在IOS XR中配置gNMI並實施pYANG

目錄

簡介

本文檔簡要介紹思科IOS® XR中的gNMI，以及如何使用PYANG和檢查模型樹。

必要條件

需求

思科建議您瞭解以下主題：

- Cisco IOS XR平台。
- 蟒蛇。
- 網路管理協定。

採用元件

本檔案所述內容不限於其適用於64位元版本(eXR)的特定硬體版本。

本文中的資訊是根據特定實驗室環境內的裝置所建立。文中使用到的所有裝置皆從已清除（預設）的組態來啟動。如果您的網路運作中，請確保您瞭解任何指令可能造成的影響。

背景資訊

gNMI定義

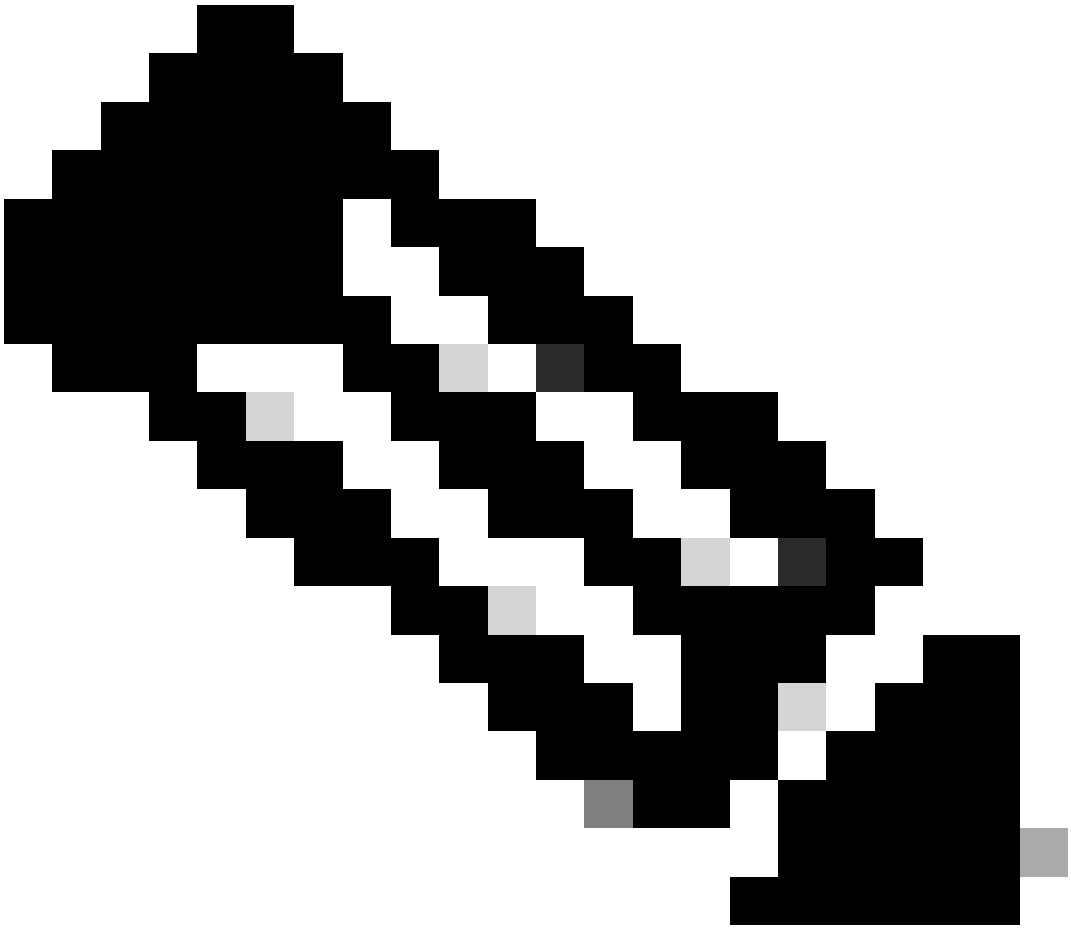
總體來說，網路配置協定各不相同，例如NETCONF、RESTCONF、gNMI(Google遠端過程呼叫(gRPC)、gRPC網路管理介面)。這些模型用於配置以管理網路裝置，並始終致力於使流程實現自動化，而流程可以是機械性的。

