RESTCONF プロトコル

この章では、HTTP ベースの Representational State Transfer コンフィギュレーション プロトコル（RESTCONF）を設定する方法を説明します。RESTCONF は、設定データ、状態データ、データ モデルに固有のリモート プロシージャ コール（RPC）操作、および YANG モデルで定義されているイベントにアクセスするための、標準的なメカニズムに基づく、プログラミングが可能なインターフェイスを提供します。

RESTCONF プロトコルの前提条件

- RESTCONF に対して Cisco IOS-HTTP サービスを有効にします。詳細については、『RESTCONF RPC の例』を参照してください。

RESTCONF プロトコルの制約事項

RESTCONF プロトコルには、次の制約事項が適用されます。

- 通知およびイベント ストリーム
- YANG パッチ
- フィルタ、開始時、停止時、再生、アクションなどのオプションのクエリ パラメータ
- RESTCONF 機能は、デュアル IOSd 設定またはソフトウェア冗長性を実行しているデバイスではサポートされていません。

RESTCONF プロトコルに関する情報

RESTCONF の概要

このセクションでは、構成をネットワーク デバイスにプログラムを使用して書き込めるようにする、プロトコルおよびモデリング言語について説明します。

- RESTCONF：構造化データ（XML または JSON）および YANG を使用して REST ライクな API を提供します。これによりさまざまなネットワーク デバイスにプログラムを使用してアクセスできます。RESTCONF API は HTTPs メソッドを使用します。
- YANG：モデル構成および操作機能に使用されるデータ モデリング言語。YANG は、NETCONF および RESTCONF API によって実行できる関数の有効範囲と種類を決定します。

Cisco IOS XE Fuji 16.8.1 よりも前のリリースでは、運用データ マネージャ（ポーリングに基づく）が個別に有効になっていました。Cisco IOS XE Fuji 16.8.1 以降のリリースでは、運用データは、NETCONF を実行しているプラットフォームで動作し（設定データの仕組みと同様）、デフォルトで有効になっています。運用データのクエリまたはストリーミングに対応するコンポーネントの詳細については、GitHub リポジトリで命名規則の **-oper* を参照してください。

HTTPs メソッド

ステートレス プロトコルである HTTPS ベースの RESTCONF プロトコル（RFC 8040）は、セキュアなHTTPメソッドを使用して、YANG定義データが含まれる概念データストア（NETCONF データストアを実装するサーバと互換性がある）で CREATE、READ、UPDATE、および DELETE（CRUD）操作を提供します。

次の表では、RESTCONF操作にNETCONFプロトコル操作を関連付ける方法を示しています。

オプション	サポートされているメソッド
GET	読み取り
PATCH	更新
PUT	作成または置換
POST	作成または操作（リロード、デフォルト）
DELETE	ターゲット リソースの削除
HEAD	ヘッダー メタデータ（応答本文なし）

RESTCONF ルート リソース

- RESTCONF デバイスは、RESTCONF 属性を含むリンク要素である /.well-known/host-meta リソースにより、RESTCONF API のルートを決定します。
- RESTCONF デバイスは、要求 URI のパスの最初の部分として RESTCONF API ルート リソースを使用します。

例：

```
Example returning /restconf:

   The client might send the following:

      GET /.well-known/host-meta HTTP/1.1
      Host: example.com
      Accept: application/xrd+xml

   The server might respond as follows:

      HTTP/1.1 200 OK
      Content-Type: application/xrd+xml
      Content-Length: nnn

      <XRD xmlns='http://docs.oasis-open.org/ns/xri/xrd-1.0'>
          <Link rel='restconf' href='/restconf'/>
      </XRD>
```

URI の例：

- GigabitEthernet0/0/2：
  https://10.104.50.97/restconf/data/Cisco-IOS-XE-native:native/interface/GigabitEthernet=0%2F0%2F2
- fields=name：
  https://10.104.50.97/restconf/data/Cisco-IOS-XE-native:native/interface/GigabitEthernet=0%2F0%2F2?fields=name
- depth=1：
  https://10.85.116.59/restconf/data/Cisco-IOS-XE-native:native/interface/GigabitEthernet?depth=1
- Name と IP：
  https://10.85.116.59/restconf/data/Cisco-IOS-XE-native:native/interface?fields=GigabitEthernet/ip/address/primary;name
- MTU（フィールド）：
  https://10.104.50.97/restconf/data/Cisco-IOS-XE-native:native/interface?fields=GigabitEthernet(mtu)
- MTU：
  https://10.85.116.59/restconf/data/Cisco-IOS-XE-native:native/interface/GigabitEthernet=3/mtu
- ポートチャネル：
  https://10.85.116.59/restconf/data/Cisco-IOS-XE-native:native/interface/Port-channel
- 「Char」から「Hex」への変換チャート：http://www.columbia.edu/kermit/ascii.html

