Configurar gNMI e implementar pYANG no IOS XR

Contents

Introdução

Este documento descreve um resumo sobre o gNMI no Cisco IOS® XR e como usar o PYANG e verificar as árvores de modelos.

Pré-requisitos

Requisitos

A Cisco recomenda que você tenha conhecimento destes tópicos:

- Plataforma Cisco IOS XR.
- Python
- Network Management Protocols (Protocolos de gerenciamento de rede).

Componentes Utilizados

Este documento não se restringe a versões de hardware específicas que se aplicam à versão de 64 bits (eXR).

As informações neste documento foram criadas a partir de dispositivos em um ambiente de laboratório específico. Todos os dispositivos utilizados neste documento foram iniciados com uma configuração (padrão) inicial. Se a rede estiver ativa, certifique-se de que você entenda o impacto potencial de qualquer comando.

Informações de Apoio

definição de gNMI

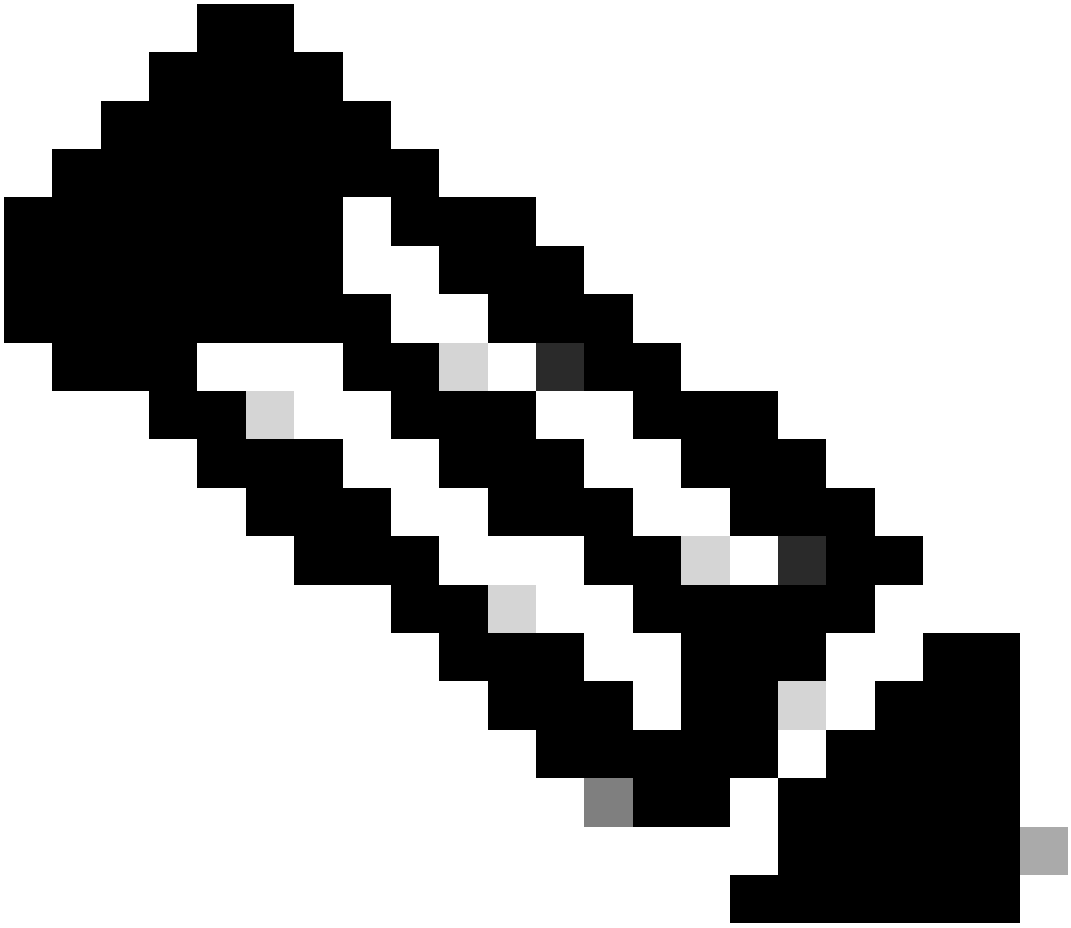
Em geral, existem diferentes protocolos de configuração de rede, de NETCONF, RESTCONF, gNMI (Google Remote Procedure Calls (gRPC), gRPC Network Management Interface), entre outros. Esses modelos são usados para configurar o gerenciamento de dispositivos de rede e sempre têm como objetivo automatizar processos que podem ser mecânicos.

