在IOS XR中配置gNMI并实施pYANG

目录

简介

先决条件

- 要求
- 使用的组件

背景信息

- gNMI定义
- gNMI功能
- 思科IOS XR中的gNMI基本配置
- pYANG作为验证器
- 故障排除：

简介

本文档简要介绍思科IOS® XR中的gNMI，以及如何使用PYANG和检查模型树。

先决条件

要求

Cisco 建议您了解以下主题：

- Cisco IOS XR平台。
- python。
- 网络管理协议。

使用的组件

本文档不限于适用于64位版本(eXR)的特定硬件版本。

本文档中的信息都是基于特定实验室环境中的设备编写的。本文档中使用的所有设备最初均采用原始（默认）配置。如果您的网络处于活动状态，请确保您了解所有命令的潜在影响。

背景信息

gNMI定义

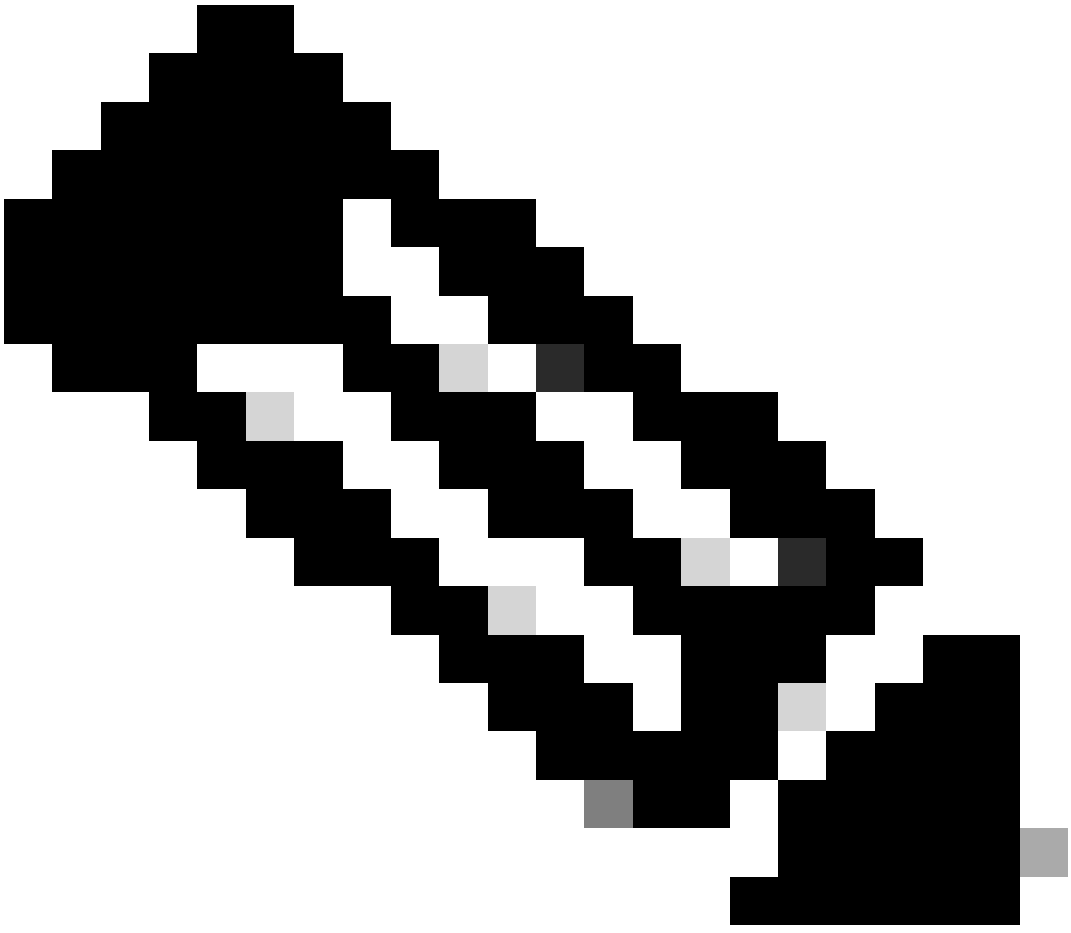
总的来说，有不同的网络配置协议，包括NETCONF、RESTCONF、gNMI(Google远程过程调用(gRPC)、gRPC网络管理接口)等。这些模型用于配置以管理网络设备，并始终致力于实现流程自动化。

