gNMI プロトコル

この機能では、gNMI CAPABILITIES、GET および SET RPC を使用したモデル駆動型の構成と運用データの取得について説明します。 gNMI バージョン 0.4.0 がサポートされています。

gNMI プロトコルの制約事項

- パスメッセージで Origin フィールドの使用はサポートされていません。空でない値を指定すると、エラーが返されます。
- RPC サービスを登録する
- JSON、BYTES、PROTO、および ASCI エンコーディングオプション

  JSON キーには、次の要素の名前空間が親とは異なる YANG プレフィックスが含まれている必要があります。たとえば、openconfig-vlan.yang の拡張から派生した routed-vlan は、親ノードの名前空間とは異なるため（親ノードはプレフィックス oc-if を持ちます）、oc-vlan:routed-vlan と入力する必要があります。
- GetRequest：
  - 運用データ型
  - 使用モデル
- GetResponse 通知
  - [エイリアス（Alias）]
  - Delete

- SetRequest では、ワイルドカードとすべてのキーはサポートされていません。完全に指定されたパスのみがサポートされます。

gNMI プロトコルの概要

gNMI について

gNMI は、Google によって開発された gRPC ネットワーク管理プロトコルです。 gNMI は、ネットワークデバイスの設定をインストール、操作、削除するメカニズムと、運用データを表示するメカニズムを提供します。gNMI を通じて提供されるコンテンツは YANG を使用してモデル化できます。

gRPC は、クラウドサーバと通信するモバイルクライアントを使用して低遅延で拡張可能な配布を実現するために Google によって開発されたリモート プロシージャ コールです。gRPC は gNMI を伝送し、データと動作要求を公式化して送信する手段を提供します。

サービスの障害が発生した場合、gNMI ブローカ（GNMIB）によって、up から down への動作状態の変化が示され、データベースが起動して実行されるまではすべての RPC がサービス利用不可のメッセージを返します。リカバリ時には、GNMIB によって down から up への動作状態の変化が示され、RPC の通常の処理が再開されます。

RFC 7951 の概要

RFC 7951 では、YANG データツリーとそのサブツリーの JavaScript オブジェクト表記（JSON）エンコーディングが規定されています。

YANG データ ノード（リーフ、コンテナ、リーフリスト、リスト、anydata ノード、および anyxml ノード）のインスタンスは、JSON オブジェクトまたは名前と値のペアのメンバーとしてエンコードされます。エンコーディング ルールは、設定データ、状態データ、RPC 操作のパラメータ、アクション、通知など、すべてのタイプのデータ ツリーで同じです。

データ ノード インスタンスはすべて名前と値のペアとしてエンコードされ、その名前はデータ ノード識別子から形成されます。値は、データ ノードのカテゴリによって異なります。

「リーフ」データ ノード

リーフ ノードは、データ ツリー内に値がありますが子はありません。リーフ インスタンスは、名前と値のペアとしてエンコードされます。値には、リーフのタイプに応じて、文字列、数値、リテラル「true」または「false」、または特殊な配列「[null]」を使用できます。指定されたパスのデータ項目がリーフノードである（子がなく、関連付けられた値を持たない）場合、そのリーフの値は直接エンコードされます。つまり、「ベア」値が指定されます（JSON オブジェクトは必須ではなく、ベアの JSON 値が含まれます)。

次に、リーフ ノード定義の例を示します。

```
leaf foo {
 type uint8;
}
```

次に、JSON でエンコードされた有効なインスタンスを示します。

```
"foo": 123
```

gNMI GET Request

gNMI GET RPC は、データツリーから、1 つ以上の設定属性、状態属性、派生状態属性、またはサポートされているモードに関連付けられたすべての属性を取得する方法を指定します。データ ツリーから値を取得するために、GetRequest がクライアントからターゲットに送信されます。GetRequest への応答として GetResponse が送信されます。