這些協定利用不同的資料模型來允許使用者瞭解網路裝置處理的過程，換句話說，它是一個結構化的資訊，即規範化資訊的方案，以及裝置（在此例中為路由器）使用資訊的方式。

gNMI監督資料處理，並提供RPC（遠端過程呼叫）來控制網路中的不同裝置。

gNMI有四個功能：

- 功能：gNMI會詢問路由器中安裝的型號，本文檔對此進行了詳細說明。
- 獲取：可以向路由器請求資料樹中的每個枝葉元件，此操作將請求所請求的資訊。
- 設定：枝葉被視為變數，什麼可提供它們變更功能，設定操作協助可讓使用者更新資料模型中的值。
- 訂閱：此函式用於遙測，可協助從模型中的特定模組擷取資料。



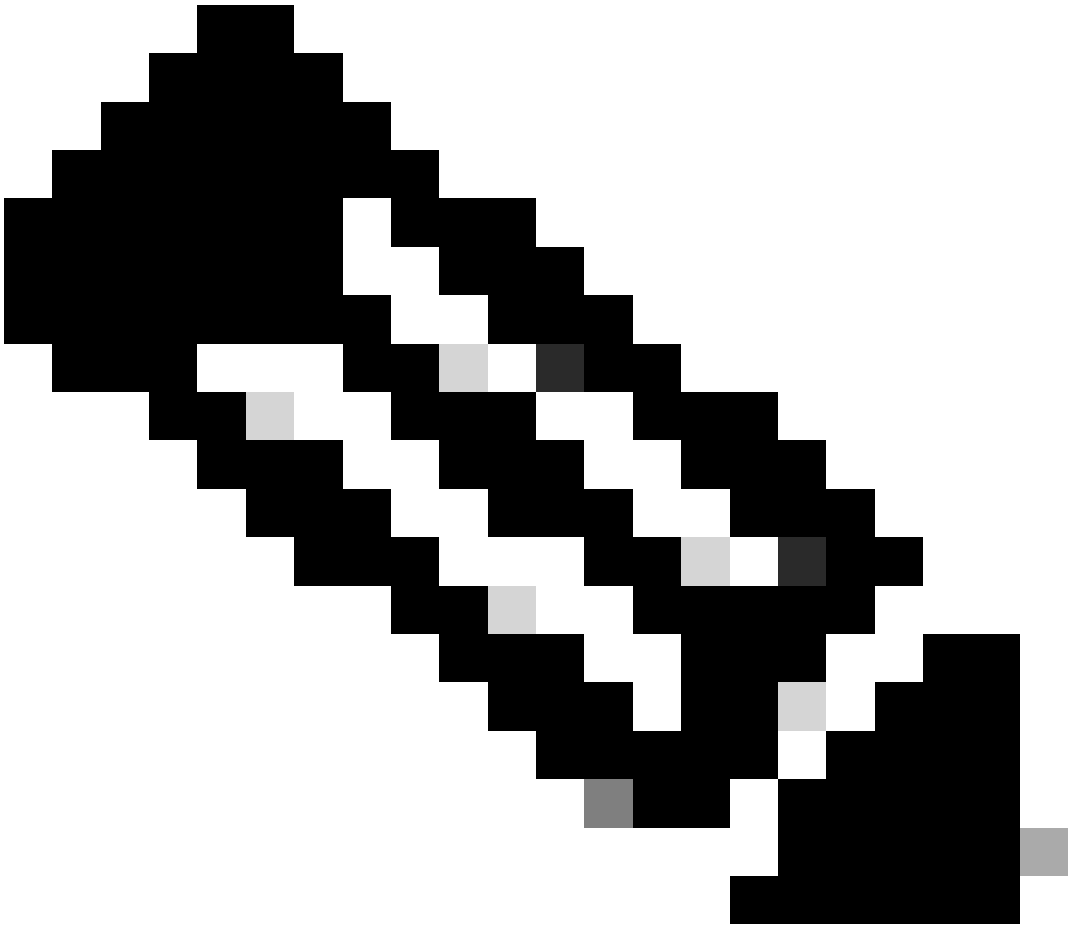
注意：思科在此主題上分享了許多資訊。有關gRPC的詳細資訊，請按一下下一個連結：xrdocs部落格- OpenConfig gNMI