这些协议使用不同的数据模型来允许用户了解网络设备进程，换句话说，它是一个结构化的信息，一个方案，用于规范化信息以及设备（在此例中为路由器）消耗信息的方式。

gNMI监控数据处理，并提供RPC（远程过程调用）来控制网络中的不同设备。

gNMI有四个功能：

- 功能：gNMI向路由器询问安装在路由器中的型号，本文档将对此进行详细说明。
- Get：可以向路由器请求数据树中的每个枝叶组件，此操作将请求所请求的信息。
- 集：枝叶被视为变量，为它们提供更改功能，对此集操作辅助，允许用户更新数据模型中的值。
- 预订：在遥测技术中，此功能有助于从模型中的特定模块提取数据。



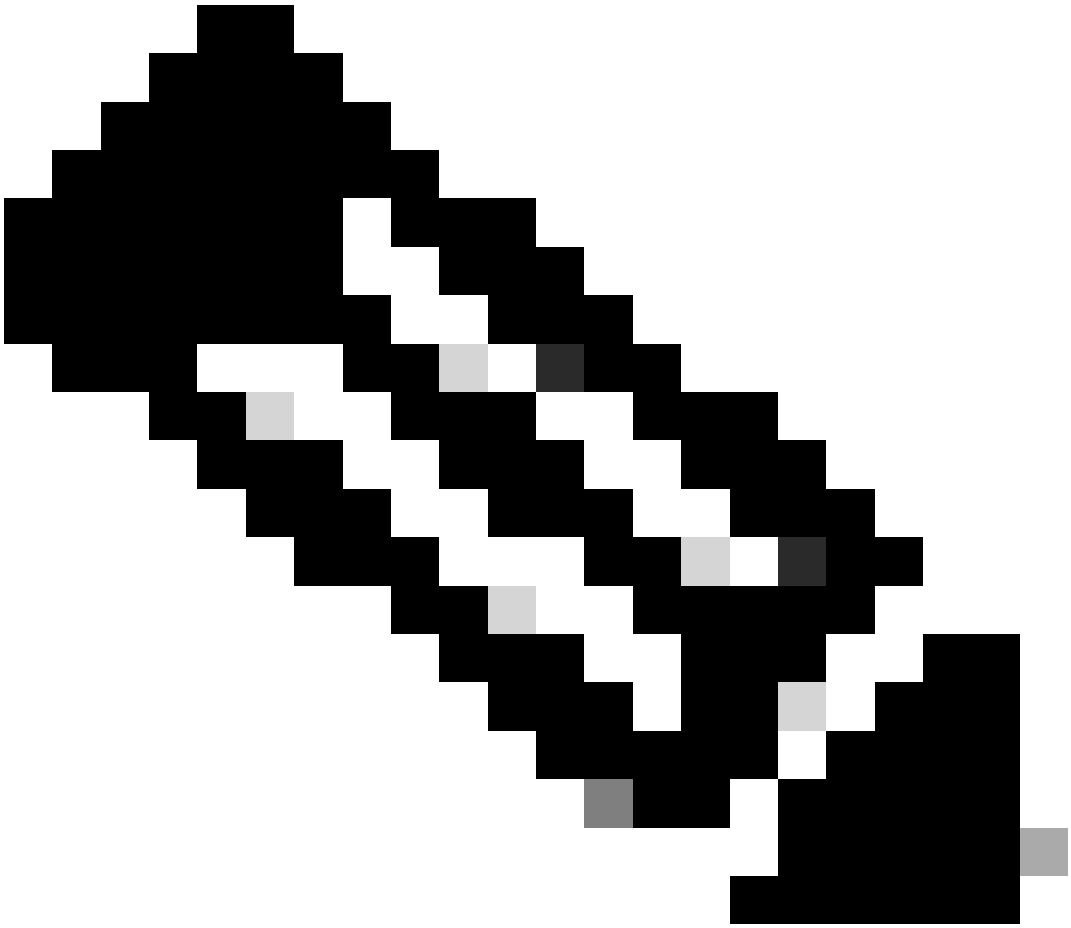
> 注意：思科已共享有关此主题的许多信息。有关gRPC的更多信息，请单击下一个链接：xrdocs博客- OpenConfig gNMI