次の例は、JSON 構造の Get リクエストを示しています。

```
Creating a path object for xpath: /oc-if:interfaces/interface[name=Loopback111]
++++++++ Sending get request: ++++++++
path {
  elem {
    name: "oc-if:interfaces"
  }
  elem {
    name: "interface"
    key {
      key: "name"
      value: "Loopback111"
    }
  }
}
encoding: JSON_IETF
++++++++ Recevied get response: ++++++++
notification {
  timestamp: 1521699434792345469
  update {
    path {
      elem {
        name: "oc-if:interfaces"
      }
      elem {
        name: "interface"
        key {
          key: "name"
          value: "\"Loopback111\""
        }
      }
    }
    val {
      json_ietf_val: "{\n\t\"openconfig-interfaces:name\":\t\"Loopback111\",\n\t\
       "openconfig-interfaces:config\":\t{\n\t\t\
       "openconfig-interfaces:type\":\t\"ianaift:softwareLoopback\",\n\t\t\
       "openconfig-interfaces:name\":\t\"Loopback111\",\n\t\t\
       "openconfig-interfaces:enabled\":\t\"true\"\n\t},\n\t\
       "openconfig-interfaces:state\":\t{\n\t\t\
       "openconfig-interfaces:type\":\t\"ianaift:softwareLoopback\",\n\t\t\
       "openconfig-interfaces:name\":\t\"Loopback111\",\n\t\t\
       "openconfig-interfaces:enabled\":\t\"true\",\n\t\t\
       "openconfig-interfaces:ifindex\":\t52,\n\t\t\
       "openconfig-interfaces:admin-status\":\t\"UP\",\n\t\t\
       "openconfig-interfaces:oper-status\":\t\"UP\",\n\t\t\
       "openconfig-interfaces:last-change\":\t2018,\n\t\t\
```

```
        "openconfig-interfaces:counters\":\t{\n\t\t\t\
        "openconfig-interfaces:in-octets\":\t0,\n\t\t\t\
        "openconfig-interfaces:in-unicast-pkts\":\t0,\n\t\t\t\
        "openconfig-interfaces:in-broadcast-pkts\":\t0,\n\t\t\t\
        "openconfig-interfaces:in-multicast-pkts\":\t0,\n\t\t\t\
        "openconfig-interfaces:in-discards\":\t0,\n\t\t\t\
        "openconfig-interfaces:in-errors\":\t0,\n\t\t\t\
        "openconfig-interfaces:in-unknown-protos\":\t0,\n\t\t\t\
        "openconfig-interfaces:out-octets\":\t0,\n\t\t\t\
        "openconfig-interfaces:out-unicast-pkts\":\t0,\n\t\t\t\
        "openconfig-interfaces:out-broadcast-pkts\":\t0,\n\t\t\t\
        "openconfig-interfaces:out-multicast-pkts\":\t0,\n\t\t\t\
        "openconfig-interfaces:out-discards\":\t0,\n\t\t\t\
        "openconfig-interfaces:out-errors\":\t0,\n\t\t\t\
        "openconfig-interfaces:last-clear\":\t2018\n\t\t},\n\t\t\
        "openconfig-platform:hardware-port\":\t\"Loopback111\"\n\t},\n\t\
        "openconfig-interfaces:subinterfaces\":\t{\n\t\t\
        "openconfig-interfaces:index\":\t0,\n\t\t\
        "openconfig-interfaces:config\":\t{\n\t\t\t\
        "openconfig-interfaces:index\":\t0,\n\t\t\t\
        "openconfig-interfaces:name\":\t\"Loopback111\",\n\t\t\t\
        "openconfig-interfaces:enabled\":\t\"true\"\n\t\t},\n\t\t\
        "openconfig-interfaces:state\":\t{\n\t\t\t\
        "openconfig-interfaces:index\":\t0,\n\t\t\t\
        "openconfig-interfaces:name\":\t\"Loopback111.0\",\n\t\t\t\
        "openconfig-interfaces:enabled\":\t\"true\",\n\t\t\t\
        "openconfig-interfaces:admin-status\":\t\"UP\",\n\t\t\t\
        "openconfig-interfaces:oper-status\":\t\"UP\",\n\t\t\t\
        "openconfig-interfaces:last-change\":\t2018,\n\t\t\t\
        "openconfig-interfaces:counters\":\t{\n\t\t\t\t\
        "openconfig-interfaces:in-octets\":\t0,\n\t\t\t\t\
        "openconfig-interfaces:in-unicast-pkts\":\t0,\n\t\t\t\t\
        "openconfig-interfaces:in-broadcast-pkts\":\t0,\n\t\t\t\t\
        "openconfig-interfaces:in-multicast-pkts\":\t0,\n\t\t\t\t\
        "openconfig-interfaces:in-discards\":\t0,\n\t\t\t\t\
        "openconfig-interfaces:in-errors\":\t0,\n\t\t\t\t\
        "openconfig-interfaces:out-octets\":\t0,\n\t\t\t\t\
        "openconfig-interfaces:out-unicast-pkts\":\t0,\n\t\t\t\t\
        "openconfig-interfaces:out-broadcast-pkts\":\t0,\n\t\t\t\t\
        "openconfig-interfaces:out-multicast-pkts\":\t0,\n\t\t\t\t\
        "openconfig-interfaces:out-discards\":\t0,\n\t\t\t\t\
        "openconfig-interfaces:out-errors\":\t0,\n\t\t\t\t\
        "openconfig-interfaces:last-clear\":\t2018\n\t\t\t\t}\n\t\t},\n\t\t\
        "openconfig-if-ip:ipv6\":\t{\n\t\t\t\
        "openconfig-if-ip:config\":\t\"false\",\n\t\t\t\
        "openconfig-if-ip:state\":\t\"false\"\n\t\t}\n\t}\n}"
    }
  }
}
```