バージョン情報の表示

Cisco-IOS-XE-install-operモジュールには、バージョン情報を表示するさまざまなノードがあります。

次のサンプル RPC は、Cisco-IOS-XE-install-oper モジュールのサポートされているノードの一部と、メジャーおよびマイナーリリースバージョンを含むホストからの応答を示しています。

```
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
        message-id="urn:uuid:7d0908d8-0d5f-4521-9d7b-380b81304776">
  <nc:get>
    <nc:filter>
      <install-oper-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-install-oper">
        <install-location-information>
          <install-version-info>
            <version/>
            <version-extension/>
            <current/>
            <src-filename/>
          </install-version-info>
        </install-location-information>
      </install-oper-data>
    </nc:filter>
  </nc:get>
</nc:rpc>

##
Received message from host

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
    xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
    message-id="urn:uuid:7d0908d8-0d5f-4521-9d7b-380b81304776">
  <data>
    <install-oper-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-install-oper">
      <install-location-information>
        <install-version-info>
          <version>17.06.04.0.3870</version>
          <version-extension>1651661105</version-extension>
          <current>install-version-state-present</current>
          <src-filename/>
        </install-version-info>
        <install-version-info>
          <version>17.09.01.0.158212</version>
          <version-extension>1651125381</version-extension>
          <current>install-version-state-present</current>
          <src-filename/>
        </install-version-info>
        <install-version-info>
          <version>17.10.01.0.158658</version>
          <version-extension>1651754624</version-extension>
          <current>install-version-state-present</current>

<src-filename>/bootflash/c8000v-universalk9nic.2022-05-05_18.13.SSA.bin</src-filename>
        </install-version-info>
        <install-version-info>
          <version>17.10.01.0.160585</version>
          <version-extension>1656581638</version-extension>
          <current>install-version-state-provisioned-committed</current>

<src-filename>/bootflash/c8000v-universalk9.2022-06-30_15.03.SSA.bin</src-filename>
        </install-version-info>
```

```
        <install-version-info>
          <version>17.10.01.0.162616</version>
          <version-extension>1657120419</version-extension>
          <current>install-version-state-present</current>
          <src-filename>/bootflash/c8000v-universalk9.BLD_POLARIS_DEV_LATEST_20220706_
            143733.SSA.bin</src-filename>
        </install-version-info>
      </install-location-information>
    </install-oper-data>
  </data>
</rpc-reply>
```

プロトコル、gNMI、NETCONF、またはRESTCONFを使用する場合、Cisco-IOS-XE-native:version モジュールは、メジャーリリースバージョンのみを表示します。

RESTCONF API リソース

APIリソースは、+restconfに位置する上位リソースです。これは次のメディアタイプをサポートします。

（注） メディアは、RESTCONF サーバ（XML または JSON）に送信される YANG 形式 RPC のタイプです。

- application/yang-data+xml または application/yang-data+json
- API リソースには、RESTCONF DATASTORE および OPERATION リソースの RESTCONF ルート リソースが含まれます。次に例を示します。

```
The client may then retrieve the top-level API resource, using the
   root resource "/restconf".

      GET /restconf HTTP/1.1
      Host: example.com
      Accept: application/yang-data+json

   The server might respond as follows:

      HTTP/1.1 200 OK
      Date: Thu, 26 Jan 2017 20:56:30 GMT
      Server: example-server
      Content-Type: application/yang-data+json

      {
        "ietf-restconf:restconf" : {
          "data" : {},
          "operations" : {},
          "yang-library-version" : "2016-06-21"
        }
      }
```

詳細については、RFC 3986 を参照してください

メソッド

メソッドは、ターゲット リソースで実行される HTTPS 操作（GET/PATCH/POST/DELETE/OPTIONS/PUT）です。YANG 形式 RPC は、RESTCONF サーバに存在するターゲット YANG モデルに関連する指定のリソースに対して、特定のメソッドを呼び出します。Uniform Resource Identifier（URI）は指定されたリソースのロケーション ID として機能するため、クライアントの RESTCONF メソッドは、その特定のリソースを探して、HTTPS のメソッドまたはプロパティで指定されたアクションを実行することができます。