---

gNMI功能

网络管理协议	gNMI
已利用传输	HTTP/2
支持者	供应商中立性
编码	Proto缓冲区

Proto Buff是在两台设备之间对数据进行反序列化和序列化的中性语言、平台方法，其中每个请求都有一个应答。



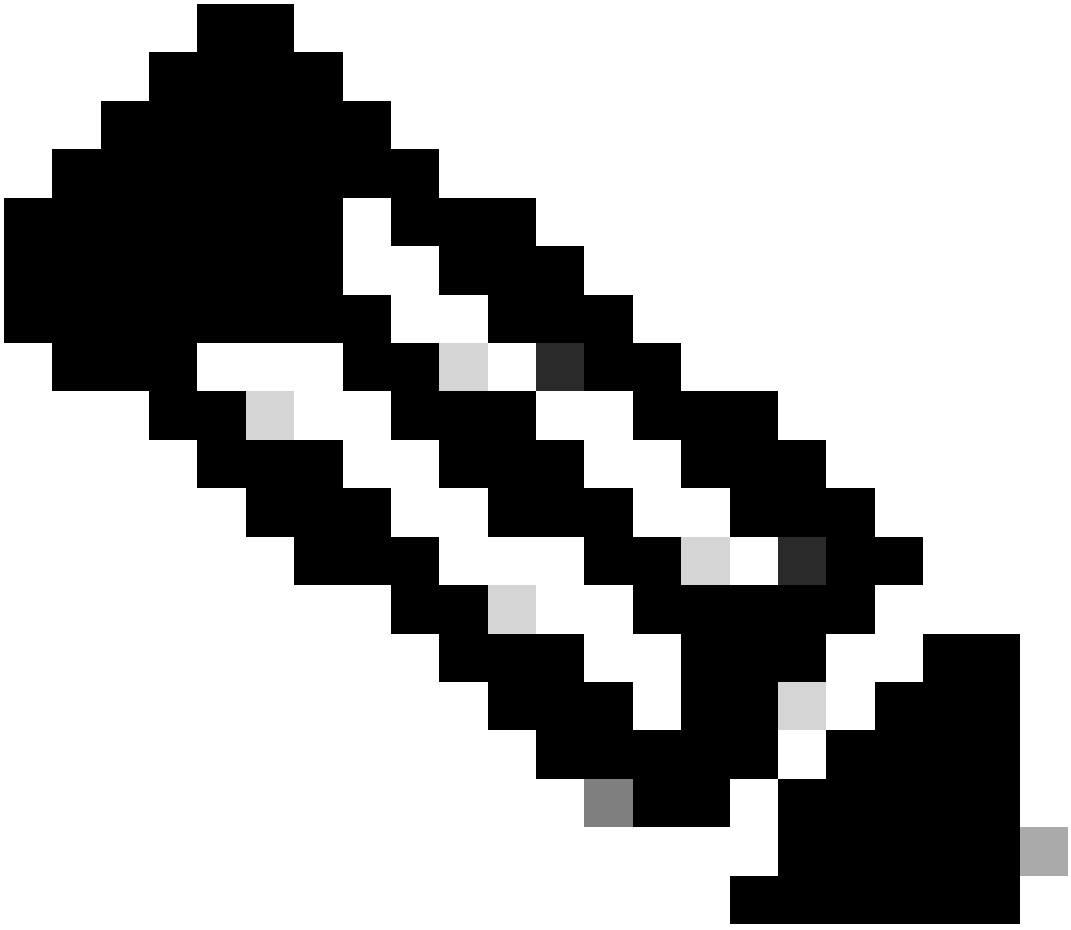
注意：有关gRCP和Proto Buff的更多详细信息，请点击下一个链接：grpc指南。

思科IOS XR中的gNMI基本配置

接下来是路由器的基本配置：

```
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
```



注意：可以基于设置配置端口，默认设置不使用TLS，57400关详细信息，请单击：[github - grpc入门]

