Configuración de gNMI e implementación de pYANG en IOS XR

Contenido

Introducción

Prerequisites

- Requirements
- Componentes Utilizados

Antecedentes

- definición gNMI
- funciones gNMI
- Configuración básica de gNMI en Cisco IOS XR
- pYANG como validador
- Resolución de problemas:

Introducción

Este documento describe un resumen sobre gNMI en Cisco IOS® XR y cómo utilizar PYANG y verificar árboles de modelos.

Prerequisites

Requirements

Cisco recomienda que tenga conocimiento sobre estos temas:

- Plataforma Cisco IOS XR.
- pitón.
- Protocolos de administración de red.

Componentes Utilizados

Este documento no se limita a versiones de hardware específicas que se aplican a la versión de 64 bits (eXR).

La información que contiene este documento se creó a partir de los dispositivos en un ambiente de laboratorio específico. Todos los dispositivos que se utilizan en este documento se pusieron en funcionamiento con una configuración verificada (predeterminada). Si tiene una red en vivo, asegúrese de entender el posible impacto de cualquier comando.

Antecedentes

definición gNMI

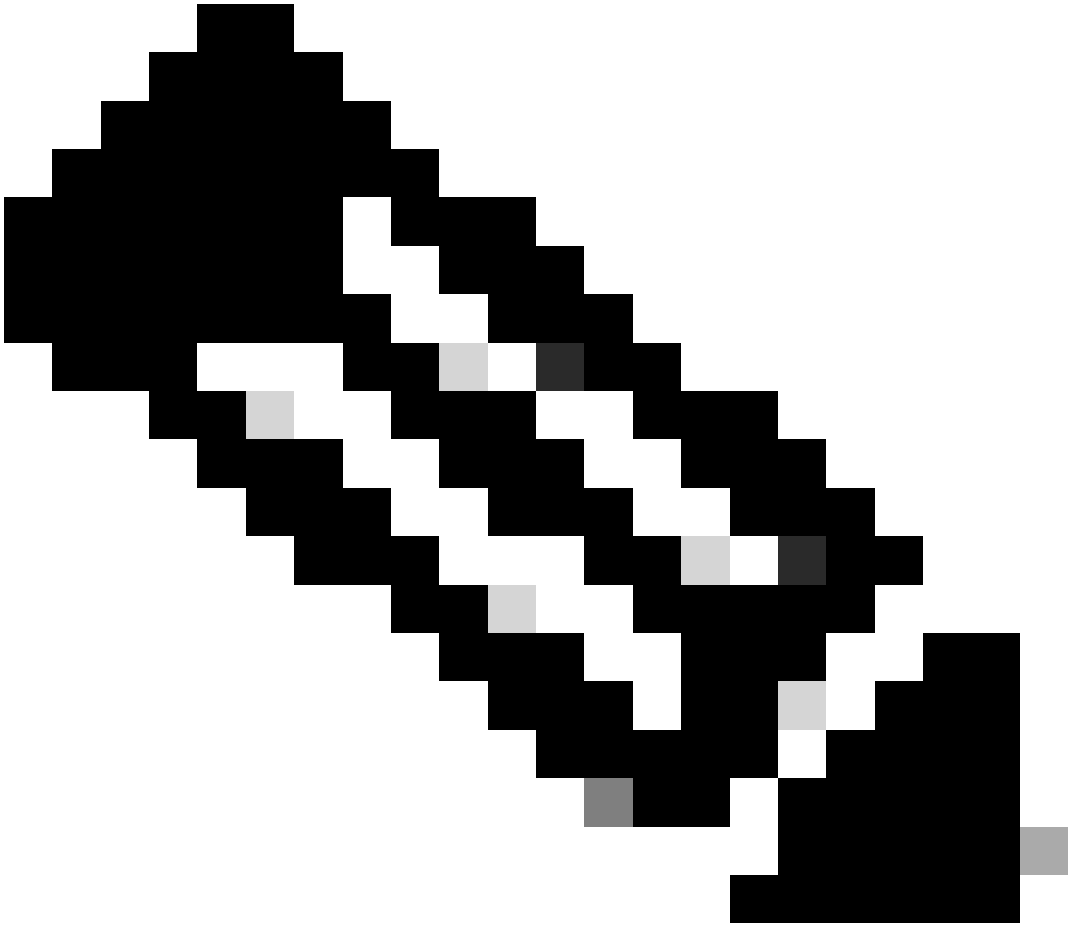
En general, hay diferentes protocolos de configuración de red, desde NETCONF, RESTCONF, gNMI (Google Remote Procedure Calls (gRPC), gRPC Network Management Interface), entre otros. Estos modelos se utilizan para configurar y gestionar los dispositivos de red y siempre pretenden automatizar procesos que pueden ser mecánicos.