Esses protocolos utilizam diferentes modelos de dados para permitir que os usuários entendam o que o dispositivo de rede processa, em outras palavras, é uma informação estruturada, um esquema, que normaliza as informações e como elas são consumidas pelo dispositivo, neste caso, o roteador.

O gNMI supervisiona o tratamento de dados e fornece RPC (Remote Procedure Calls, chamadas de procedimento remoto) para controlar os diferentes dispositivos na rede.

O gNMI tem quatro funções:

- Capacidades: o gNMI pergunta ao roteador os modelos que estão instalados no roteador, isso é explicado com mais detalhes neste documento.
- Get: cada componente folha na árvore de dados pode ser solicitado ao roteador; essa operação solicita as informações solicitadas.
- Conjunto: Leafs são considerados como variáveis, o que lhes fornece os recursos de mudança, Ajustar assistência de operação sobre isso, permitindo que o usuário atualize um valor no modelo de dados.
- Assinar: Utilizada na telemetria, essa função ajuda a obter dados de um módulo específico no modelo.



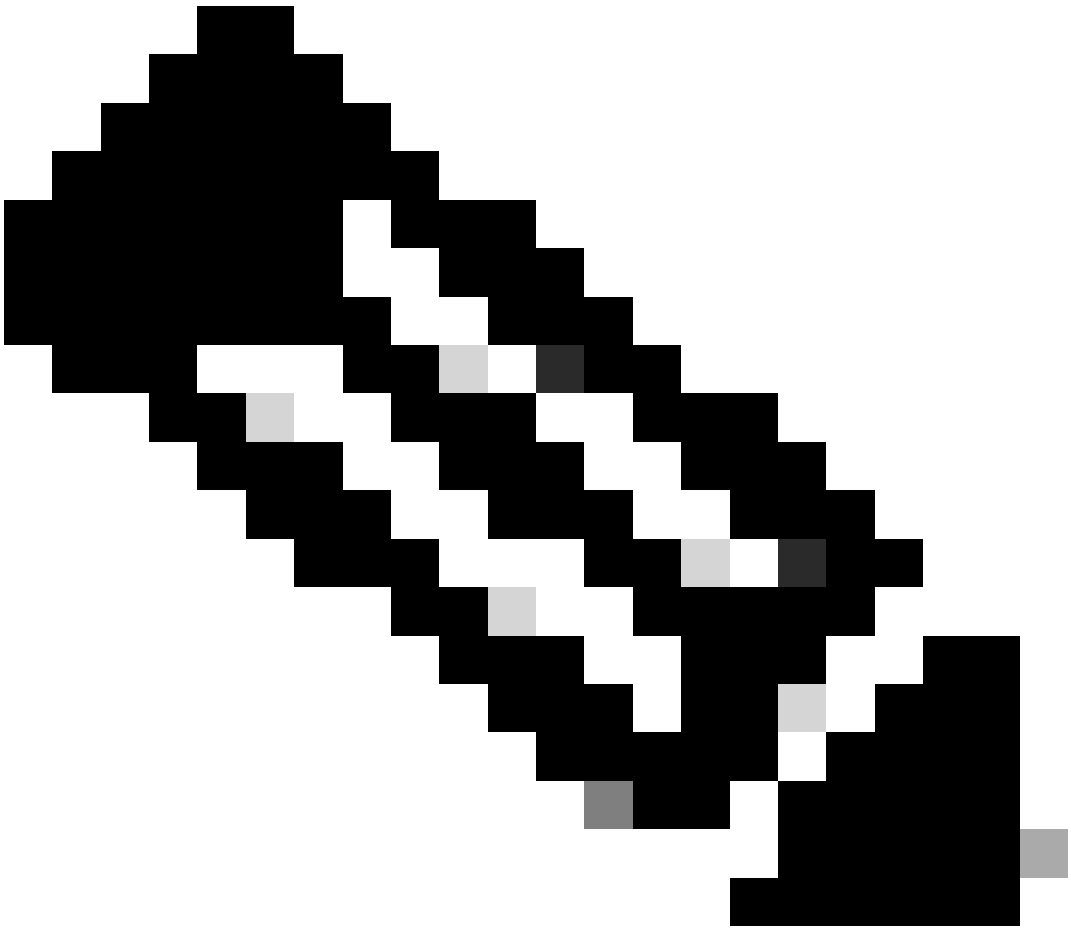
Observação: a Cisco compartilhou muitas informações sobre esse tópico. Para obter mais informações sobre o gRPC, clique no próximo link: xrdocs blog - OpenConfig gNMI