詳細については、「RFC 8040：RESTCONF プロトコル」を参照してください。

RESTCONF YANG パッチのサポート

RESTCONF は、RFC 8072 で指定されている YANG パッチメディアタイプをサポートしています。YANG パッチは、RESTCONF サーバによってターゲットデータストアに適用される編集の順序付きリストです。YANG パッチ操作は、メディアタイプ *application/yang-patch+xml* または *application/yang-patch+json* のいずれかを使用した表現でパッチメソッド要求を送信することによって RESTCONF クライアントにより呼び出されます。

YANG パッチは一意のパッチ ID で識別されます。パッチは編集の順序付けられたコレクションであり、各編集は編集 ID によって識別されます。ターゲットリソースに適用される編集操作（「作成」、「削除（delete）」、「挿入」、「マージ」、「移動」、「置換」、「削除（remove）」）があります。

RESTCONF YANG パッチがサポートされているかどうかを確認するには、次の RESTCONF Get 要求を発行します。

```
$ curl -k -s -u admin:DMIdmi1! --location-trusted
"https://10.1.1.1/restconf/data/ietf-restconf-monitoring:restconf-state/capabilities"
-X GET

<capabilities xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf-monitoring"
xmlns:rcmon="urn:ietf:params:xml:ns:yang:ietf-restconf-monitoring">

<capability>urn:ietf:params:restconf:capability:defaults:1.0?basic-mode=explicit</capability>

  <capability>urn:ietf:params:restconf:capability:depth:1.0</capability>
  <capability>urn:ietf:params:restconf:capability:fields:1.0</capability>
  <capability>urn:ietf:params:restconf:capability:with-defaults:1.0</capability>
  <capability>urn:ietf:params:restconf:capability:filter:1.0</capability>
  <capability>urn:ietf:params:restconf:capability:replay:1.0</capability>

<capability>urn:ietf:params:restconf:capability:yang-patch:1.0</capability>

  <capability>http://tail-f.com/ns/restconf/collection/1.0</capability>
  <capability>http://tail-f.com/ns/restconf/query-api/1.0</capability>
</capabilities>
```

このセクションでは、いくつかの RESTCONF YANG パッチの例を示します。

リソースの追加エラー

ファイルを編集しようとしているときに、最初の編集がすでに存在し、エラーが報告されます。最初の編集が失敗したため、残りの編集は試行されません。この例では、XML エンコーディングが使用されています。

次の例は、RESTCONF クライアントからのリソース追加要求を示しています。

```
<yang-patch xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-patch">
        <patch-id>add-hostname-patch</patch-id>
        <edit>
          <edit-id>edit1</edit-id>
          <operation>create</operation>
          <target>/hostname</target>
          <value>
              <hostname
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">Cat9K-test</hostname>
          </value>
        </edit>
        <edit>
          <edit-id>edit2</edit-id>
          <operation>create</operation>
          <target>/interface/Loopback=1</target>
          <value>
              <interface xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
              <Loopback>
                <name>1</name>
              </Loopback>
              </interface>
          </value>
        </edit>
</yang-patch>
```

次の例は、RESTCONF サーバからの JSON 応答を示しています。

```
Device:/nobackup/folder1/confd_6313/bin $ curl -k -s -u admin:DMIdmi1! --location-trusted

"https://10.1.1.1/restconf/data/Cisco-IOS-XE-native:native" -X PATCH -H "Accept:
application/yang-data+json" -d
'@yang_patch_create_hostname' -H "Content-type: application/yang-patch+xml"
{
  "ietf-yang-patch:yang-patch-status": {
    "patch-id": "add-hostname-patch",
    "edit-status": {
      "edit": [
        {
          "edit-id": "edit1",
          "errors": {
            "error": [
              {
                "error-type": "application",
                "error-tag": "data-exists",
                "error-path": "/Cisco-IOS-XE-native:native/hostname",
                "error-message": "object already exists: /ios:native/ios:hostname"
              }
            ]
          }
        }
      ]
    }
  }
}
```