---

pYANG作为验证器

pYANG是用python编写的YANG验证器。此python库有助于检查YANG模型并了解它们。

要使此软件以文档(pYANG文档)中的形式运行，建议在计算机中创建虚拟环境。

为使虚拟环境运行venv文档

必须运行：

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

例如（在MacOS终端中）：

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

要在此虚拟环境cd中安装pYANG到目录并粘贴下一个：

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

在本演示中，使用了python3 pip，一旦发出pip install -e，就将激活venv：source <virtual environment directory>/bin/activate（对于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>...]
```

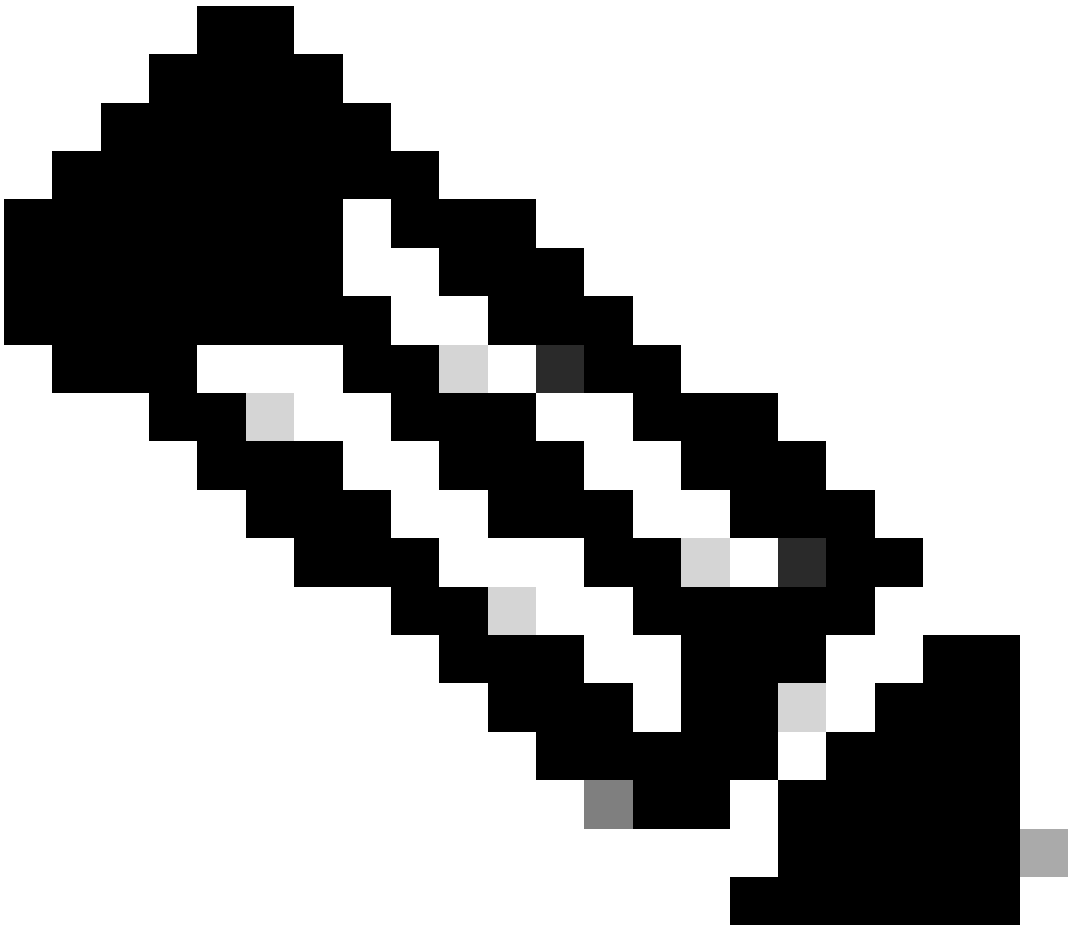
```
Validates the YANG module in <filename> (or stdin), and all its dependencies.

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

安装pYANG并正常工作时，继续下载型号。

在下一个链接中，将显示思科IOS XR运行的所有型号：[思科IOS XR型号。]

建议git clone this model in the venv directory with the next code link：
https://github.com/YangModels/yang.git