recursos gNMI

Protocolo de gerenciamento de rede	gNMI
Transporte utilizado	HTTP/2
Suportado por	Neutro para o fornecedor
Codificação	Buff de proto

Proto Buff é o método de desserialização e serialização de dados entre dois dispositivos, em que

cada solicitação tem uma resposta.



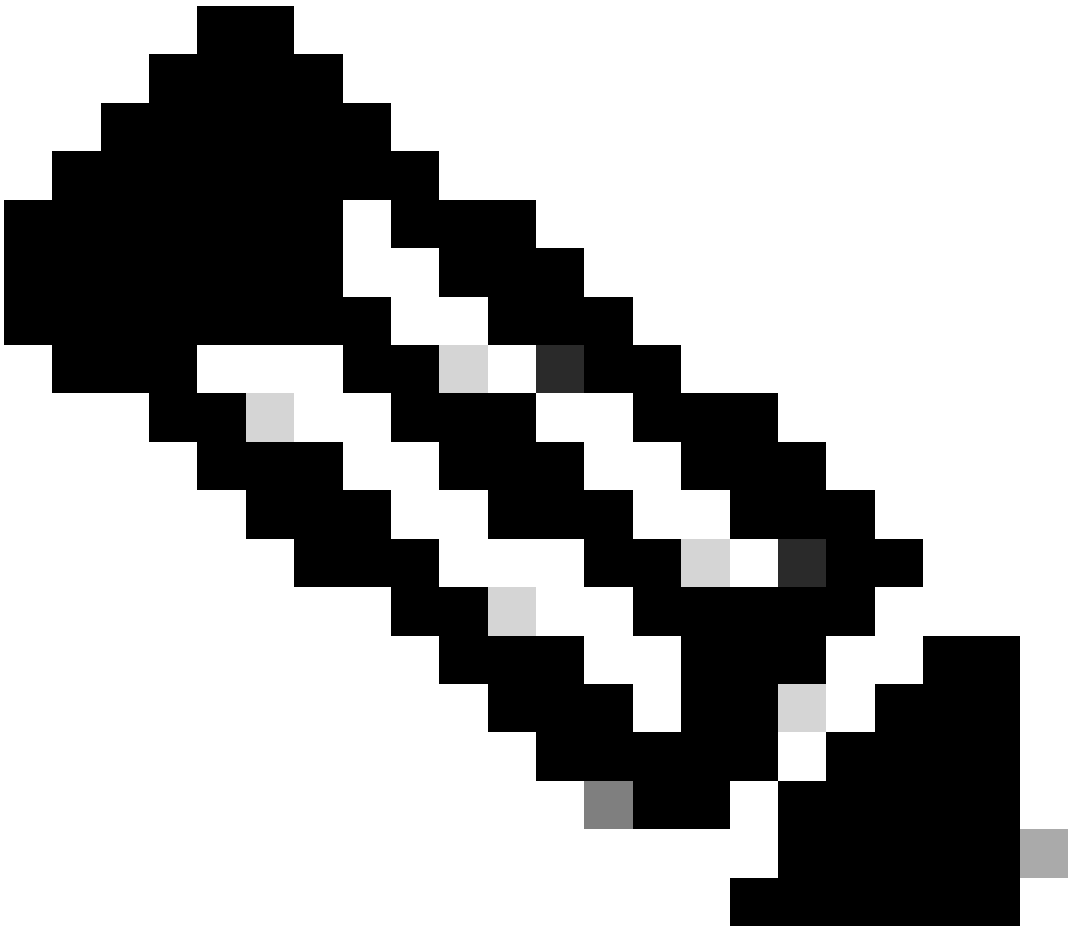
Observação: para obter mais detalhes sobre gRCP e Proto Buff, clique no próximo link: grpc Guide.