次の例は、JSON 構造のリーフでの GetRequest を示しています。

```
Creating a path object for xpath:
/oc-if:interfaces/interface[name=Loopback111]/state/oper-status
++++++++ Sending get request: ++++++++
path {
  elem {
    name: "oc-if:interfaces"
  }
  elem {
    name: "interface"
    key {
      key: "name"
```

```
      value: "Loopback111"
    }
  }
  elem {
    name: "state"
  }
  elem {
    name: "oper-status"
  }
}
encoding: JSON_IETF
++++++++ Recevied get response: ++++++++
notification {
  timestamp: 1521699326012374332
  update {
    path {
      elem {
        name: "oc-if:interfaces"
      }
      elem {
        name: "interface"
        key {
          key: "name"
          value: "\"Loopback111\""
        }
      }
      elem {
        name: "state"
      }
      elem {
        name: "oper-status"
      }
    }
    val {
      json_ietf_val: "\"UP\""
    }
  }
}
```

gNMI SetRequest

Set RPC は、サポートされているモデルに関連付けられた 1 つ以上の設定可能な属性を設定する方法を指定します。データツリー内の値を更新するために、SetRequest がクライアントからターゲットに送信されます。

SetRequest では、完全に指定された(ワイルドカード、およびすべてのキー指定のパスはサポートされていません)パス、および "json_ietf_val" または "json_val" TypedValue のみがサポートされます。JSON キーには YANG プレフィックスが含まれている必要があります。このプレフィックスでは次の要素の名前空間が親とは異なります。openconfig-vlan.yang の拡張から派生した "routed-vlan" は、親ノードの名前空間とは異なるため（親ノードのプレフィックスは oc-if）、"oc-vlan:routed-vlan" と入力する必要があります。

1 つの SetRequest に含まれる削除、置換、および更新は、全体で 1 つのトランザクション セットとして扱われます。トランザクションのいずれかの下位要素で障害が発生した場合は、トランザクション全体が拒否されてロールバックされます。SetRequest に対して SetResponse が返信されます。