Estos protocolos utilizan diferentes modelos de datos para permitir a los usuarios entender lo que el dispositivo de red procesa, en otras palabras, es una información estructurada, un esquema, que normaliza la información y cómo la consume el dispositivo, en este caso, el router.

gNMI supervisa el manejo de los datos y proporciona RPC (llamadas a procedimiento remoto) para controlar los diferentes dispositivos de la red.

gNMI tiene cuatro funciones:

- Capacidades: gNMI pregunta al router los modelos que están instalados en el router, esto se explica más adelante en este documento.
- Get: Cada componente de hoja del árbol de datos se puede solicitar al router, esta operación solicita la información solicitada.
- Establecer: Las hojas se consideran como variables, lo que les proporciona las capacidades de cambio, establecer la asistencia de operación en esto que permite al usuario actualizar un valor en el modelo de datos.
- Suscripción: Esta función, que se utiliza en telemetría, ayuda a extraer datos de un módulo concreto del modelo.



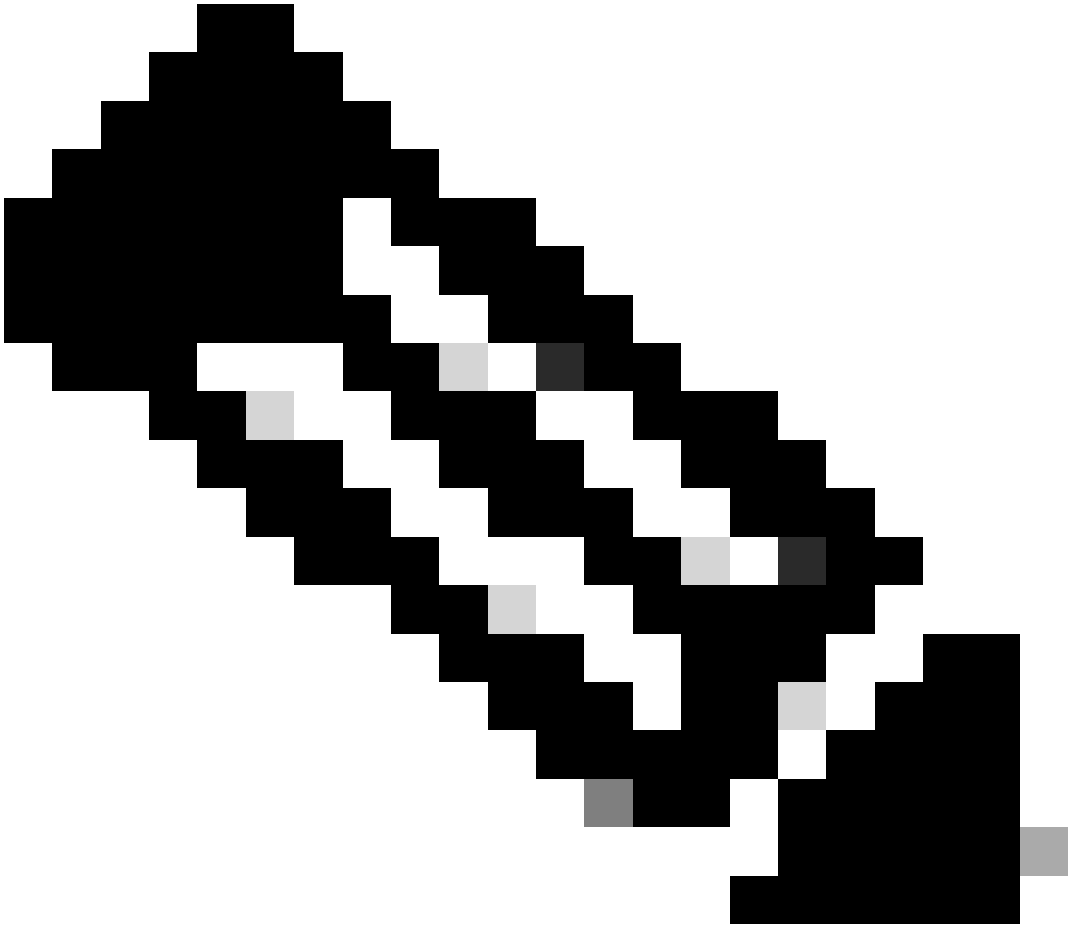
Nota: Cisco ha compartido mucha información sobre este tema. Para obtener más información sobre gRPC, haga clic en el siguiente enlace: [xrdocs blog - OpenConfig gNMI]

