

# Configurazione di GMI e implementazione di YPNG in IOS XR

## Sommario

---

[Introduzione](#)

[Prerequisiti](#)

[Requisiti](#)

[Componenti usati](#)

[Premesse](#)

[definizione di gNMI](#)

[Funzioni gNMI](#)

[Configurazione di base di gNMI in Cisco IOS XR](#)

[Yang come validator](#)

[Risoluzione dei problemi:](#)

---

## Introduzione

Questo documento descrive una descrizione del GMI in Cisco IOS® XR e come utilizzare PYANG e controllare gli alberi dei modelli.

## Prerequisiti

### Requisiti

Cisco raccomanda la conoscenza dei seguenti argomenti:

- piattaforma Cisco IOS XR.
- pitone.
- Protocolli di gestione della rete.

### Componenti usati

Il documento può essere consultato per tutte le versioni hardware applicate alle versioni a 64 bit (eXR).

Le informazioni discusse in questo documento fanno riferimento a dispositivi usati in uno specifico ambiente di emulazione. Su tutti i dispositivi menzionati nel documento la configurazione è stata ripristinata ai valori predefiniti. Se la rete è operativa, valutare attentamente eventuali conseguenze derivanti dall'uso dei comandi.

## Premesse

## definizione di gNMI

Nel complesso, esistono diversi protocolli di configurazione della rete, tra cui NETCONF, RESTCONF, gNMI (Google Remote Procedure Call (gRPC), gRPC Network Management Interface). Questi modelli vengono utilizzati per configurare e gestire i dispositivi di rete e hanno sempre lo scopo di automatizzare i processi che possono essere meccanici.

Questi protocolli utilizzano modelli di dati diversi per consentire agli utenti di comprendere il processo del dispositivo di rete, in altre parole, si tratta di informazioni strutturate, di uno schema, che normalizzano le informazioni e come vengono consumate dal dispositivo, in questo caso il router.

gNMI controlla la gestione dei dati e fornisce RPC (Remote Procedure Call) per controllare i diversi dispositivi nella rete.

gNMI ha quattro funzioni:

- **Funzionalità:** gNMI chiede al router quali modelli sono installati sul router, come spiegato più avanti in questo documento.
- **Scarica:** Ogni componente foglia nella struttura dati può essere richiesto al router. Questa operazione richiede le informazioni richieste.
- **Imposta:** I fogli sono considerati come variabili, ciò che fornisce loro le funzionalità di modifica, Imposta assistenza operativa su questo consente all'utente di aggiornare un valore nel modello di dati.
- **Sottoscrivi:** Utilizzata nella telemetria, questa funzione facilita il recupero dei dati da un particolare modulo del modello.



Nota: Cisco ha condiviso molte informazioni su questo argomento. Per ulteriori informazioni su gRPC, fare clic sul collegamento successivo: [blog xrdocs - OpenConfig GMI](#)