Configuração básica do gNMI no Cisco IOS XR

A próxima é a configuração básica do roteador:

```
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
```



Nota: Uma porta pode ser configurada com base na configuração, o padrão, sem usar TLS é 57400, para mais informações clique: github - grpc getting started

pYANG como validador

pYANG é um validador YANG escrito em python. Esta biblioteca para python ajuda na verificação dos modelos YANG e também, conhecendo-os.

Para que isso seja executado como na documentação (documentação pYANG), sugere-se criar um ambiente virtual no computador.

Para o ambiente virtual executar a documentação do venv

É necessário executar:

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

Por exemplo (no terminal MacOS):

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

Para instalar o pYANG neste cd de ambiente virtual para o diretório e colar o seguinte:

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

Para esta demonstração, o python3 pip foi usado, uma vez que o pip install -e foi emitido, ative o venv: source <diretório de ambiente virtual>/bin/ativate (para 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
```

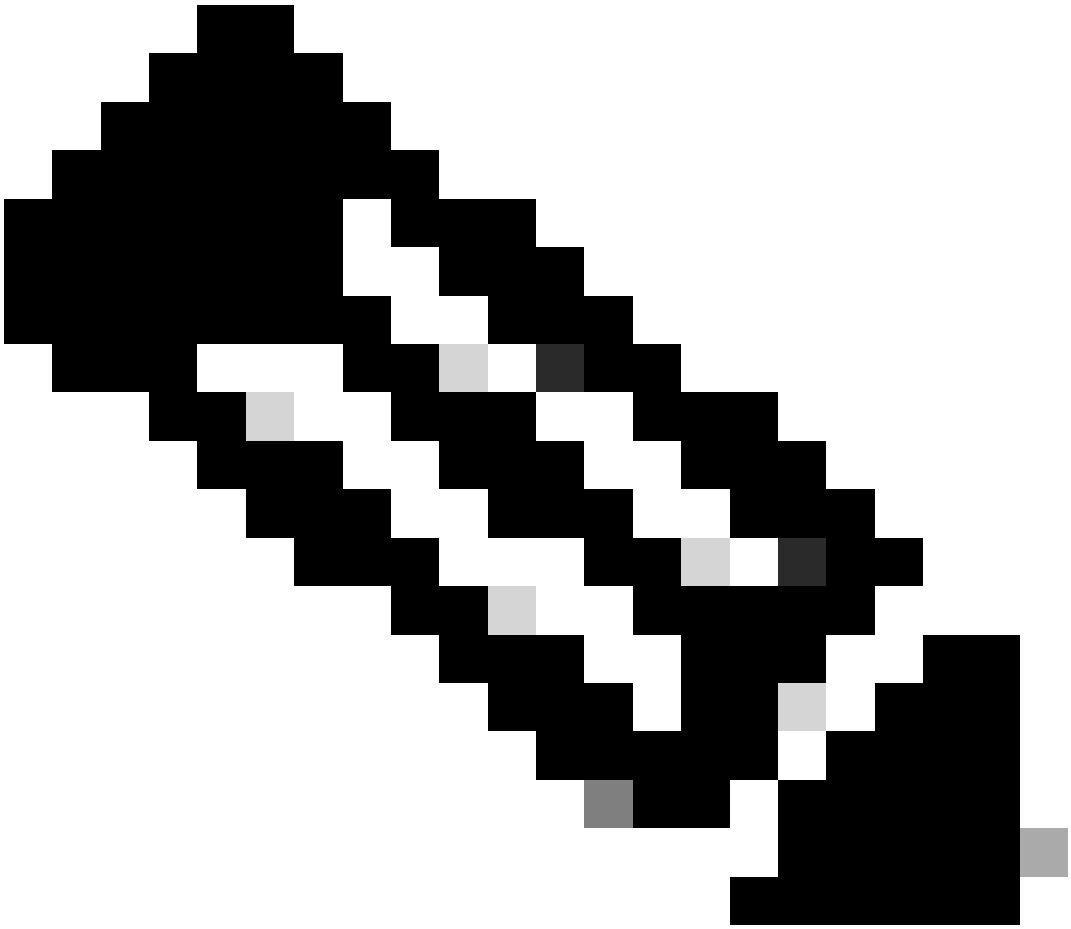
```
% pyang -h
Usage: pyang [options] [<filename>...]