次の例は、RESTCONF サーバからの XML 応答を示しています。

```
Device:/nobackup/folder1/confd_6313/bin $ curl -k -s -u admin:DMIdmi1! --location-trusted

"https://10.1.1.1/restconf/data/Cisco-IOS-XE-native:native" -X PATCH -H "Accept:
application/yang-data+xml" -d
'@yang_patch_create_hostname' -H "Content-type: application/yang-patch+xml"

 <yang-patch-status xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-patch">
   <patch-id>add-hostname-patch</patch-id>
   <edit-status>
     <edit>
       <edit-id>edit1</edit-id>
       <errors>
         <error>
           <error-type>application</error-type>
           <error-tag>data-exists</error-tag>
           <error-path
xmlns:ios="http://cisco.com/ns/yang/Cisco-IOS-XE-native">/ios:native/ios:hostname</error-path>

          <error-message>object already exists: /ios:native/ios:hostname</error-message>

         </error>
       </errors>
     </edit>
   </edit-status>
 </yang-patch-status>device:/nobackup/folder1/confd_6313/bin $
```

リソースの追加成功

次の例は、編集要求を示しています。

```
<yang-patch xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-patch">
        <patch-id>add-Loopback-patch</patch-id>
        <edit>
          <edit-id>edit1</edit-id>
          <operation>create</operation>
          <target>/Loopback=1</target>
          <value>
              <Loopback xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
                <name>1</name>
              </Loopback
          </value>
        </edit>
</yang-patch>
```

次の例は、編集要求が成功したことを示しています。

```
Device:/nobackup/folder1/confd_6313/bin $ curl -k -s -u admin:DMIdmi1! --location-trusted

"https://10.1.1.1/restconf/data/Cisco-IOS-XE-native:native/interface" -X PATCH -H "Accept:
 application/yang-data+json"
-d '@yang_patch_create_Loopback_interface' -H "Content-type: application/yang-patch+xml"
Device:/nobackup/folder1/confd_6313/bin
{
  "ietf-yang-patch:yang-patch-status": {
    "patch-id": "add-Loopback-patch",
    "ok" : [null]
  }
}
```

リストエントリの挿入

次に、ループバック 1 がループバック 0 の後に挿入される例を示します。

```
<yang-patch xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-patch">
        <patch-id>insert-Loopback-patch</patch-id>
        <edit>
          <edit-id>edit1</edit-id>
          <operation>insert</operation>
          <target>/Loopback=1</target>
          <point>/Loopback=0</point>
          <where>after</where>
          <value>
              <Loopback xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
                <name>1</name>
              </Loopback>
          </value>
        </edit>
</yang-patch>
```

次の例は、リストの挿入要求が成功したことを示しています。

```
Device:/nobackup/folder1/confd_6313/bin $ curl -k -s -u admin:DMIdmi1! --location-trusted

"https://10.1.1.1/restconf/data/Cisco-IOS-XE-native:native/interface" -X PATCH -H "Accept:
 application/yang-data+json" -d
'@yang_patch_create_Loopback_interface' -H "Content-type: application/yang-patch+xml"
Device:/nobackup/folder1/confd_6313/bin
{
  "ietf-yang-patch:yang-patch-status": {
    "patch-id": "insert-Loopback-patch",
    "ok" : [null]
  }
}
```

リストエントリの移動

次に、ループバック 1 がループバック 0 の前に移動される例を示します。

```
<yang-patch xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-patch">
        <patch-id>move-Loopback-patch</patch-id>
        <edit>
          <edit-id>edit1</edit-id>
          <operation>move</operation>
          <target>/Loopback=1</target>
          <point>/Loopback=0</point>
          <where>before</where>
          <value>
              <Loopback xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
                <name>1</name>
              </Loopback>
          </value>
        </edit>
</yang-patch>
```

次の例は、移動要求が成功したことを示しています。

```
Device:/nobackup/folder1/confd_6313/bin $ curl -k -s -u admin:DMIdmi1! --location-trusted

"https://10.1.1.1/restconf/data/Cisco-IOS-XE-native:native/interface" -X PATCH -H "Accept:
```

```
 application/yang-data+json" -d
'@yang_patch_create_Loopback_interface' -H "Content-type: application/yang-patch+xml"
Device:/nobackup/folder1/confd_6313/bin
{
  "ietf-yang-patch:yang-patch-status": {
    "patch-id": "move-Loopback-patch",
    "ok" : [null]
  }
}
```

RESTCONF プロトコルの設定方法

AAA を使用した NETCONF/RESTCONF の認証

始める前に

NETCONF 接続と RESTCONF 接続は、認証、許可、およびアカウンティング（AAA）を使用して認証する必要があります。その結果、権限レベル 15 のアクセスで定義された RADIUS または TACACS + ユーザに、システムへのアクセスが許可されます。