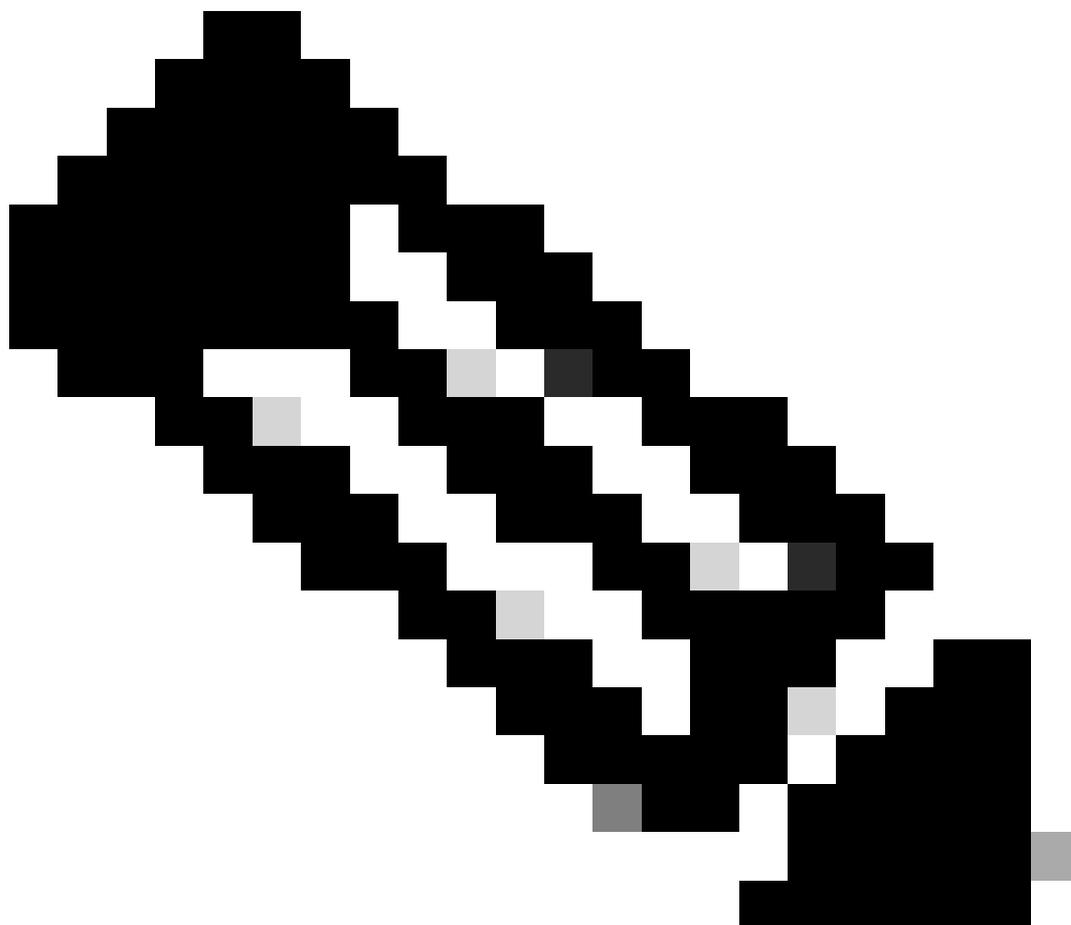
---

## Funzioni gNMI

Protocollo di gestione della rete	gNMI
Trasporto utilizzato	HTTP/2
Supportato da	Indipendente dal fornitore
Codifica	Proto Buff

Proto Buff è il metodo indipendente dalla lingua e dalla piattaforma per la deserializzazione e la serializzazione dei dati tra due dispositivi, in cui ogni richiesta ha una risposta.

---



Nota: Per ulteriori informazioni su RCP e Proto Buff, fare clic sul collegamento successivo:  
[Guida di GRPC.](#)