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>
```

Com o pYANG instalado e funcionando, continue com o download de modelos.

No próximo link, há todos os modelos que o Cisco IOS XR executa: [modelos Cisco IOS XR.]

Sugere-se a clonagem git destes modelos no diretório venv com o próximo link de código: https://github.com/YangModels/yang.git

---



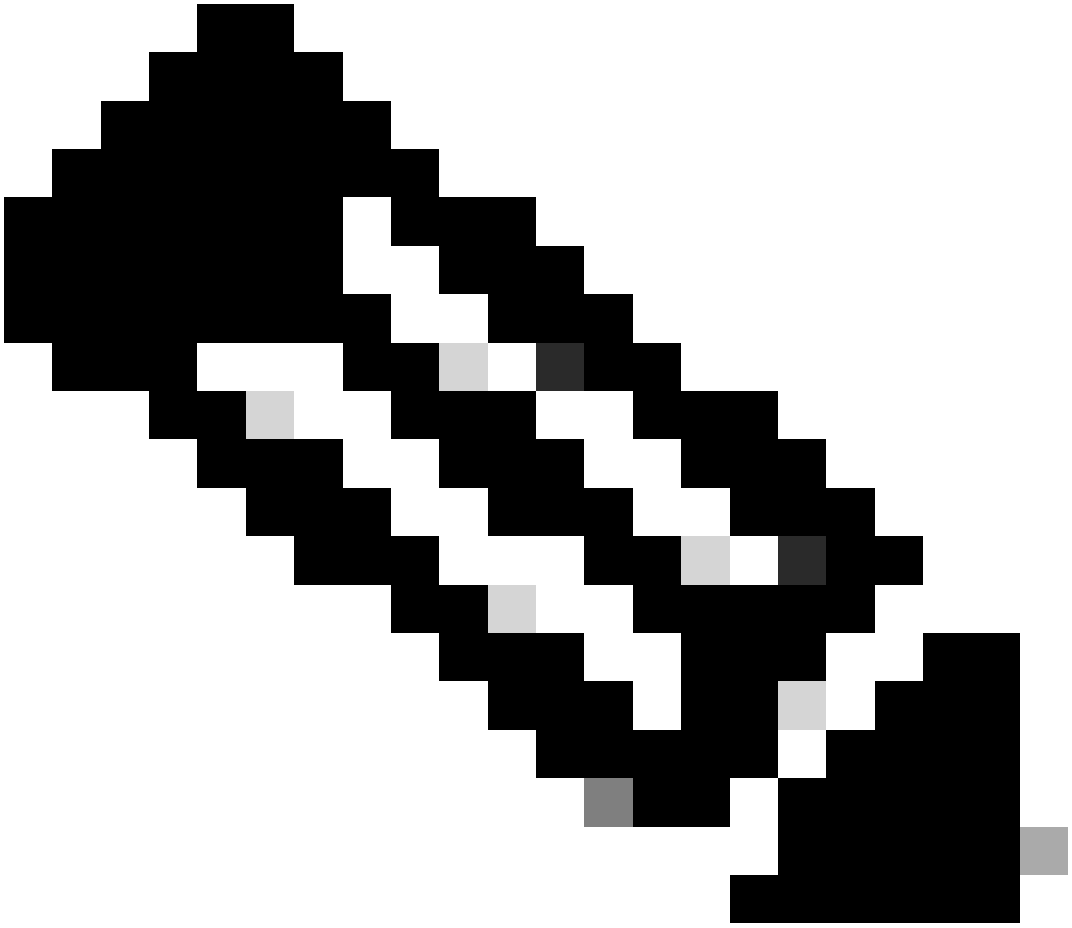
Observação: isso não é feito com o ambiente virtual ativado.