次の例は、JSON 構造の SetRequest を示しています。

```
Creating UPDATE update for /oc-if:interfaces/interface[name=Loopback111]/config/
Creating a path object for xpath: /oc-if:interfaces/interface[name=Loopback111]/config/
++++++++ Sending set request: ++++++++
update {
  path {
    elem {
      name: "oc-if:interfaces"
    }
    elem {
      name: "interface"
      key {
        key: "name"
        value: "Loopback111"
      }
    }
    elem {
      name: "config"
    }
  }
  val {
    json_ietf_val: "{\"openconfig-interfaces:enabled\":\"false\"}"
  }
}
++++++++ Recevied set response: ++++++++
response {
  path {
    elem {
      name: "oc-if:interfaces"
    }
    elem {
      name: "interface"
      key {
        key: "name"
        value: "Loopback111"
      }
    }
    elem {
      name: "config"
    }
  }
  op: UPDATE
}
timestamp: 1521699342123890045
```

次の例は、JSON 構造のリーフの SetRequest を示しています。

```
Creating UPDATE update for /oc-if:interfaces/interface[name=Loopback111]/config/description
Creating a path object for xpath:
/oc-if:interfaces/interface[name=Loopback111]/config/description
++++++++ Sending set request: ++++++++
update {
  path {
    elem {
      name: "oc-if:interfaces"
    }
    elem {
      name: "interface"
      key {
        key: "name"
        value: "Loopback111"
```

```
      }
    }
    elem {
      name: "config"
    }
    elem {
      name: "description"
    }
  }
  val {
    json_ietf_val: "\"UPDATE DESCRIPTION\""
  }
}
++++++++ Recevied set response: ++++++++
response {
  path {
    elem {
      name: "oc-if:interfaces"
    }
    elem {
      name: "interface"
      key {
        key: "name"
        value: "Loopback111"
      }
    }
    elem {
      name: "config"
    }
    elem {
      name: "description"
    }
  }
  op: UPDATE
}
timestamp: 1521699342123890045
```

gNMI JSON_ietf_val

JSON タイプは、値が RFC 7159 で指定されている JSON 文字列としてエンコードされていることを示します。追加のタイプ（JSON_IETF など）は、JSON データのエンコードの特定の追加特性を示します（特に、YANG モデル化データのシリアル化に関連する場合）。

次に、 JSON_ietf_val メッセージの例を示します。

```
val {
  json_ietf_val:"{
   "oc-if:config": {
     "oc-if:description":
       "UPDATE DESCRIPTION"
   }
  }"
}
```

gNMI のエラー メッセージ

エラーが発生すると、gNMI は説明的なエラー メッセージを返します。次のセクションでは gNMI エラー メッセージをいくつか示します。

次に、パスが無効な場合に表示されるエラー メッセージの例を示します。

```
gNMI Error Response:
  <_Rendezvous of RPC that terminated with (StatusCode.TERMINATED,
   An error occurred while parsing provided xpath: unknown tag:
   "someinvalidxpath" Additional information: badly formatted or nonexistent path)>
```

次に、非実装エラーが発生した場合に表示されるエラー メッセージの例を示します。

```
gNMI Error Response:
  <_Rendezvous of RPC that terminated with (StatusCode.UNIMPLEMENTED,
    Requested encoding "ASCII" not supported)>
```

次に、データ要素が空の場合に表示されるエラー メッセージの例を示します。

```
gNMI Error Response:
  <_Rendezvous of RPC that terminated with (StatusCode.NOT_FOUND,
   Empty set returned for path "/oc-if:interfaces/noinfohere")>
```