手順の概要

1. **enable**
2. **configure terminal**
3. **aaa new-model**
4. **aaa group server radius** *server-name*
5. **server-private** *ip-address* **key** *key-name*
6. **ip vrf forwarding** *vrf-name*
7. **exit**
8. **aaa authentication login default group** *group-name***local**
9. **aaa authentication login** *list-name* **none**
10. **aaa authorization exec default group** *group-name***local**
11. **aaa session-id common**
12. **line console** *number*
13. **login authentication** *authentication-list*
14. **end**

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 モードをイネーブルにします • パスワードを入力します（要求された場合）。

	コマンドまたはアクション	目的
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	**aaa new-model** 例： `Device(config)# aaa new-model`	AAA をイネーブルにします。
ステップ 4	**aaa group server radius** *server-name* 例： `Device(config)# aaa group server radius ISE`	RADIUS サーバを追加し、サーバ グループ RADIUS コンフィギュレーション モードを開始します。 • server-name 引数には、RADIUS サーバ グループ名を指定します。
ステップ 5	**server-private** *ip-address* **key** *key-name* 例： `Device(config-sg-radius)# server-private 172.25.73.76 key Cisco123`	プライベート RADIUS サーバの IP アドレスと暗号キーを設定します。
ステップ 6	**ip vrf forwarding** *vrf-name* 例： `Device(config-sg-radius)# ip vrf forwarding Mgmt-intf`	AAA RADIUS または TACACS+ サーバ グループの Virtual Route Forwarding（VRF）参照情報を設定します。
ステップ 7	**exit** 例： `Device(config-sg-radius)# exit`	サーバ グループ RADIUS コンフィギュレーション モードを終了し、グローバル コンフィギュレーション モードに戻ります。
ステップ 8	**aaa authentication login default group** *group-name***local** 例： `Device(config)# aaa authentication login default group ISE local`	ログイン時に、指定されたグループ名をデフォルトのローカル AAA 認証として設定します。
ステップ 9	**aaa authentication login** *list-name* **none** 例： `Device(config)# aaa authentication login NOAUTH none`	システムへのログイン中に認証が不要であることを指定します。
ステップ 10	**aaa authorization exec default group** *group-name***local** 例： `Device(config)# aaa authorization exec default group ISE local`	許可を実行して、EXEC シェルの実行がユーザに許可されているかどうかを確認します。
ステップ 11	**aaa session-id common** 例：	指定のコールに対して送信されたセッション ID 情報が同じになるようにします。

	コマンドまたはアクション	目的
	`Device(config)# aaa session-id common`	
ステップ 12	**line console** *number* 例： `Device(config)# line console 0`	設定する特定の回線を識別し、ライン コンフィギュレーション モードを開始します。
ステップ 13	**login authentication** *authentication-list* 例： `Device(config-line)# login authentication NOAUTH`	ログインに対する AAA 認証をイネーブルにします。
ステップ 14	**end** 例： `Device(config-line)# end`	回線コンフィギュレーションモードを終了します。続いて、特権 EXEC モードに戻ります。

RESTCONF の Cisco IOS HTTP サービスの有効化

RESTCONF インターフェイスを使用するには、次の作業を行います。

手順の概要

1. **enable**
2. **configure terminal**
3. **restconf**
4. **ip http secure-server**
5. **end**

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	**restconf** 例： `Device(config)# restconf`	ネットワーク デバイスで RESTCONF インターフェイスを有効にします。
ステップ 4	**ip http secure-server** 例：	セキュア HTTP（HTTPS）サーバをイネーブルにします。

	コマンドまたはアクション	目的
	`Device(config)# ip http secure-server`	
ステップ 5	**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 モードを開始します。

RESTCONF の設定の検証

スタートアップ コンフィギュレーションを使用してデバイスが起動すると、*nginx* プロセスが実行中になります。ただし、DMI プロセスは有効にはなりません。

次の **show platform software yang-management process monitor** コマンドの出力例は、nginx プロセスが実行中であることを示しています。

```
Device# show platform software yang-management process monitor

COMMAND           PID S    VSZ   RSS %CPU %MEM     ELAPSED
nginx           27026 S 332356 18428  0.0  0.4       01:34
nginx           27032 S 337852 13600  0.0  0.3       01:34
```

NGINX は、プロキシ Web サーバとして機能する内部 Web サーバで、Transport Layer Security（TLS）ベースの HTTPS を提供します。HTTPS を介して送信された RESTCONF 要求は、最初に NGINX プロキシ Web サービスによって受信され、さらに要求が構文/セマンティックチェックのために confd Web サーバに転送されます。