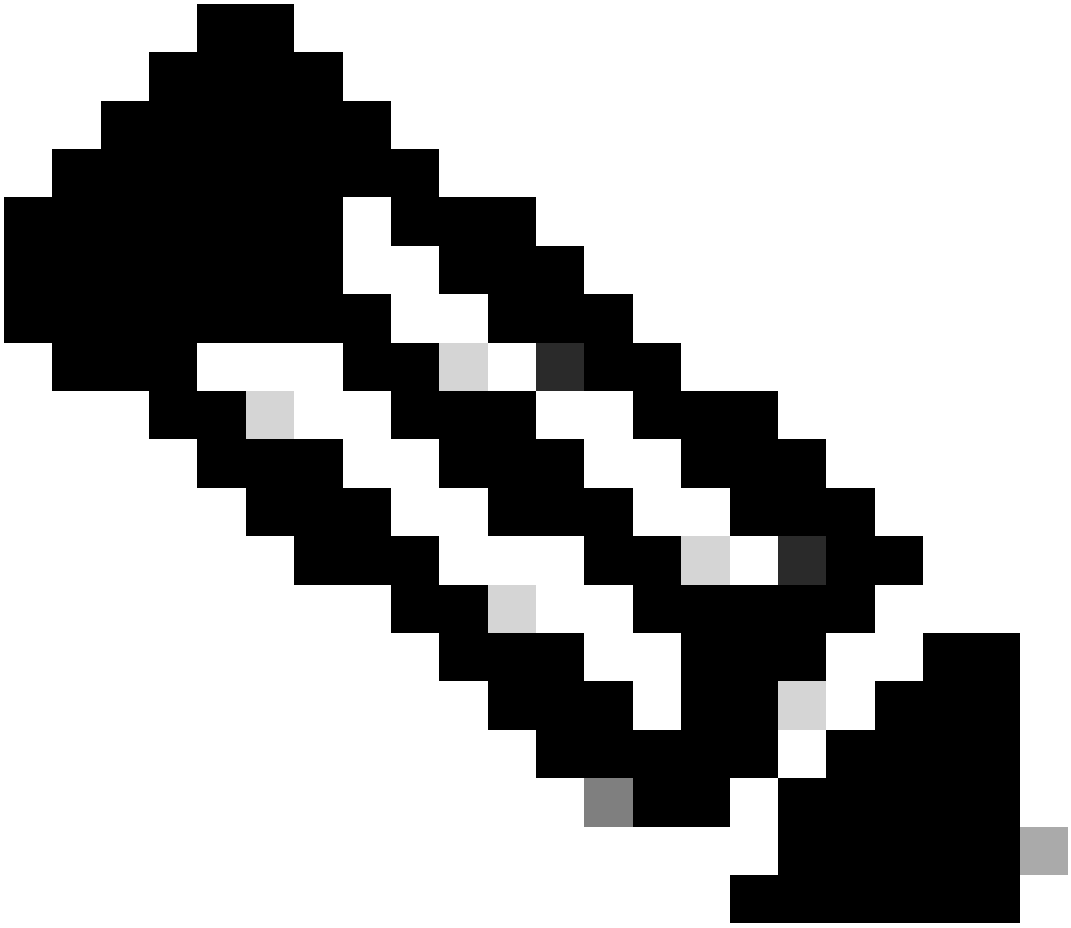
注意：激活虚拟环境后不会执行此操作。

```
% 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.
```

再次激活虚拟环境并测试下一个查询：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
```



注意：请注意，枝叶的数据格式类似于String、uint32等；而根不会显示此信息。GET和SET等操作专用于提取/更新这些值。

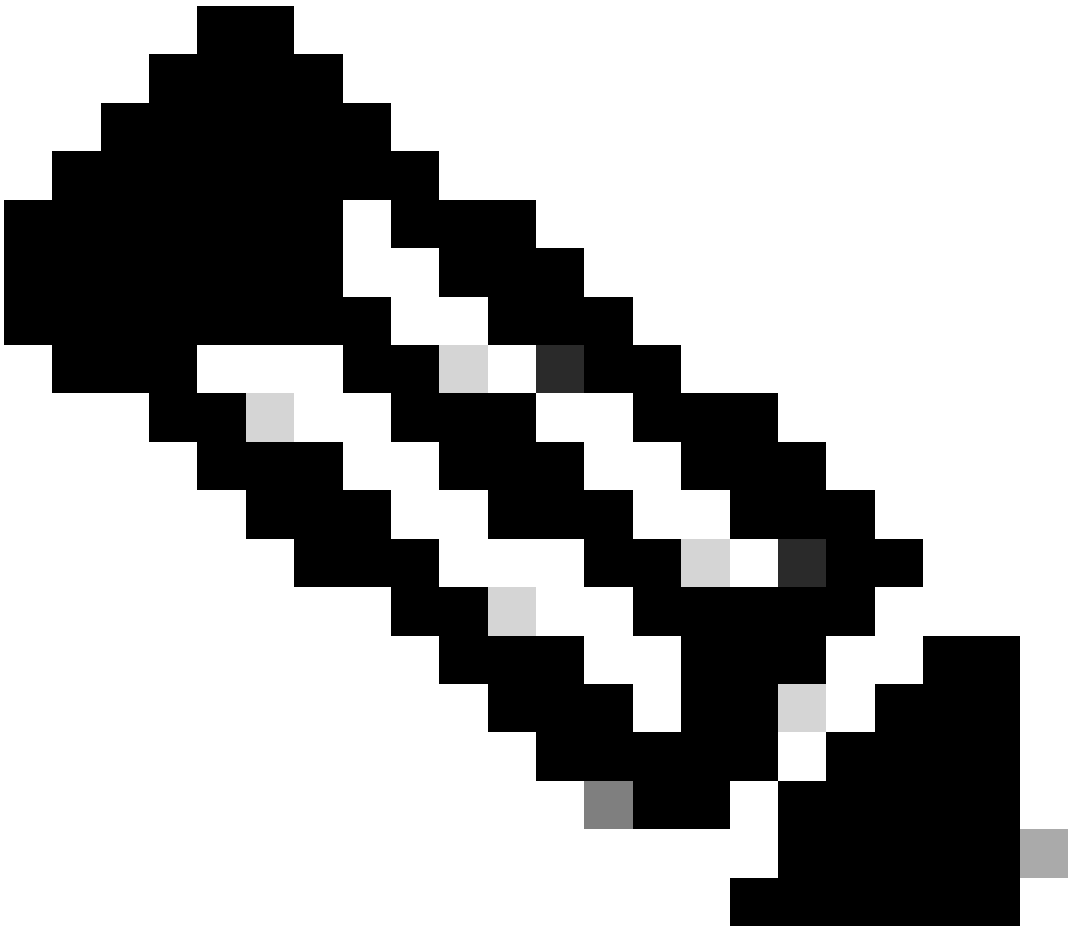
另一个注意是大多数型号都需要增加才能具有完整配置，在CLI输出中提供了基本接口管理配置，如果需要显示IPv4，请使用以下命令：