gNMI プロトコルを有効にする方法

Linux での OpenSSL を使用した証明書の作成

証明書とトラストポイントは、セキュア gNMI サーバにのみ必要です。

次に、Linux マシン上で OpenSSL を使用して証明書を作成する例を示します。

```
# Setting up a CA
openssl genrsa -out rootCA.key 2048
openssl req -subj /C=/ST=/L=/O=/CN=rootCA -x509 -new -nodes -key rootCA.key -sha256 -out
 rootCA.pem

# Setting up device cert and key
openssl genrsa -out device.key 2048
openssl req -subj /C=/ST=/L=/O=/CN=<hostnameFQDN> -new -key device.key -out device.csr
openssl x509 -req -in device.csr -CA rootCA.pem -CAkey rootCA.key -CAcreateserial -out
device.crt -sha256
# Encrpyt device key - needed for input to IOS
openssl rsa -des3 -in device.key -out device.des3.key -passout pass:<password - remember
 this for later>

# Setting up client cert and key
openssl genrsa -out client.key 2048
openssl req -subj /C=/ST=/L=/O=/CN=gnmi_client -new -key client.key -out client.csr
openssl x509 -req -in client.csr -CA rootCA.pem -CAkey rootCA.key -CAcreateserial -out
client.crt -sha256
```

デバイスへの証明書のインストール

次の例は、デバイスに証明書をインストールする方法を示しています。

```
# Send:
Device# configure terminal
Device(config)# crypto pki import trustpoint1 pem terminal password password1

# Receive:
% Enter PEM-formatted CA certificate.
% End with a blank line or "quit" on a line by itself.

# Send:
# Contents of rootCA.pem, followed by newline + 'quit' + newline:
-----BEGIN CERTIFICATE-----
<snip>
-----END CERTIFICATE-----
quit

# Receive:
% Enter PEM-formatted encrypted private General Purpose key.
% End with "quit" on a line by itself.

# Send:
# Contents of device.des3.key, followed by newline + 'quit' + newline:
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,D954FF9E43F1BA20
<snip>
-----END RSA PRIVATE KEY-----
quit

# Receive:
% Enter PEM-formatted General Purpose certificate.
% End with a blank line or "quit" on a line by itself.

# Send:
# Contents of device.crt, followed by newline + 'quit' + newline:
-----BEGIN CERTIFICATE-----
<snip>
-----END CERTIFICATE-----
quit

# Receive:
% PEM files import succeeded.
Device(config)#

# Send:
Device(config)# crypto pki trustpoint trustpoint1
Device(ca-trustpoint)# revocation-check none
Device(ca-trustpoint)# end
Device#
```

非セキュア モードでの gNMI の有効化

（注） このタスクは、Cisco IOS XE Fuji 16.8.1 ー Amsterdam 17.2.x に適用されます。

[Day Zero setup] で、最初にデバイスを非セキュア モードで有効にしてから、デバイスを無効にし、セキュア モードを有効にします。非セキュア モードで gNMI を停止するには、**no gnmi-yang server** コマンドを使用します。

（注） gNMI 非セキュア サーバとセキュア サーバは同時に実行できます。

手順の概要

1. **enable**
2. **configure terminal**
3. **gnmi-yang**
4. **gnmi-yang server**
5. **gnmi-yang port** *port-number*
6. **end**
7. **show gnmi-yang state**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**enable** 例： `Device> enable`	特権 EXEC モードを有効にします。 • パスワードを入力します（要求された場合）。
ステップ 2	**configure terminal** 例： `Device# configure terminal`	グローバル コンフィギュレーション モードを開始します。
ステップ 3	**gnmi-yang** 例： `Device(config)# gnmi-yang`	gNMI プロセスを開始します。
ステップ 4	**gnmi-yang server** 例： `Device(config)# gnmi-yang server`	gNMI サーバを非セキュア モードで有効にします。
ステップ 5	**gnmi-yang port** *port-number* 例： `(Optional) Device(config)# gnmi-yang port 50000`	リッスンする gNMI ポートを設定します。 • デフォルトの非セキュア gNMI ポートは 9339 です。
ステップ 6	**end** 例： `Device(config)# end`	グローバル コンフィギュレーション モードを終了し、特権 EXEC モードに戻ります。

	コマンドまたはアクション	目的
ステップ 7	**show gnmi-yang state** 例： `Device# show gnmi-yang state`	gNMI サーバのステータスを表示します。

例

次に、**show gnmi-yang state** コマンドの出力例を示します。

```
Device# show gnmi-yang state

State Status
--------------------------------
Enabled Up
```