次の **show platform software yang-management process** コマンドの出力例は、スタートアップ コンフィギュレーションを使用してデバイスが起動されたときのすべてのプロセスのステータスを示しています。

```
Device# show platform software yang-management process

confd            : Not Running
nesd             : Not Running
syncfd           : Not Running
ncsshd           : Not Running
dmiauthd         : Not Running
nginx            : Running
ndbmand          : Not Running
pubd             : Not Running
```

restconf コマンドが設定されている場合、nginx プロセスが再起動され、DMI プロセスが起動されます。

次の **show platform software yang-management process** コマンドの出力例は、nginx プロセス と DMI プロセスが起動して実行中であることを示しています。

```
Device# show platform software yang-management process

confd            : Running
nesd             : Running
syncfd           : Running
```

```
ncsshd           : Not Running ! NETCONF-YANG is not configured, hence ncsshd process
is in not running.
dmiauthd         : Running
vtyserverutild   : Running
opdatamgrd       : Running
nginx            : Running ! nginx process is up due to the HTTP configuration, and it
is restarted when RESTCONF is enabled.
ndbmand          : Running
```

次の show platform software yang-management process monitor コマンドの出力例では、すべてのプロセスに関する詳細情報が表示されています。

```
Device#show platform software yang-management process monitor

COMMAND           PID S    VSZ   RSS %CPU %MEM     ELAPSED
confd           28728 S 860396 168496 42.2  4.2       00:12
confd-startup.s 28448 S  19664  4496  0.2  0.1       00:12
dmiauthd        29499 S 275356 23340  0.2  0.5       00:10
ndbmand         29321 S 567232 65564  2.1  1.6       00:11
nesd            29029 S 189952 14224  0.1  0.3       00:11
nginx           29711 S 332288 18420  0.6  0.4       00:09
nginx           29717 S 337636 12216  0.0  0.3       00:09
pubd            28237 S 631848 68624  2.1  1.7       00:13
syncfd          28776 S 189656 16744  0.2  0.4       00:12
```

AAA と RESTCONF インターフェイスが設定され、nginx プロセスと関連する DMI プロセスが実行中になった後、デバイスは RESTCONF 要求を受信できる状態になります。

NETCONF/RESTCONF セッションのステータスを表示するには、**show netconf-yang sessions** コマンドを使用します。

```
Device# show netconf-yang sessions

R: Global-lock on running datastore
C: Global-lock on candidate datastore
S: Global-lock on startup datastore

Number of sessions : 1

session-id  transport    username              source-host            global-lock
--------------------------------------------------------------------------------
19          netconf-ssh  admin                 2001:db8::1             None
```

NETCONF/RESTCONF セッションに関する詳細情報を表示するには、**show netconf-yang sessions detail** コマンドを使用します。

```
Device# show netconf-yang sessions detail

R: Global-lock on running datastore
C: Global-lock on candidate datastore
S: Global-lock on startup datastore

Number of sessions     : 1

session-id             : 19
transport              : netconf-ssh
username               : admin
source-host            : 2001:db8::1
```

```
login-time             : 2018-10-26T12:37:22+00:00
in-rpcs                : 0
in-bad-rpcs            : 0
out-rpc-errors         : 0
out-notifications      : 0
global-lock            : None
```

RESTCONF プロトコルの設定例

例：RESTCONF プロトコルの設定

RESTCONF 要求（HTTPS Verb）：

次に、ターゲット リソースで許可されている HTTPS Verb を示す RESTCONF 要求の例を示します。この例では **logging monitor** コマンドを使用しています。

```
root:~# curl -i -k -X "OPTIONS"
"https://10.85.116.30:443/restconf/data/Cisco-IOS-XE-native:native/logging/monitor/severity"
 \
>      -H 'Accept: application/yang-data+json' \
>      -u 'admin:admin'
HTTP/1.1 200 OK
Server: nginx
Date: Mon, 23 Apr 2018 15:27:57 GMT
Content-Type: text/html
Content-Length: 0
Connection: keep-alive
Allow: DELETE, GET, HEAD, PATCH, POST, PUT, OPTIONS    >>>>>>>>>>>    Allowed methods
Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
Accept-Patch: application/yang-data+xml, application/yang-data+json
Pragma: no-cache

root:~#
```

