuration* [active interface-name]
        +--rw dampening
        |  +--rw args?                enumeration
        |  +--rw half-life?           uint32
        |  +--rw reuse-threshold?     uint32
```

```
|  +--rw suppress-threshold?   uint32
|  +--rw suppress-time?        uint32
|  +--rw restart-penalty?      uint32
+--rw mtus
|  +--rw mtu* [owner]
|     +--rw owner    xr:Cisco-ios-xr-string
|     +--rw mtu      uint32
+--rw encapsulation
|  +--rw encapsulation?         string
|  +--rw capsulation-options?   uint32
+--rw shutdown?                                 empty
+--rw interface-virtual?                        empty
+--rw secondary-admin-state?                    Secondary-admin-state-enum
+--rw interface-mode-non-physical?              Interface-mode-enum
+--rw bandwidth?                                uint32
+--rw link-status?                              empty
+--rw description?                              string
+--rw active                                    Interface-active
+--rw interface-name                            xr:Interface-name
+--rw ipv4-io-cfg:ipv4-network
|  +--rw ipv4-io-cfg:bgp-pa
|  |  +--rw ipv4-io-cfg:input
|  |  |  +--rw ipv4-io-cfg:source-accounting?        boolean
|  |  |  +--rw ipv4-io-cfg:destination-accounting?   boolean
|  |  +--rw ipv4-io-cfg:output
|  |     +--rw ipv4-io-cfg:source-accounting?        boolean
|  |     +--rw ipv4-io-cfg:destination-accounting?   boolean
|  +--rw ipv4-io-cfg:verify
|  |  +--rw ipv4-io-cfg:reachable?      Ipv4-reachable
|  |  +--rw ipv4-io-cfg:self-ping?      Ipv4-self-ping
|  |  +--rw ipv4-io-cfg:default-ping?   Ipv4-default-ping
|  +--rw ipv4-io-cfg:bgp
|  |  +--rw ipv4-io-cfg:qppb
|  |  |  +--rw ipv4-io-cfg:input
|  |  |     +--rw ipv4-io-cfg:source?        Ipv4-interface-qppb
|  |  |     +--rw ipv4-io-cfg:destination?   Ipv4-interface-qppb
|  |  +--rw ipv4-io-cfg:flow-tag
|  |     +--rw ipv4-io-cfg:flow-tag-input
|  |        +--rw ipv4-io-cfg:source?        boolean
|  |        +--rw ipv4-io-cfg:destination?   boolean
|  +--rw ipv4-io-cfg:addresses
|  |  +--rw ipv4-io-cfg:secondaries
|  |  |  +--rw ipv4-io-cfg:secondary* [address]
|  |  |     +--rw ipv4-io-cfg:address      inet:ipv4-address-no-zone
|  |  |     +--rw ipv4-io-cfg:netmask      inet:ipv4-address-no-zone
|  |  |     +--rw ipv4-io-cfg:route-tag?   uint32
|  |  +--rw ipv4-io-cfg:primary!
|  |  |  +--rw ipv4-io-cfg:address      inet:ipv4-address-no-zone
|  |  |  +--rw ipv4-io-cfg:netmask      inet:ipv4-address-no-zone
|  |  |  +--rw ipv4-io-cfg:route-tag?   uint32
|  |  +--rw ipv4-io-cfg:unnumbered?    xr:Interface-name
|  |  +--rw ipv4-io-cfg:dhcp?          empty
|  +--rw ipv4-io-cfg:helper-addresses
|  |  +--rw ipv4-io-cfg:helper-address* [address vrf-name]
|  |     +--rw ipv4-io-cfg:address     inet:ipv4-address-no-zone
|  |     +--rw ipv4-io-cfg:vrf-name    xr:Cisco-ios-xr-string
|  +--rw ipv4-io-cfg:forwarding-enable?       empty
|  +--rw ipv4-io-cfg:icmp-mask-reply?         empty
|  +--rw ipv4-io-cfg:tcp-mss-adjust-enable?   empty
|  +--rw ipv4-io-cfg:ttl-propagate-disable?   empty
|  +--rw ipv4-io-cfg:point-to-point?          empty
|  +--rw ipv4-io-cfg:mtu?                     uint32
```