---

gNMI功能

網路管理協定	gNMI
已利用傳輸	HTTP/2
支援者	供應商中立
編碼	Proto緩衝區

Proto Buff是一種語言中立、平台中立的方法，用於解序列化和序列化兩個裝置之間的資料，其中每個請求都有一個應答。

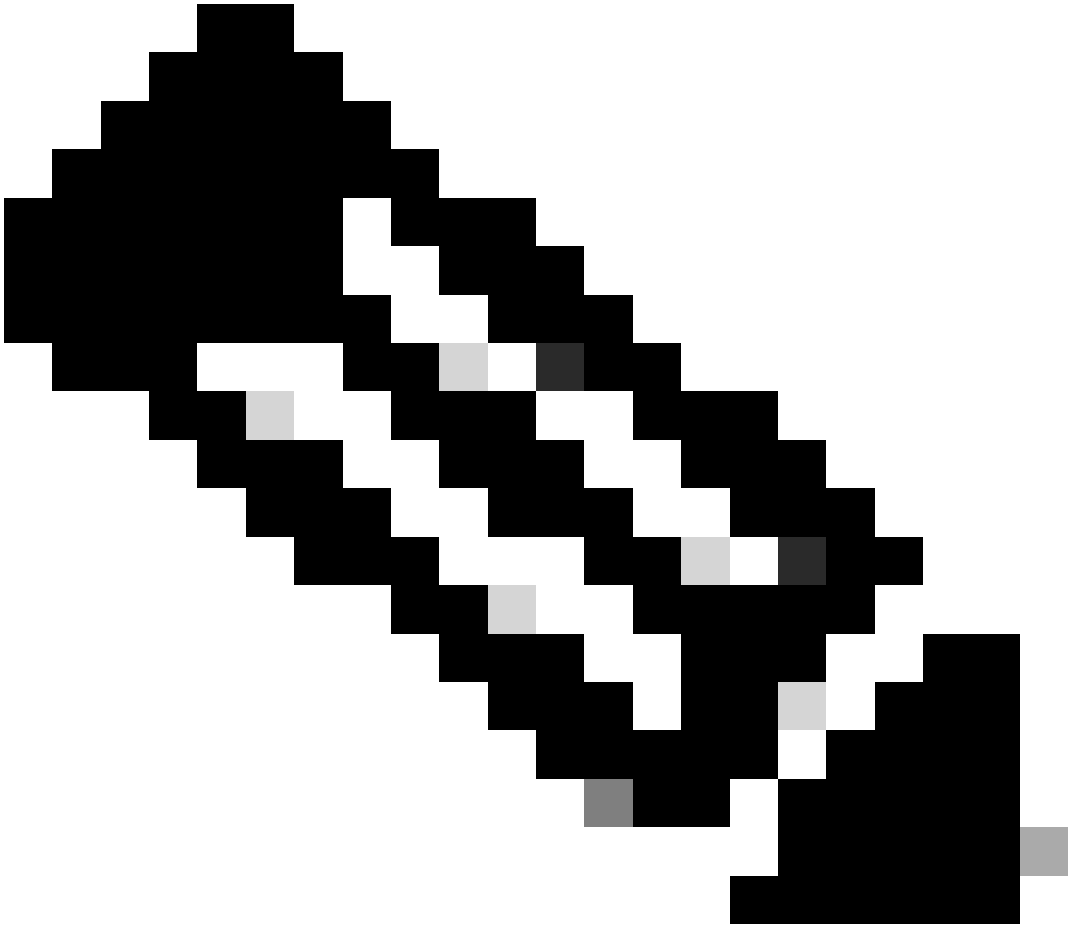


註：有關gRCP和Proto Buff的詳細資訊，請按一下下一個連結：grpc指南。

Cisco IOS XR中的gNMI基本配置

接下來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
```



57400 註：可以基於設定（預設）配置埠，而不使用TLS，有關詳情，請點選：github - grpc入門

---

pYANG作為驗證器

pYANG是用python編寫的YANG驗證器。這個python程式庫可協助您檢查YANG模型，並且瞭解它們。

要使此操作與文檔(pYANG文檔)中一樣運行，建議在電腦中建立虛擬環境。

讓虛擬環境執行venv檔案

必須執行：

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

例如（在MacOS終端中）：

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

要在此虛擬環境cd中安裝pYANG到目錄並貼上下一個：