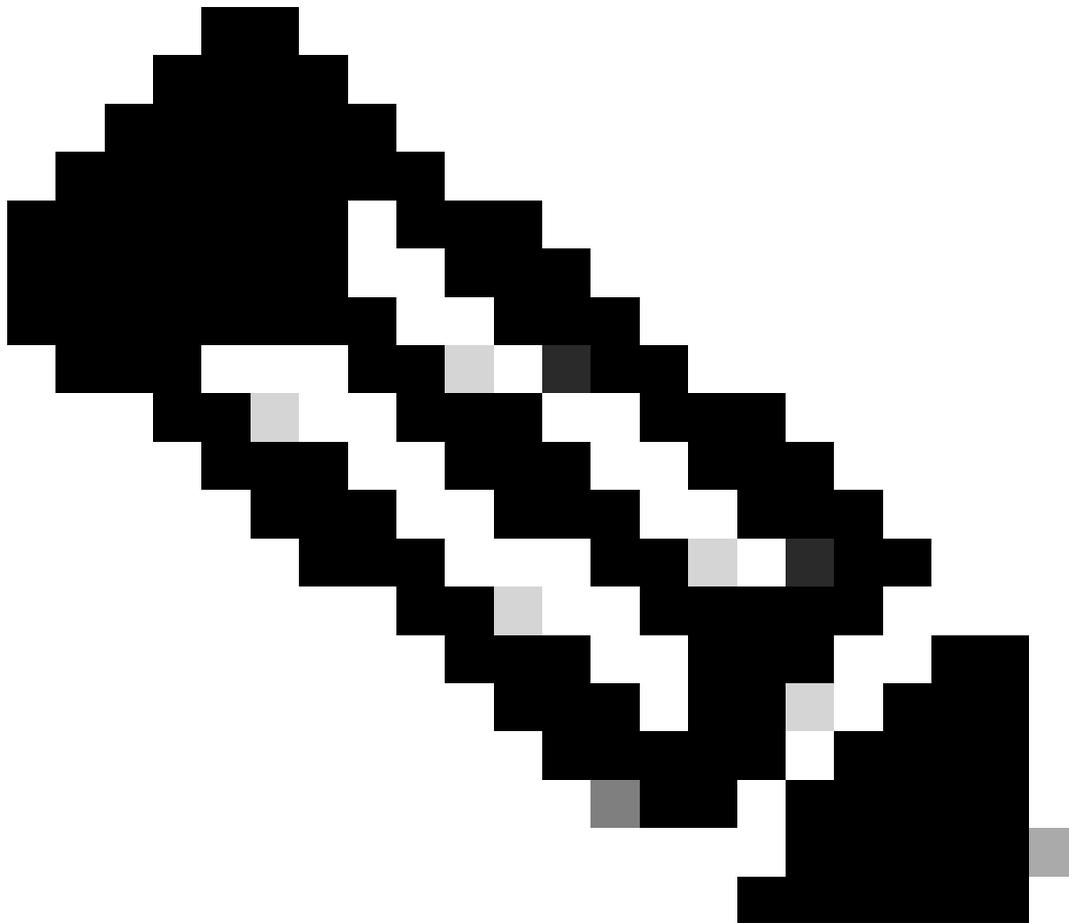
---

## Configurazione di base di gNMI in Cisco IOS XR

Di seguito viene riportata la configurazione base 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
```



Nota: È possibile configurare una porta in base all'impostazione. L'impostazione predefinita, senza utilizzare TLS, è 57400. Per ulteriori informazioni, fare clic su: [guida introduttiva a github - grpc](#)

---

## Yang come validator

Peyang è un validator di Yang scritto a Python. Questa biblioteca per Python aiuta a controllare i modelli YANG e anche, conoscendoli.

Per l'esecuzione di questa operazione come nella documentazione ([documentazione di Yang](#)) si consiglia di creare un ambiente virtuale nel computer.

Per l'esecuzione della [documentazione](#) di [venv](#) negli ambienti virtuali

È necessario eseguire:

```
python -m venv
```

Ad esempio (nel terminale MacOS):

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

Per installare Pyang in questo ambiente virtuale, inserire il CD nella directory e incollare il seguente:

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

Per questa dimostrazione, è stato usato python3 pip, una volta che il pip install -e è stato emesso, attivare il venv: source <directory ambiente virtuale>/bin/activate (per 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] [
```

...]

Validates the YANG module in

(or stdin), and all its dependencies.

Options:

-h, --help	Show this help message and exit
-v, --version	Show version number and exit

Con Pyang installato e funzionante, continuare con il download modelli.

Nel link successivo vengono elencati tutti i modelli eseguiti da Cisco IOS XR: [Modelli Cisco IOS XR](#).

Si consiglia di fare clonare questi modelli nella directory venv con il successivo collegamento al codice: <https://github.com/YangModels/yang.git>



Nota: Questa operazione non viene eseguita con l'ambiente virtuale attivato.