POST（作成）要求

POST 操作では、ターゲット デバイスに存在しないコンフィギュレーションが作成されます。

（注） 実行コンフィギュレーションで **logging monitor** コマンドを使用できないことを確認してください。

次の POST 要求の例では **logging monitor alerts** コマンドを使用しています。

```
Device:~# curl -i -k -X "POST"
"https://10.85.116.30:443/restconf/data/Cisco-IOS-XE-native:native/logging/monitor" \
>      -H 'Content-Type: application/yang-data+json' \
>      -H 'Accept: application/yang-data+json' \
>      -u 'admin:admin' \
```

```
>      -d $'{
>   "severity": "alerts"
> }'
HTTP/1.1 201 Created
Server: nginx
Date: Mon, 23 Apr 2018 14:53:51 GMT
Content-Type: text/html
Content-Length: 0
Location:
https://10.85.116.30/restconf/data/Cisco-IOS-XE-native:native/logging/monitor/severity
Connection: keep-alive
Last-Modified: Mon, 23 Apr 2018 14:53:51 GMT
Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
Etag: 1524-495231-97239
Pragma: no-cache

Device:~#
```

PUT：（作成または置換）要求：

指定されたコマンドがデバイスに存在しない場合は、POST 要求によって作成されます。ただし、実行コンフィギュレーションにすでに存在する場合は、この要求によってコマンドが置き換えられます。

次の PUT 要求の例では **logging monitor warnings** コマンドを使用しています。

```
Device:~# curl -i -k -X "PUT"
"https://10.85.116.30:443/restconf/data/Cisco-IOS-XE-native:native/logging/monitor/severity"
 \
>      -H 'Content-Type: application/yang-data+json' \
>      -H 'Accept: application/yang-data+json' \
>      -u 'admin:admin' \
>      -d $'{
>   "severity": "warnings"
> }'
HTTP/1.1 204 No Content
Server: nginx
Date: Mon, 23 Apr 2018 14:58:36 GMT
Content-Type: text/html
Content-Length: 0
Connection: keep-alive
Last-Modified: Mon, 23 Apr 2018 14:57:46 GMT
Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
Etag: 1524-495466-326956
Pragma: no-cache

Device:~#
```

PATCH：（更新）要求

次の PATCH 要求の例では **logging monitor informational** コマンドを使用しています。

```
Device:~# curl -i -k -X "PATCH"
"https://10.85.116.30:443/restconf/data/Cisco-IOS-XE-native:native" \
>      -H 'Content-Type: application/yang-data+json' \
>      -H 'Accept: application/yang-data+json' \
>      -u 'admin:admin' \
>      -d $'{
>   "native": {
```

```
>     "logging": {
>       "monitor": {
>         "severity": "informational"
>       }
>     }
>   }
> }'
HTTP/1.1 204 No Content
Server: nginx
Date: Mon, 23 Apr 2018 15:07:56 GMT
Content-Type: text/html
Content-Length: 0
Connection: keep-alive
Last-Modified: Mon, 23 Apr 2018 15:07:56 GMT
Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
Etag: 1524-496076-273016
Pragma: no-cache
Device:~#
```

GET 要求（読み取り）

次の GET 要求の例では **logging monitor informational** コマンドを使用しています。

```
Device:~# curl -i -k -X "GET"
"https://10.85.116.30:443/restconf/data/Cisco-IOS-XE-native:native/logging/monitor/severity"
 \
>      -H 'Accept: application/yang-data+json' \
>      -u 'admin:admin'
HTTP/1.1 200 OK
Server: nginx
Date: Mon, 23 Apr 2018 15:10:59 GMT
Content-Type: application/yang-data+json
Transfer-Encoding: chunked
Connection: keep-alive
Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
Pragma: no-cache

{
  "Cisco-IOS-XE-native:severity": "informational"
}
Device:~#
```

DELETE 要求（コンフィギュレーションの削除）

```
Device:~# curl -i -k -X "DELETE"
"https://10.85.116.30:443/restconf/data/Cisco-IOS-XE-native:native/logging/monitor/severity"
 \
>      -H 'Content-Type: application/yang-data+json' \
>      -H 'Accept: application/yang-data+json' \
>      -u 'admin:admin'
HTTP/1.1 204 No Content
Server: nginx
Date: Mon, 23 Apr 2018 15:26:05 GMT
Content-Type: text/html
Content-Length: 0
Connection: keep-alive
```

```
Last-Modified: Mon, 23 Apr 2018 15:26:05 GMT
Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
Etag: 1524-497165-473206
Pragma: no-cache

linux_host:~#
```