```
% 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，一旦發出pip install -e，就會啟用venv： source <virtual environment directory>/bin/activate（對於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
```

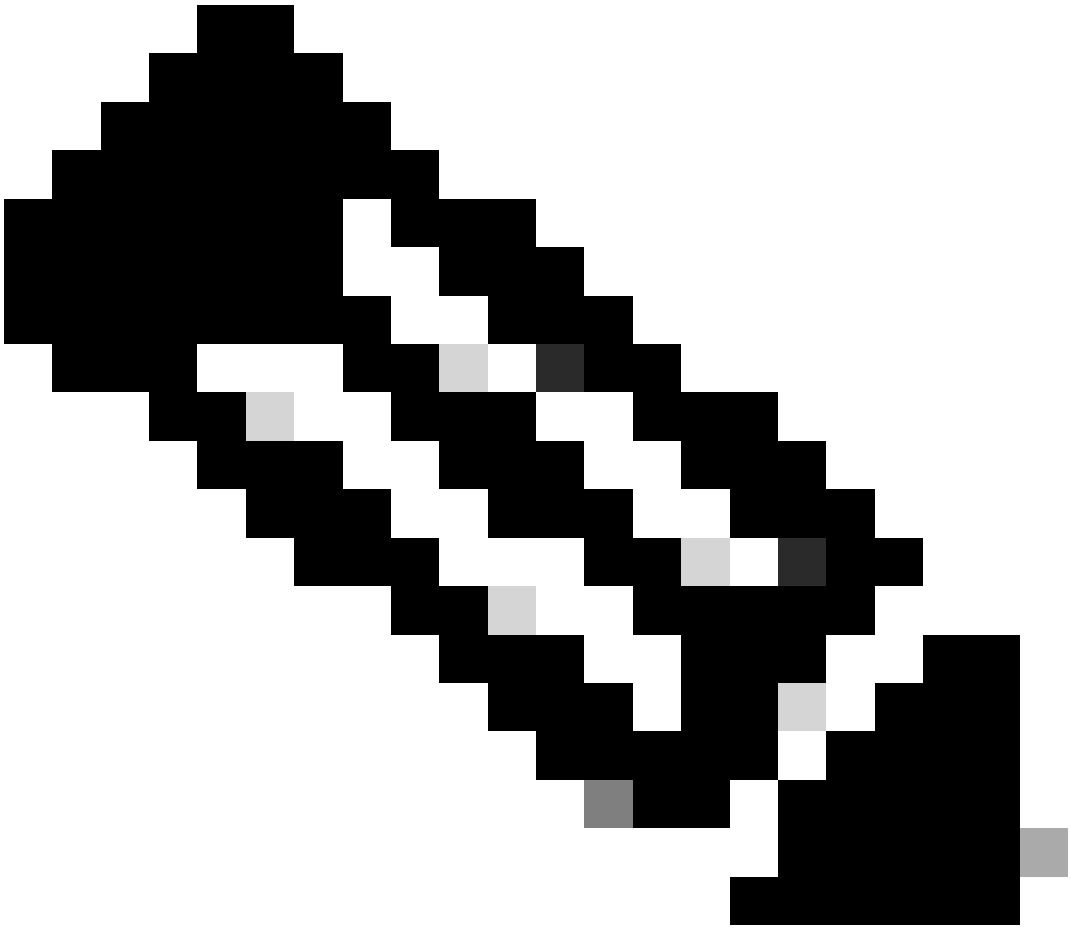
```
Usage: pyang [options] [<filename>...]