セキュア モードでの gNMI の有効化

（注） このタスクは、Cisco IOS XE Fuji 16.8.1 ー Amsterdam 17.2.x に適用されます。

セキュア モードで gNMI を停止するには、**no gnmi-yang secure-server** コマンドを使用します。

（注） gNMI 非セキュア サーバとセキュア サーバは同時に実行できます。

手順の概要

1. **enable**
2. **configure terminal**
3. **gnmi-yang**
4. **gnmi-yang secure-server**
5. **gnmi-yang secure-trustpoint** *trustpoint-name*
6. **gnmi-yang secure-client-auth**
7. **gnmi-yang secure-port**
8. **end**
9. **show gnmi-yang state**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**enable** 例： `Device> enable`	特権 EXEC モードを有効にします。 • パスワードを入力します（要求された場合）。
ステップ 2	**configure terminal** 例： `Device# configure terminal`	グローバル コンフィギュレーション モードを開始します。
ステップ 3	**gnmi-yang** 例： `Device(config)# gnmi-yang`	gNMI プロセスを開始します。
ステップ 4	**gnmi-yang secure-server** 例： `Device(config)# gnmi-yang secure-server`	gNMI サーバをセキュア モードで有効にします。
ステップ 5	**gnmi-yang secure-trustpoint** *trustpoint-name* 例： `Device(config)# gnmi-yang secure-trustpoint trustpoint1`	gNMI が認証に使用するトラストポイントと証明書セットを指定します。
ステップ 6	**gnmi-yang secure-client-auth** 例： `Device(config)# gnmi-yang secure-client-auth`	（任意）gNMI プロセスは、ルート証明書と照合してクライアント証明書を認証します。
ステップ 7	**gnmi-yang secure-port** 例： `Device(config)# gnmi-yang secure-port`	（任意）リッスンする gNMI ポートを設定します。 • デフォルトの非セキュア gNMI ポートは 9339 です。
ステップ 8	**end** 例： `Device(config)# end`	グローバル コンフィギュレーション モードを終了し、特権 EXEC モードに戻ります。
ステップ 9	**show gnmi-yang state** 例： `Device# show gnmi-yang state`	gNMI サーバのステータスを表示します。

例

次に、**show gnmi-yang state** コマンドの出力例を示します。

```
Device# show gnmi-yang state

State Status
--------------------------------
Enabled Up
```

gNMI クライアントの接続

以前に設定したクライアント証明書とルート証明書を使用して gNMI クライアントが接続されます。

次に、Python を使用して gNMI クライアントを接続する例を示します。

```
# gRPC Must be compiled in local dir under path below:
>>> import sys
>>> sys.path.insert(0, "reference/rpc/gnmi/")
>>> import grpc
>>> import gnmi_pb2
>>> import gnmi_pb2_grpc
>>> gnmi_dir = '/path/to/where/openssl/creds/were/generated/'

# Certs must be read in as bytes
>>> with open(gnmi_dir + 'rootCA.pem', 'rb') as f:
>>>     ca_cert = f.read()
>>> with open(gnmi_dir + 'client.crt', 'rb') as f:
>>>     client_cert = f.read()
>>> with open(gnmi_dir + 'client.key', 'rb') as f:
>>>     client_key = f.read()

# Create credentials object
>>> credentials = grpc.ssl_channel_credentials(root_certificates=ca_cert,
private_key=client_key, certificate_chain=client_cert)

# Create a secure channel:
# Default port is 50052, can be changed on ios device with 'gnmi-yang secure-port ####'
>>> port = 50052
>>> host = <HOSTNAME FQDN>
>>> secure_channel = grpc.secure_channel("%s:%d" % (host, port), credentials)

# Create secure stub:
>>> secure_stub = gnmi_pb2_grpc.gNMIStub(secure_channel)

# Done! Let's test to make sure it works:
>>> secure_stub.Capabilities(gnmi_pb2.CapabilityRequest())
supported_models {
<snip>
}
supported_encodings: <snip>
gNMI_version: "0.4.0"
```

gNMI プロトコルを有効にするための設定例

例：gNMI プロトコルの有効化

（注） この例は、Cisco IOS XE Fuji 16.8.1 ～ Amsterdam 17.2.x に適用されます。

例：非セキュア モードでの gNMI の有効化

次に、gNMI サーバを非セキュア モードで有効にする例を示します。

```
Device# configure terminal
Device(config)# gnmi-yang
Device(config)# gnmi-yang server
Device(config)# gnmi-yang port 50000 <The default port is 9339.>
Device(config)# end
Device#
```