```
% 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
```

在此查询中使用两个模型：Cisco-IOS-XR-ifmgr-cfg.yang和Cisco-IOS-XR-ipv4-io-cfg.yang，现在IPv4地址显示为叶节点。



> 注意：如果看到以下错误，例如：“yang/vendor/cisco/xr/711/Cisco-IOS-XR-ifmgr-cfg.yang：5： error： module "Cisco-IOS-XR-types" not found in search path”，请在命令中添加—path=。

完成并选中此操作后，任何用户都可以请求有关gNMI操作和更改日期的信息，对于更多示例，请点击以下链接：[可编程性配置指南]

如果用户想要运行一个简单的API，可以使用以下工具：[grpcc。]

此API通过NPM安装，这是在《可编程性配置指南》链接中使用的工具，所述链接分享更多示例供用户测试查询和回复。

故障排除：

对于gNMI，在收集任何条目之前需要检查查询，大多数API如：

- gnmic
- grpcc
- gRPC

All，显示路由器生成的错误。

例如：

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

或者

```
"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
   }
  ]
```

这些是需要在路由器上检查的平台相关错误。建议检查查询中的命令是否也可以通过CLI在路由器中发出。

对于此类错误，或者任何其他与思科IOS XR平台相关的错误，请与TAC共享以下信息：

- 使用的查询和操作：

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

- 显示的错误（以上所示的任何错误）。
- 发出下一条命令：

```
show grpc trace all
```

请多次测试该查询并重复“show grpc trace all”命令。

这些错误各不相同，但它们也显示了可能产生问题的组件：

例如：

- "'sysdb'检测到'warning'条件'返回了验证程序或EDEDM回调函数： 'not found'"：此错误描述了sysdb，在路由器中为此进程收集show tech命令是必需的。

  下一个示例显示了显示错误的show tech for sysdb进程。

```
show tech-support sysdb
```

  对于此输出，错误显示一个组件，并且错误收集可能与所显示的错误相关的任何show tech-support。

- "'YANG framework'检测到'fatal'条件'Operation failed'"：此错误未显示路由器中的进程，这意味着查询在模型中失败，请将此信息共享给TAC以查看可以失败的内容。

收集此信息后，还要添加下一组命令：

在XR VM中：

  show tech-support tctcpsr

show tech-support grpcc

show tech-support gsp

show tech-support

关于此翻译
思科采用人工翻译与机器翻译相结合的方式将此文档翻译成不同语言，希望全球的用户都能通过各自的语言得到支持性的内容。

请注意：即使是最好的机器翻译，其准确度也不及专业翻译人员的水平。

Cisco Systems, Inc. 对于翻译的准确性不承担任何责任，并建议您总是参考英文原始文档（已提供链接）。