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>
```

安裝並使用pYANG後，繼續下載型號。

在下一個連結中，將顯示思科IOS XR運行的所有型號：[思科IOS XR型號。]

建議在venv目錄中克隆此型號，並帶有下一個代碼連結：https://github.com/YangModels/yang.git

---



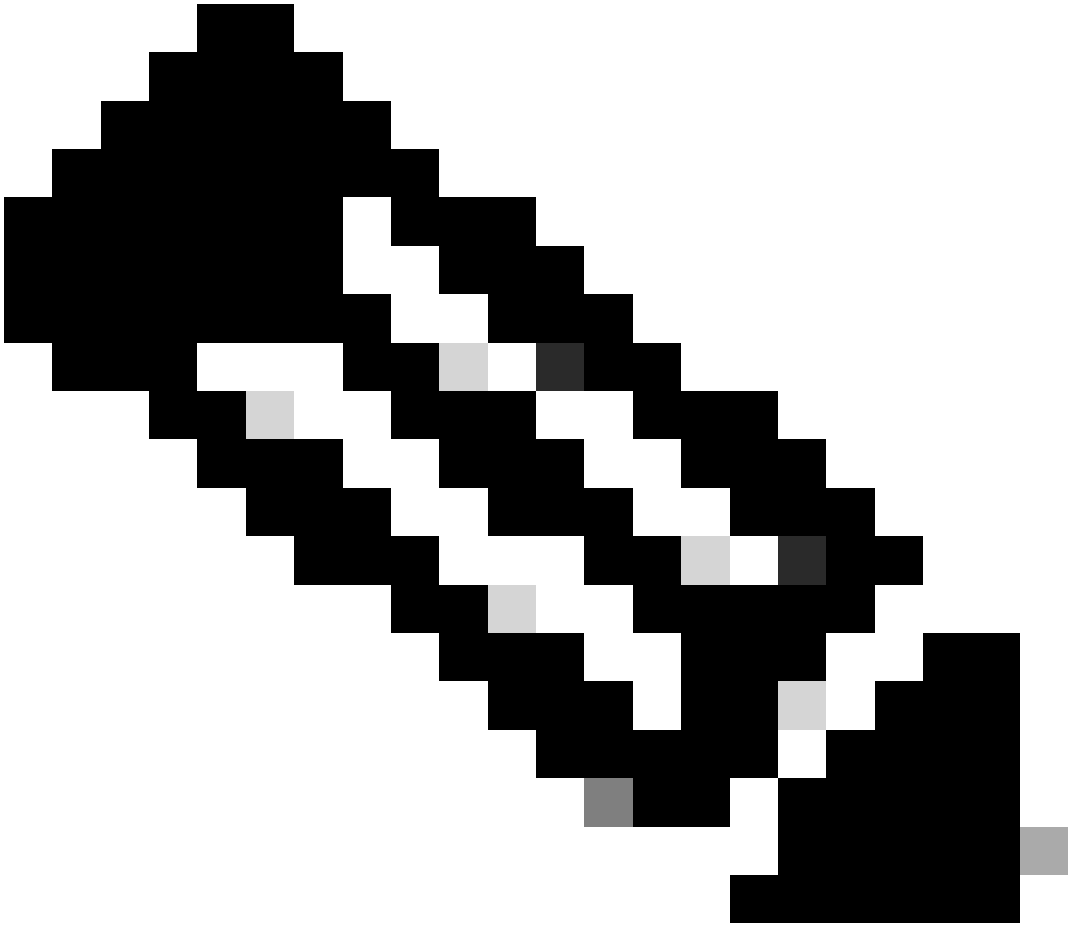
附註：啟動虛擬環境時不會執行此作業。

```
% 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.
```

再次啟用虛擬環境並測試下一個查詢：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：請注意，枝葉的資料格式類似於String、uint32等；而根不顯示此資訊。GET和SET等操作專門用於提取/更新這些值。

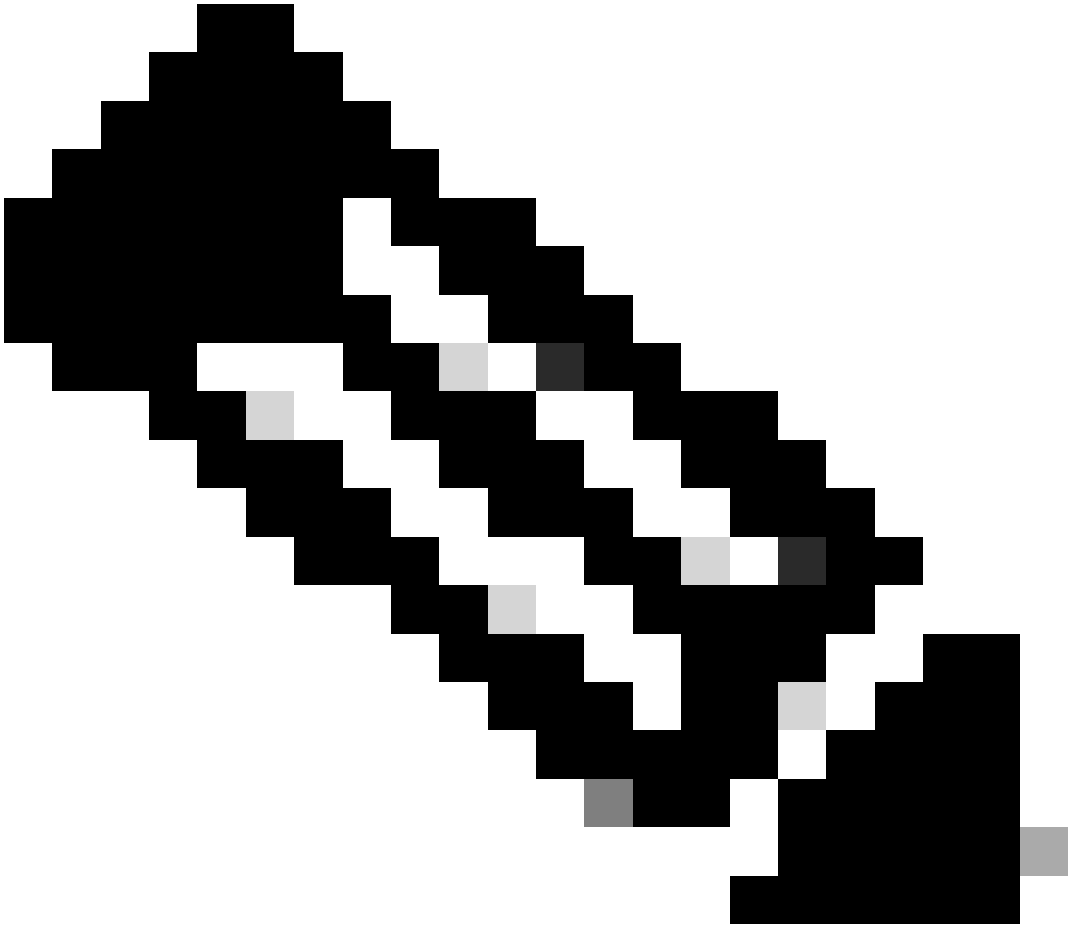
另一個注意事項是，大多數型號都需要增加才能具有完整配置，在CLI輸出中有基本的介面管理配置，如果需要顯示IPv4，請使用以下命令：