```
% 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.
```

Ative o ambiente virtual novamente e teste a próxima consulta: 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
```



> Observação: observe que os leafs têm um formato de dados como String, uint32, etc. e assim por diante, enquanto as raízes não exibem essa informação. Operações como GET e SET são dedicadas a receber/atualizar esses valores.

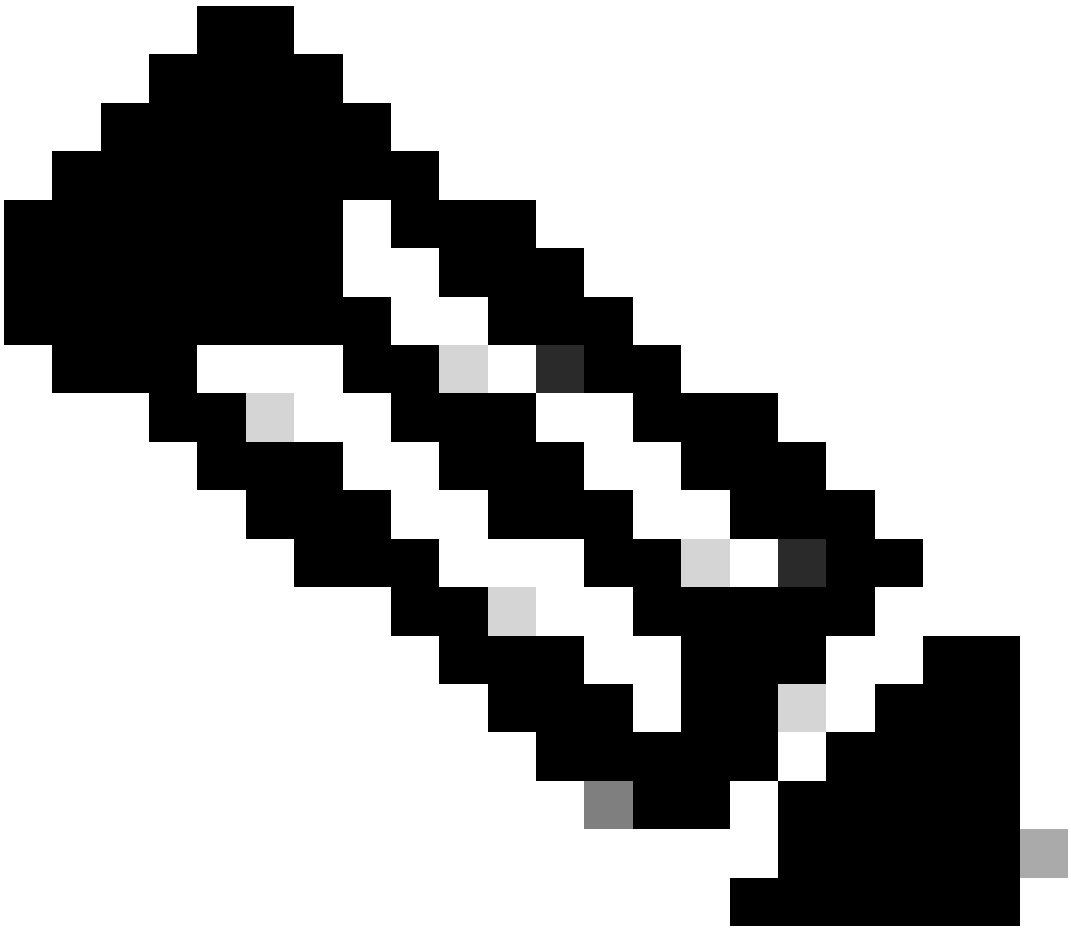
Outra observação é que a maioria dos modelos exige aumentos para ter a configuração completa, na saída CLI há a configuração de gerenciamento de interface básica, caso o IPv4 precise ser exibido, use o seguinte:

```
% 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
```