---

```
% 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.
```

Attivare di nuovo l'ambiente virtuale e verificare la query successiva: `pyang -f albero 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 search  
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: Si noti che i fogli hanno un formato di dati quale String, uint32 e così via. while roots non visualizza queste informazioni. Operazioni come GET e SET sono dedicate al pull/aggiornamento di questi valori.

---

Un'altra nota è che la maggior parte dei modelli richiede aumenti per avere la configurazione completa. Nell'output della CLI è presente la configurazione di gestione dell'interfaccia di base. Se è necessario visualizzare il protocollo IPv4, usare il comando seguente:

```
% pyang -f tree yan2/vendor/cisco/xr/711/Cisco-IOS-XR-ifmgr-cfg.yang yan2/vendor/cisco/xr/711/Cisco-IOS-XR-ifmgr-cfg.yang
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
```

```

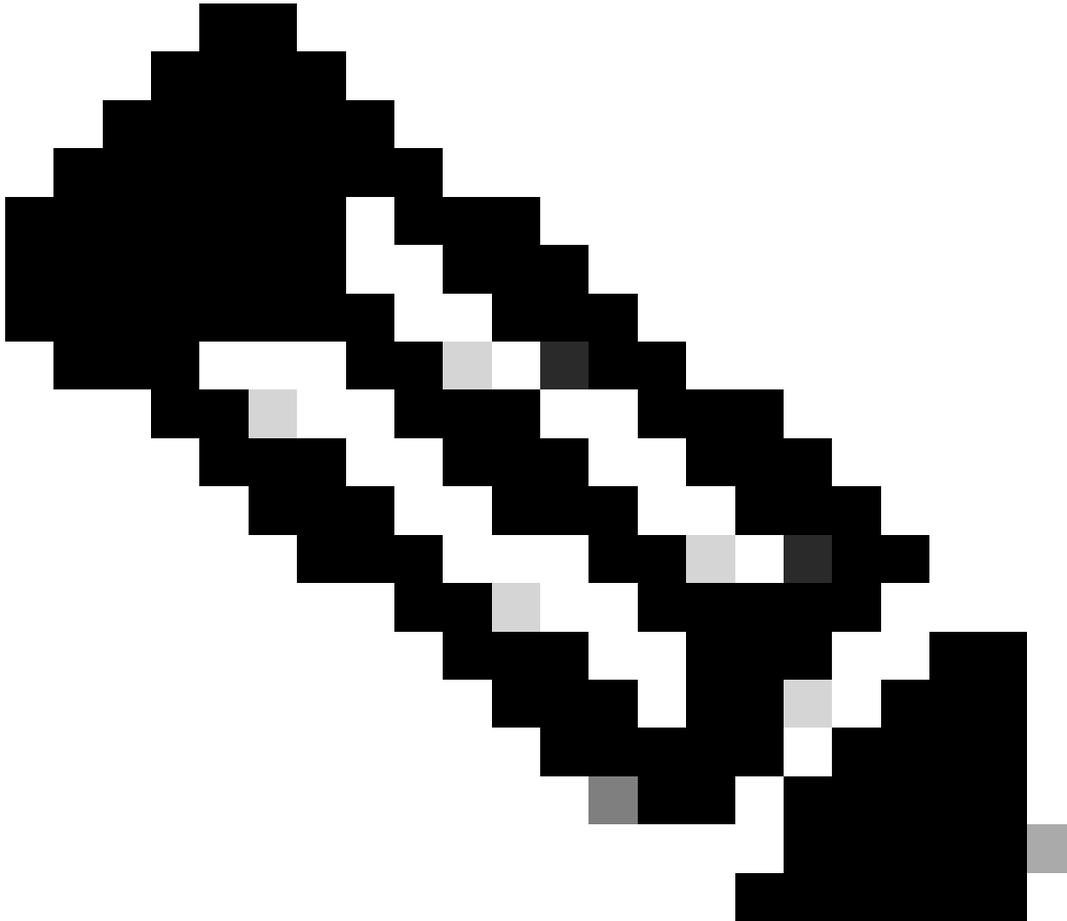
| +--rw suppress-threshold? uint32
| +--rw suppress-time?      uint32
| +--rw restart-penalty?    uint32
+--rw mtus
| +--rw mtu* [owner]
|   +--rw owner      xr:Cisco-ios-xr-string
|   +--rw mtu        uint32
+--rw encapsulation
| +--rw encapsulation?      string
| +--rw capsulation-options? uint32
+--rw 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 questa query vengono utilizzati due modelli: Cisco-IOS-XR-ifmgr-cfg.yang e Cisco-IOS-XR-ipv4-io-cfg.yang, quindi l'indirizzo IPv4 viene visualizzato come foglia.

---



Nota: nel caso in cui venga visualizzato un errore simile al seguente:  
"yang/vendor/cisco/xr/711/Cisco-IOS-XR-ifmgr-cfg.yang:5: errore: Il modulo "Cisco-IOS-XR-types" not found in search path" aggiunge `—path=` nel comando.

---

Dopo aver eseguito e selezionato questa operazione, qualsiasi utente può richiedere informazioni con le operazioni gNMI e modificare la data. Per ulteriori esempi, fare clic sul collegamento successivo: [Guida alla configurazione della programmabilità](#)

Se l'utente desidera eseguire una semplice API, sono disponibili strumenti quali: [grpcc](#)

Questa API è installata tramite NPM, è lo strumento utilizzato nel collegamento Guida alla configurazione della programmabilità, detto collegamento condivide altri esempi per gli utenti per testare query e risposte.

## Risoluzione dei problemi:

Per gNMI è necessario controllare la query prima di raccogliere qualsiasi voce, la maggior parte delle API come:

- gnmic
- grpc
- RPC

All, visualizza l'errore generato dal router.

Ad esempio:

```
"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

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

Questi sono errori dipendenti dalla piattaforma che devono essere controllati lungo il router. Si

consiglia di controllare che i comandi nella query possano essere emessi anche nel router tramite la CLI.

Per questo tipo di errori o per qualsiasi altro errore relativo alla piattaforma Cisco IOS XR, condividere le prossime informazioni su TAC:

- Query utilizzata e operazione:

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

- Errore visualizzato (uno qualsiasi degli errori mostrati in precedenza).
- Usare il comando successivo:

```
show grpc trace all
```

Verificare la query un paio di volte e ripetere il comando "show grpc trace all".

Gli errori sono di tipo Variant, ma mostrano anche il componente che può generare un problema:

Ad esempio:

- "sysdb" ha rilevato la condizione "warning" restituita da un verificatore o da una funzione di callback EDEDM: 'non trovato': Questo errore descrive sysdb, necessario per raccogliere i comandi show tech per questo processo nel router.

Nell'esempio successivo viene mostrato il processo show tech for sysdb con l'errore.

```
show tech-support sysdb
```

Per questo output, l'errore visualizza un componente e l'errore, raccogliere qualsiasi show tech-support che può essere correlato all'errore visualizzato.

- "Il framework YANG" ha rilevato la condizione "fatale" "Operazione non riuscita": Questo errore non visualizza un processo nel router, ossia la query non riesce nel modello. Condividere queste informazioni con TAC per verificare gli errori.

Dopo aver raccolto le informazioni, aggiungere anche il set di comandi successivo:

Nella VM XR:

```
show tech-support tctcsr
```

```
show tech-support grpcc
```

```
mostra gsp supporto tecnico
```

```
show tech-support
```

## Informazioni su questa traduzione

Cisco ha tradotto questo documento utilizzando una combinazione di tecnologie automatiche e umane per offrire ai nostri utenti in tutto il mondo contenuti di supporto nella propria lingua. Si noti che anche la migliore traduzione automatica non sarà mai accurata come quella fornita da un traduttore professionista. Cisco Systems, Inc. non si assume alcuna responsabilità per l'accuratezza di queste traduzioni e consiglia di consultare sempre il documento originale in inglese (disponibile al link fornito).