funciones gNMI

Protocolo de administración de red	gNMI
Transporte utilizado	HTTP/2
Compatible con	Neutro del proveedor
Codificación	Promoción de proto

Proto Buff es el método neutral de lenguaje y plataforma para deserializar y serializar datos entre

dos dispositivos, en el que cada solicitud tiene una respuesta.



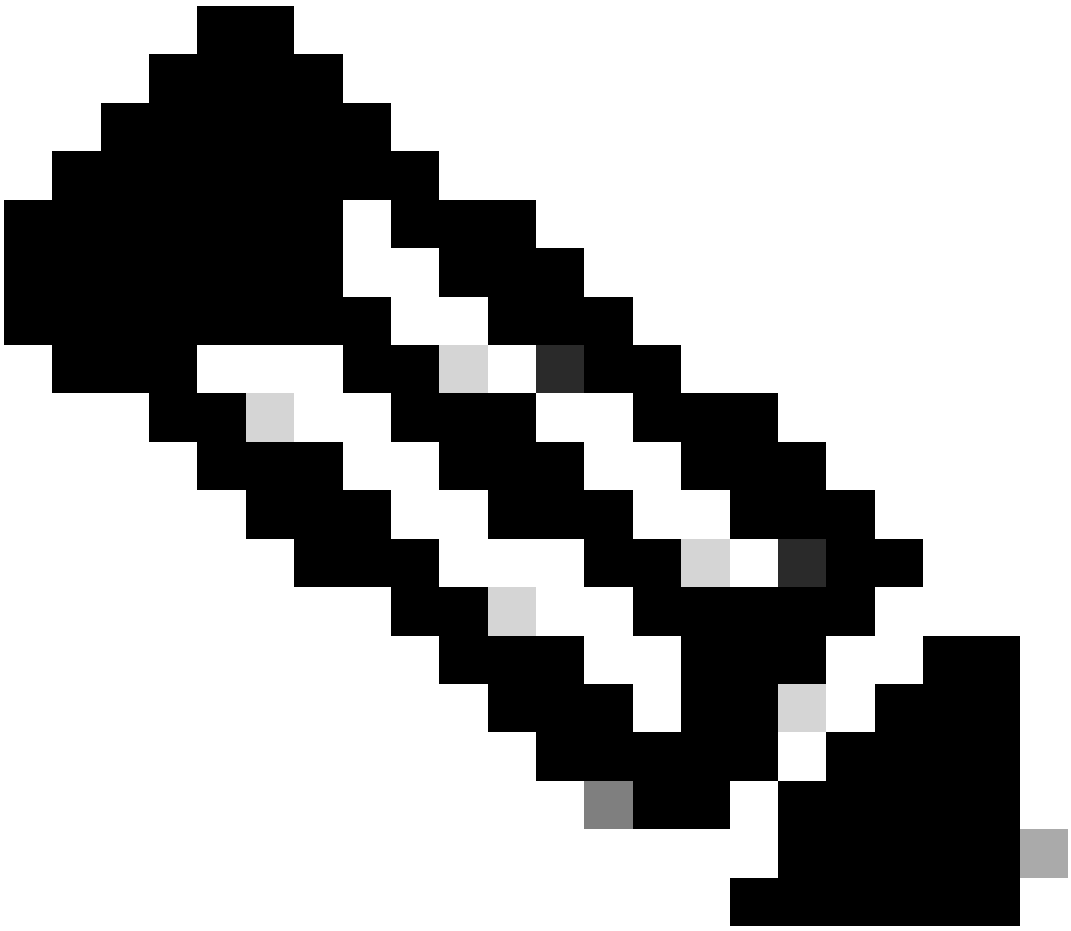
> Nota: Para obtener más información sobre gRCP y Proto Buff, haga clic en el siguiente enlace: grpc Guide.