Nesta consulta, dois modelos são usados: Cisco-IOS-XR-ifmgr-cfg.yang e Cisco-IOS-XR-ipv4-io-cfg.yang, e agora o endereço IPv4 é mostrado como uma folha.



> Observação: caso você veja um erro como: "yang/vendor/cisco/xr/711/Cisco-IOS-XR-ifmgr-cfg.yang:5: error: module "Cisco-IOS-XR-types" not found in search path", adicione —path= no comando.

Com esta opção feita e marcada, qualquer usuário pode solicitar informações com as operações gNMI e alterar a data, para mais exemplos, clique no próximo link: Guia de Configuração de Programabilidade

Caso o usuário queira executar uma API simples, existem ferramentas como: grpcc.

Esta API é instalada via NPM, esta é a ferramenta utilizada no link Guia de Configuração de Programabilidade, disse link compartilha mais exemplos para usuários testarem consultas e respostas.

Troubleshooting:

Para o gNMI, é necessário verificar a consulta antes de coletar qualquer entrada, a maioria das APIs como:

- gnmic
- grpcc
- gRPC

Todos, exiba o erro gerado pelo roteador.

Por exemplo:

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

Ou

```
"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
   }
  ]
```

Esses são erros dependentes de plataforma que precisam ser verificados no roteador. Sugerimos verificar se os comandos na consulta também podem ser emitidos no roteador através da CLI.

Para este tipo de erro, ou qualquer outro relacionado à plataforma Cisco IOS XR, compartilhe as próximas informações com o TAC:

- Consulta usada e operação:

```
{
 "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"
       }
      }
     }
    }
   ]
 }
}
```

- Erro exibido (qualquer um dos mostrados acima).
- Emita o próximo comando:

```
show grpc trace all
```

Teste a consulta algumas vezes e repita o comando "show grpc trace all".

Os erros são variantes, mas também mostram o componente que pode gerar um problema:

Por exemplo:

- "'sysdb' detectou a condição 'aviso' 'Uma função de retorno de chamada de verificador ou EDEDM retornada: 'não encontrada'": Este erro descreve o sysdb necessário para coletar comandos show tech para esse processo no roteador.

  O próximo exemplo mostra o processo show tech for sysdb mostrando o erro.

```
show tech-support sysdb
```

  Para essa saída, o erro exibe um componente e o erro, colete qualquer show tech-support que possa estar relacionado ao erro que está sendo exibido.

- "'YANG framework' detectou a condição 'fatal' 'Operation failed'": Este erro não mostra um processo no roteador, o que significa que a consulta está falhando no modelo, compartilhe

estas informações ao TAC para revisar o que pode estar falhando.

Depois que essas informações forem coletadas, adicione também o próximo conjunto de comandos:

Na VM XR:

show tech-support tctcpsr

show tech-support grpcc

show tech-support gsp

show tech-support

Sobre esta tradução
A Cisco traduziu este documento com a ajuda de tecnologias de tradução automática e humana para oferecer conteúdo de suporte aos seus usuários no seu próprio idioma, independentemente da localização.

Observe que mesmo a melhor tradução automática não será tão precisa quanto as realizadas por um tradutor profissional.

A Cisco Systems, Inc. não se responsabiliza pela precisão destas traduções e recomenda que o documento original em inglês (link fornecido) seja sempre consultado.