例：セキュア モードでの gNMI の有効化

次に、gNMI サーバをセキュア モードで有効にする例を示します。

```
Device# configure terminal
Device(config)# gnmi-yang server
Device(config)# gnmi-yang secure-server
Device(config)# gnmi-yang secure-trustpoint trustpoint1
Device(config)# gnmi-yang secure-client-auth
Device(config)# gnmi-yang secure-port 50001 <The default port is 9339.>
Device(config)# end
Device#
```

gNMI プロトコルの関連資料

関連資料

関連項目	マニュアル タイトル
DevNet	https://developer.cisco.com/site/ios-xe/
gNMI	https://github.com/openconfig/reference/blob/master/rpc/gnmi/gnmi-specification.md
gNMI パス エンコーディング	https://github.com/openconfig/reference/blob/master/rpc/gnmi/gnmi-path-conventions.md

標準および RFC

標準/RFC	タイトル
RFC 7951	YANG でモデル化されたデータの JSON エンコーディング

シスコのテクニカル サポート

説明	リンク
シスコのサポート Web サイトでは、シスコの製品やテクノロジーに関するトラブルシューティングにお役立ていただけるように、マニュアルやツールをはじめとする豊富なオンライン リソースを提供しています。 お使いの製品のセキュリティ情報や技術情報を入手するために、Cisco Notification Service（Field Notice からアクセス）、Cisco Technical Services Newsletter、Really Simple Syndication（RSS）フィードなどの各種サービスに加入できます。 シスコのサポート Web サイトのツールにアクセスする際は、Cisco.com のユーザ ID およびパスワードが必要です。	http://www.cisco.com/support

gNMI プロトコルの機能情報

次の表に、このモジュールで説明した機能に関するリリース情報を示します。この表は、ソフトウェア リリース トレインで各機能のサポートが導入されたときのソフトウェア リリースだけを示しています。その機能は、特に断りがない限り、それ以降の一連のソフトウェア リリースでもサポートされます。

プラットフォームのサポートおよびシスコ ソフトウェア イメージのサポートに関する情報を検索するには、Cisco Feature Navigator を使用します。Cisco Feature Navigator にアクセスするには、www.cisco.com/go/cfn に移動します。Cisco.com のアカウントは必要ありません。

表 *1 : gNMI* プロトコルの機能情報

機能名	リリース	機能情報
gNMI プロトコル	Cisco IOS XE Fuji 16.8.1a	この機能では、gNMI の機能と GET および SET RPC を使用したモデル駆動型の設定と運用データの取得について説明します。 この機能は、次のプラットフォームに実装されていました。 • Cisco Catalyst 9300 シリーズ スイッチ • Cisco Catalyst 9400 シリーズ スイッチ • Cisco Catalyst 9500 シリーズ スイッチ
	Cisco IOS XE Gibraltar 16.10.1	gNMI 名前空間と gNMI ワイルドカードのサポートは、次のプラットフォームに追加されました。 • Cisco Catalyst 9300 シリーズ スイッチ • Cisco Catalyst 9400 シリーズ スイッチ • Cisco Catalyst 9500 シリーズ スイッチ

翻訳について
このドキュメントは、米国シスコ発行ドキュメントの参考和訳です。リンク情報につきましては、日本語版掲載時点で、英語版にアップデートがあり、リンク先のページが移動/変更されている場合がありますことをご了承ください。あくまでも参考和訳となりますので、正式な内容については米国サイトのドキュメントを参照ください。