Configuración básica de gNMI en Cisco IOS XR

La siguiente es la configuración básica del 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
```



Nota: Se puede configurar un puerto en función de la configuración, el valor predeterminado, sin utilizar TLS es 57400, para obtener más información, haga clic en: github - grpc getting started

pYANG como validador

pYANG es un validador YANG escrito en python. Esta biblioteca de ayuda python en la comprobación de los modelos YANG y también, conociéndolos.

Para que esto se ejecute como en la documentación (documentación de pYANG) se sugiere crear un entorno virtual en el equipo.

Para que el entorno virtual ejecute la documentación de venv

Se requiere para ejecutar:

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

Por ejemplo (en el terminal MacOS):

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

Para instalar pYANG en este cd de entorno virtual en el directorio y pegar el siguiente:

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

Para esta demostración, se utilizó python3 pip, una vez que se ejecuta pip install -e, active venv: source <directorio de entorno virtual>/bin/activate (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
```

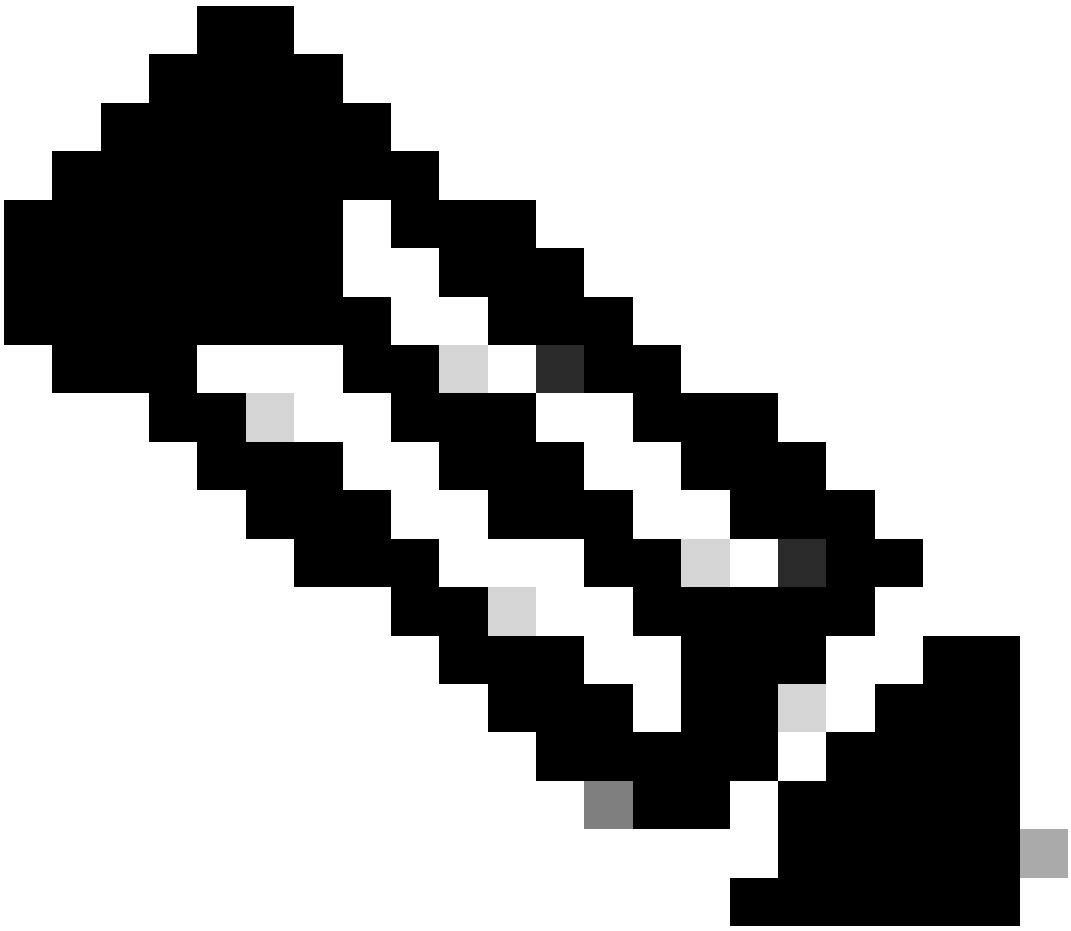
```
% pyang -h
Usage: pyang [options] [<filename>...]

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>
```