RESTCONF 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
IOS-XE、IOS-XR、および NX-OS プラットフォームのさまざまなリリースの YANG データ モデル	開発者に分かりやすい方法で Cisco YANG モデルにアクセスするには、GitHub リポジトリを複製し、vendor/cisco サブディレクトリに移動します。ここでは、IOS XE、IOS-XR、および NX-OS プラットフォームのさまざまなリリースのモデルを使用できます。

標準および RFC

標準/RFC	タイトル
RFC 6020	*YANG：Network Configuration Protocol（NETCONF）向けデータ モデリング言語*
RFC 8040	*RESTCONF プロトコル*
RFC 8072	*YANG パッチメディアタイプ*

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。	https://www.cisco.com/c/en/us/support/index.html

RESTCONF 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 : RESTCONF* プロトコルの機能情報

機能名	リリース	機能情報
RESTCONF プロトコル	Cisco IOS XE Everest 16.6.1	RESTCONF は、YANG モデルで定義されている設定データ、状態データ、データモデル固有の RPC の操作およびイベント通知にアクセスするための、標準メカニズムに基づくプログラマチック インターフェイスを提供します。 この機能が次のプラットフォームで追加されました。 • Cisco 4000 シリーズ サービス統合型ルータ • Cisco ASR 1000 アグリゲーション サービス ルータ • Cisco Cloud Services Router 1000V シリーズ 次のコマンドが導入または変更されました：**ip http server** および **restconf**
	Cisco IOS XE Fuji 16.8.1a	Cisco IOS XE Fuji 16.8.1a では、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 1000 シリーズ サービス統合型ルータ • Cisco ASR 900 シリーズ アグリゲーション サービス ルータ • Cisco ASR 920 シリーズ アグリゲーション サービス ルータ • Cisco Catalyst 3650 シリーズ スイッチ • Cisco Catalyst 3850 シリーズ スイッチ • 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 および 9500 ハイ パフォーマンス シリーズ スイッチ • Cisco cBR-8 コンバージド ブロードバンド ルータ • Cisco Network Convergence System 4200 シリーズ
	Cisco IOS XE Fuji 16.9.2	Cisco IOS XE Fuji 16.9.2 では、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 9200 および 9200L シリーズ スイッチ • Cisco Catalyst 9300L SKU
	Cisco IOS XE Gibraltar 16.11.1	

機能名	リリース	機能情報
		Cisco IOS XE Gibraltar 16.11.1 では、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 9600 シリーズ スイッチ • Cisco Catalyst 9800-CL ワイヤレスコントローラ • Cisco Catalyst 9800-40 ワイヤレスコントローラ • Cisco Catalyst 9800-80 ワイヤレスコントローラ • Cisco Network Convergence System 520 シリーズ
	Cisco IOS XE Gibraltar 16.12.1	この機能は、Cisco IOS XE Gibraltar 16.12.1 で、Cisco Catalyst 9800-L ワイヤレスコントローラに実装されました。
	Cisco IOS XE Amsterdam 17.3.1	この機能は、Cisco IOS XE Amsterdam 17.3.1 で次のプラットフォームに実装されました。 • Cisco Catalyst 8200 シリーズ エッジ プラットフォーム • Cisco Catalyst 8300 シリーズ エッジ プラットフォーム • Cisco Catalyst 8500 および 8500L シリーズ エッジ プラットフォーム
	Cisco IOS XE Bengaluru 17.4.1	この機能は、Cisco IOS XE Bengaluru 17.4.1 で、Cisco Catalyst 8000V Edge ソフトウェアに実装されました。

機能名	リリース	機能情報
RESTCONF YANG パッチのサポート	Cisco IOS XE Amsterdam 17.1.1	RESTCONF は、RFC 8072 で指定されている YANG パッチメディアタイプをサポートしています。 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 1000 シリーズ サービス統合型ルータ • Cisco 4000 シリーズ サービス統合型ルータ • Cisco ASR 900 シリーズ アグリゲーション サービス ルータ • Cisco ASR 1000 アグリゲーション サービス ルータ（ASR1000-RP2、ASR1000-RP3、ASR1001-HX、ASR1001-X、ASR1002-HX、ASR1002-X） • Cisco Catalyst 9200 シリーズ スイッチ • 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 cBR-8 コンバージド ブロードバンド ルータ • Cisco Cloud Services Router 1000V シリーズ • Cisco Network Convergence System 520 シリーズ • Cisco Network Convergence System 4200 シリーズ

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。