```
% 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
```

在此查詢中，使用了兩種模型：Cisco-IOS-XR-ifmgr-cfg.yang和Cisco-IOS-XR-ipv4-io-cfg.yang，現在IPv4地址顯示為枝葉。



> 注意：如果看到類似以下錯誤：「yang/vendor/cisco/xr/711/Cisco-IOS-XR-ifmgr-cfg.yang：5： error： module "Cisco-IOS-XR-types" not found in search path"，請增加—path=在命令中。

完成並檢查後，任何使用者都可以請求有關gNMI操作和更改日期的資訊，對於更多示例，請按一下下一個連結：可程式設計性配置指南

如果使用者想要運行一個簡單的API，可以使用如下工具：grpcc。

此API是透過NPM安裝的，這是《可程式設計性配置指南》連結中使用的工具，所述連結共用更多示例供使用者測試查詢和回覆。

疑難排解:

對於gNMI，在收集任何條目之前需要檢查查詢，大多數API如下：

- gnmic
- grpcc
- gRPC

All，顯示路由器生成的錯誤。

舉例來說：

```
"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

```
"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
   }
  ]
```

這些是需要在路由器上檢查的平台相關錯誤。建議檢查查詢中的命令是否也可以透過CLI在路由器中發出。

對於此類錯誤，或與思科IOS XR平台相關的任何其他錯誤，請與TAC共用以下資訊：

- 使用的查詢和操作：

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

- 顯示的錯誤（上述任何一項）。
- 發出下一個命令：

```
show grpc trace all
```

請對該查詢進行幾次測試並重複「show grpc trace all」命令。

這些錯誤各不相同，但也會顯示可產生問題的元件：

舉例來說：

- "'sysdb'偵測到'warning'條件'傳回'verifier或EDEDM回呼函式： 'not found'"：此錯誤說明sysdb路由器中必須為此程式收集show tech命令。

  下一個示例顯示了顯示錯誤的show tech for sysdb進程。

```
show tech-support sysdb
```

  對於此輸出，錯誤顯示一個元件，並且錯誤，收集可能與所顯示的錯誤相關的任何show tech-support。

- 「'YANG framework'檢測到'fatal' condition 'Operation failed'」：此錯誤未顯示路由器中的進程，這意味著查詢在模型中失敗，請與TAC共用此資訊，以檢查可能失敗的情況。

收集此資訊後，另新增下一組指令：

在XR VM中：

  show tech-support tctcpsr

show tech-support grpcc

show tech-support gsp

show tech-support

關於此翻譯

思科已使用電腦和人工技術翻譯本文件，讓全世界的使用者能夠以自己的語言理解支援內容。請注意，即使是最佳機器翻譯，也不如專業譯者翻譯的內容準確。Cisco Systems, Inc. 對這些翻譯的準確度概不負責，並建議一律查看原始英文文件（提供連結）。