Con pYANG instalado y funcionando, continuar con la descarga de modelos.

En el siguiente enlace se encuentran todos los modelos que ejecuta Cisco IOS XR: [modelos Cisco IOS XR.]

Se sugiere clonar git estos modelos en el directorio venv con el siguiente enlace de código: https://github.com/YangModels/yang.git

---

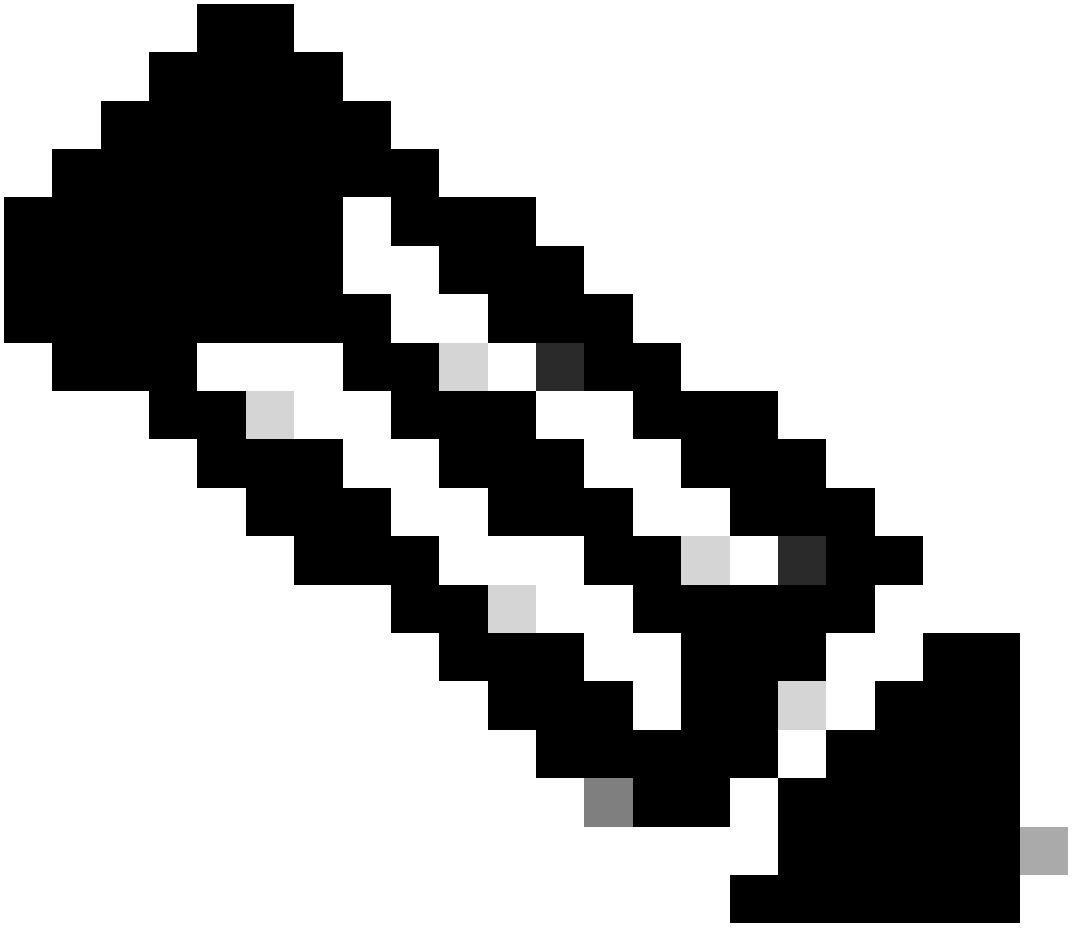


Nota: Esto no se realiza con el entorno virtual activado.

```
% 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.
```

Vuelva a activar el entorno virtual y pruebe la siguiente 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