```
+--rw ipv4-io-cfg:ipv4-network-forwarding
   +--rw ipv4-io-cfg:directed-broadcast?   empty
   +--rw ipv4-io-cfg:unreachables?         empty
   +--rw ipv4-io-cfg:redirects?            empty
```

In deze query worden twee modellen gebruikt: Cisco-IOS-XR-ifmgr-cfg.yang en Cisco-IOS-XR-ipv4-io-cfg.yang, en nu wordt het IPv4-adres weergegeven als een blad.



> Opmerking: voor het geval u een fout ziet zoals: "yang/vendor/cisco/xr/711/Cisco-IOS-XR-ifmgr-cfg.yang:5: fout: module "Cisco-IOS-XR-types" niet gevonden in zoekpad" voeg de —path= in de opdracht toe.

Met dit gedaan en gecontroleerd, kan elke gebruiker informatie aanvragen met de gNMI-bewerkingen en de wijzigingsdatum, voor meer voorbeelden klik op de volgende link: [Programmability Configuration Guide]

In het geval dat de gebruiker een eenvoudige API wil uitvoeren, zijn er tools zoals: [grpcc.]

Deze API is geïnstalleerd via NPM, dit is de tool gebruikt in de koppeling Programmability Configuration Guide, genoemde link deelt meer voorbeelden voor gebruikers om vragen en antwoorden te testen.

Probleemoplossing:

Voor gNMI is het vereist om de query te controleren voordat het verzamelen van een ingang, de meeste van de API's zoals:

- muggezifterig
- grpcc
- gRPC

Al, toon de fout die de router produceerde.

Voorbeeld:

```
"cisco-grpc:errors": {
"error": [
{
"error-type": "application",
"error-tag": "operation-failed",
"error-severity": "error",
"error-message": "'YANG framework' detected the 'fatal' condition 'Operation failed'"
}
]
}
}
```

OF

```
"error": [
   {
    "error-type": "application",
    "error-tag": "operation-failed",
    "error-severity": "error",
    "error-path": <path>,
"error-message": "'sysdb' detected the 'warning' condition 'A verifier or EDM callback function returne
   }
  ]
```

Dit zijn Platform-afhankelijke fouten die langs de router moeten worden gecontroleerd. Aanbevolen wordt om te controleren dat de opdrachten in de query ook in de router via de CLI kunnen worden uitgegeven.

Voor dit type fouten of voor andere fouten die betrekking hebben op het Cisco IOS XR-platform, wordt de volgende informatie over TAC gedeeld:

- Query die wordt gebruikt en bewerking:

```
{
 "Cisco-IOS-XR-ifmgr-cfg:interface-configurations":
  { "interface-configuration": [
    {
      "active": "act",
      "interface-name": "Loopback0",
      "description": "LOCAL TERMINATION ADDRESS",
      "interface-virtual": [
        null
       ],
       "Cisco-IOS-XR-ipv4-io-cfg:ipv4-network": {
        "addresses": {
            "primary": {
              "address": "172.16.255.1",
              "netmask": "255.255.255.255"
       }
      }
     }
    }
   ]
 }
}
```

- Fout die wordt weergegeven (een van de bovenstaande beelden).
- Geef het volgende bevel uit:

```
show grpc trace all
```

Test de query een paar keer en herhaal de opdracht "show grpc trace all".

De fouten zijn variabel, maar ze tonen ook de component die een probleem kan genereren:

Voorbeeld:

- "'sysdb' ontdekte de 'waarschuwing' voorwaarde 'Een verificateur of EDEDM callback functie teruggestuurd: 'niet gevonden'": Deze fout beschrijft sysdb het moet verzamelen toont tech commando's voor dit proces in de router.

  Het volgende voorbeeld toont de show technologie voor sysdb proces die de fout tonen.

```
show tech-support sysdb
```

  Voor deze output, de fout een component en de fout toont, verzamel om het even welke show technologie-steun die met de fout kan worden verwant die wordt getoond.

- ""YANG framework" ontdekte de 'fatale' voorwaarde 'Operation fail'": Deze fout toont geen

proces in de router, wat betekent dat de query faalt in het model, deel deze informatie aan TAC om te bekijken wat kan falen.

Nadat deze informatie is verzameld, voegt u ook de volgende set opdrachten toe:

In XR VM:

toon tech-support tccpr

show tech-support grpcc

laat het sap voor technische ondersteuning zien

laten zien dat technologie ondersteuning biedt

Over deze vertaling

Cisco heeft dit document vertaald via een combinatie van machine- en menselijke technologie om onze gebruikers wereldwijd ondersteuningscontent te bieden in hun eigen taal. Houd er rekening mee dat zelfs de beste machinevertaling niet net zo nauwkeurig is als die van een professionele vertaler. Cisco Systems, Inc. is niet aansprakelijk voor de nauwkeurigheid van deze vertalingen en raadt aan altijd het oorspronkelijke Engelstalige document (link) te raadplegen.