```



Nota: Observe que las hojas tienen un formato de datos como String, uint32, etc.; mientras que las raíces no muestran esta información. Operaciones como GET y SET se dedican a extraer/actualizar estos valores.

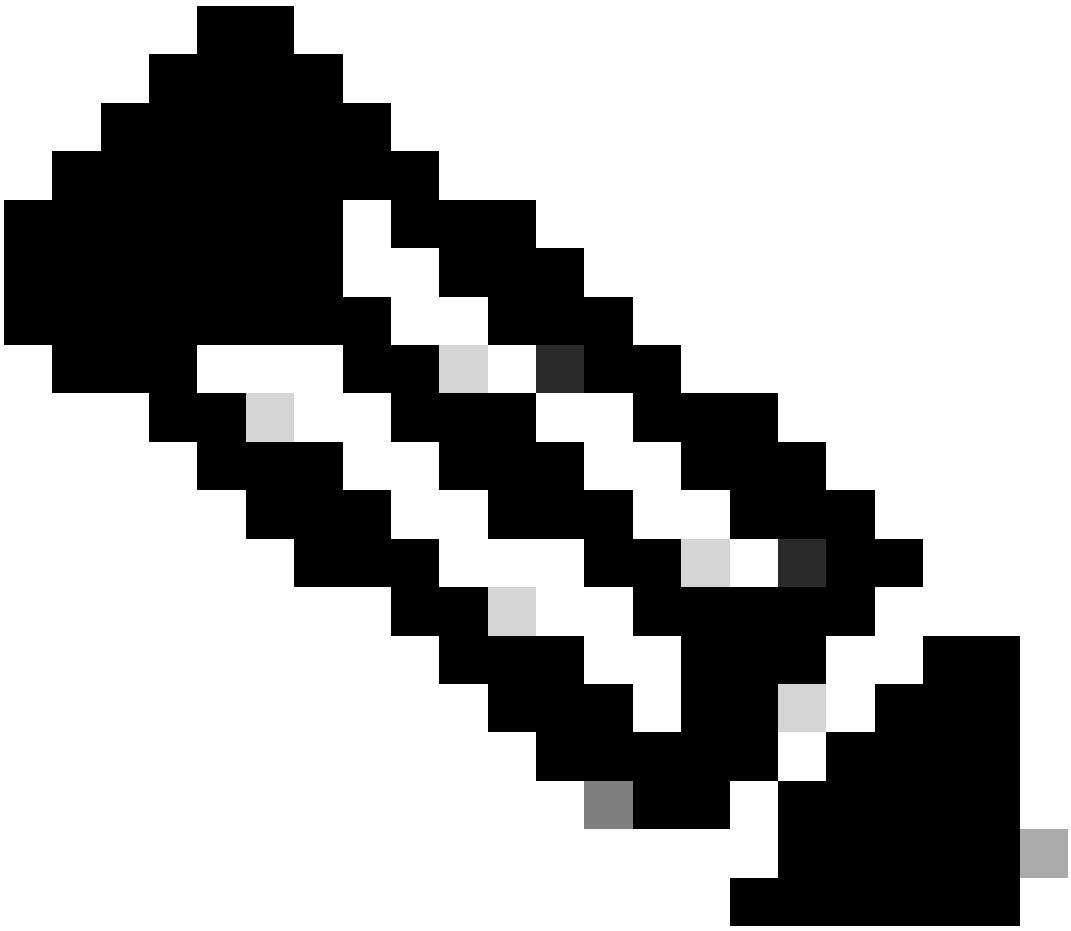
Otra nota es que la mayoría de los modelos requieren aumentos para tener la configuración completa, en la salida CLI está la configuración básica de administración de la interfaz, en caso de que se deba mostrar IPv4, utilice la siguiente:

```
% 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
```

En esta consulta se utilizan dos modelos: Cisco-IOS-XR-ifmgr-cfg.yang y Cisco-IOS-XR-ipv4-io-cfg.yang, y ahora la dirección IPv4 se muestra como una hoja.



> Nota: en caso de que vea un error como: "yang/vendor/cisco/xr/711/Cisco-IOS-XR-ifmgr-cfg.yang:5: error: module "Cisco-IOS-XR-types" not found in search path" agregue el comando -path= en el comando.

Con esto hecho y comprobado, cualquier usuario puede solicitar información con las operaciones gNMI y cambiar la fecha, para más ejemplos haga clic en el siguiente enlace: Guía de Configuración de Programabilidad

En caso de que el usuario quiera ejecutar una API simple, hay herramientas como: grpcc.

Esta API se instala a través de NPM, esta es la herramienta utilizada en el enlace Guía de configuración de programabilidad, dicho enlace comparte más ejemplos para que los usuarios prueben consultas y respuestas.

Resolución de problemas:

Para gNMI es necesario verificar la consulta antes de recopilar cualquier entrada, la mayoría de las API como:

- gnmic
- grpcc
- gRPC

Todos, muestran el error que generó el router.

Por ejemplo:

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

O bien

```
"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
   }
  ]
```

Se trata de errores dependientes de la plataforma que deben comprobarse a lo largo del router. Se sugiere verificar que los comandos de la consulta también se puedan ejecutar en el router a través de la CLI.

Para este tipo de errores, o cualquier otro relacionado con la plataforma Cisco IOS XR, comparta la siguiente información con el TAC:

- Consulta que se utiliza y operación:

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

- Error que se muestra (cualquiera de los anteriores).
- Ejecute el siguiente comando:

```
show grpc trace all
```

Pruebe la consulta un par de veces y repita el comando "show grpc trace all".

Los errores son variantes pero también muestran el componente que puede generar un problema:

Por ejemplo:

- "'sysdb' detectó la condición 'warning' 'Un verificador o función de devolución de llamada EDEDM devolvió: 'not found'": Este error describe sysdb; se requiere recopilar comandos show tech para este proceso en el router.

  El siguiente ejemplo muestra el proceso show tech for sysdb que muestra el error.

```
show tech-support sysdb
```

  Para este resultado, el error muestra un componente y el error, recopila cualquier show tech-support que pueda estar relacionado con el error que se muestra.

- "'YANG framework' detectó la condición 'fatal' 'Operation failed'": Este error no muestra un

proceso en el router, lo que significa que la consulta está fallando en el modelo, comparta esta información con el TAC para revisar lo que puede estar fallando.

Después de recopilar esta información, agregue también el siguiente conjunto de comandos:

En XR VM:

show tech-support tctcpsr

show tech-support grpcc

show tech-support gsp

show tech-support

Acerca de esta traducción
Cisco ha traducido este documento combinando la traducción automática y los recursos humanos a fin de ofrecer a nuestros usuarios en todo el mundo contenido en su propio idioma.

Tenga en cuenta que incluso la mejor traducción automática podría no ser tan precisa como la proporcionada por un traductor profesional.

Cisco Systems, Inc. no asume ninguna responsabilidad por la precisión de estas traducciones y recomienda remitirse siempre al documento original escrito en inglés (insertar vínculo URL).