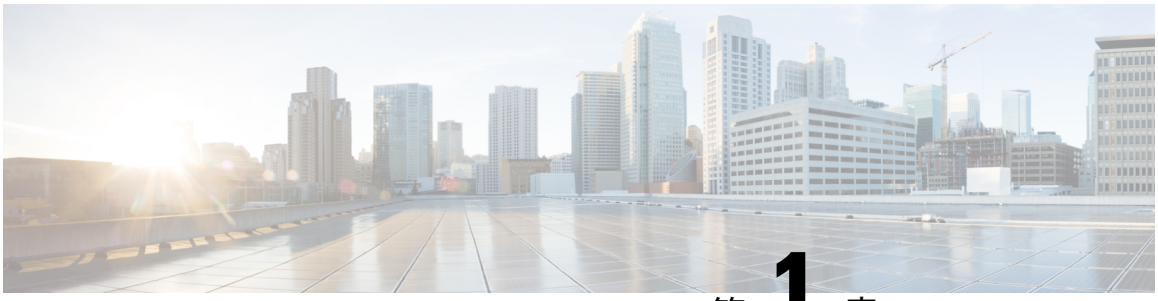
プログラマビリティ コンフィギュレーション ガイド（Cisco IOS XE Gibraltar 16.10.x 向け）

初版：2018 年 11 月 15 日

シスコシステムズ合同会社
〒107-6227 東京都港区赤坂9-7-1 ミッドタウン・タワー
http://www.cisco.com/jp
お問い合わせ先：シスコ コンタクトセンター
0120-092-255 （フリーコール、携帯・PHS含む）
電話受付時間：平日 10:00～12:00、13:00～17:00
http://www.cisco.com/jp/go/contactcenter/

目 次



第 1 章

新機能および変更機能に関する情報

次の表は、新機能および変更機能、サポート対象のプラットフォーム、および機能へのリンクをまとめたものです。

表 *1*：新機能および変更機能に関する情報

機能	リリースとプラットフォーム
プロビジョニング	

機能	リリースとプラットフォーム
ゼロ タッチ プロビジョニング	Cisco IOS XE Everest 16.5.1a • 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 9500 シリーズ スイッチ Cisco IOS XE Everest 16.5.1b • Cisco 4000 シリーズ サービス統合型ルータ Cisco IOS XE Everest 16.6.2 • Cisco Catalyst 9400 シリーズ スイッチ Cisco IOS XE Fuji 16.7.1 • Cisco ASR 1000 アグリゲーション サービス ルータ（ASR1001-X、ASR1001-HX、ASR1002-X、ASR1002-HX） Cisco IOS XE Fuji 16.8.1a • Cisco Catalyst 9500 ハイ パフォーマンス シリーズ スイッチ Cisco IOS XE Fuji 16.8.2 • Cisco ASR 1000 アグリゲーション サービス ルータ（ASR1004、ASR1006、ASR1006-X、ASR1009-X、ASR1013）

機能	リリースとプラットフォーム
ゼロ タッチ プロビジョニング：HTTP コピー	Cisco IOS XE Everest 16.6.1 • Cisco 4000 シリーズ サービス統合型ルータ • 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 9500 シリーズ スイッチ Cisco IOS XE Everest 16.6.2 • Cisco Catalyst 9400 シリーズ スイッチ Cisco IOS XE Fuji 16.7.1 • Cisco ASR 1000 アグリゲーション サービス ルータ（ASR1001-X、ASR1001-HX、ASR1002-X、ASR1002-HX） Cisco IOS XE Fuji 16.8.1a • Cisco Catalyst 9500 ハイ パフォーマンス シリーズ スイッチ Cisco IOS XE Fuji 16.8.2 • Cisco ASR 1000 アグリゲーション サービス ルータ（ASR1004、ASR1006、ASR1006-X、ASR1009-X、ASR1013）

機能	リリースとプラットフォーム
iPXE	Cisco IOS XE Denali 16.3.2 および Cisco IOS XE Everest 16.5.1a • Cisco Catalyst 3650 シリーズ スイッチ • Cisco Catalyst 3650 シリーズ スイッチ Cisco IOS XE Everest 16.6.1 • Cisco Catalyst 9300 シリーズ スイッチ • Cisco Catalyst 9500 シリーズ スイッチ Cisco IOS XE Everest 16.6.2 • Cisco Catalyst 9400 シリーズ スイッチ Cisco IOS XE Fuji 16.8.1a • Cisco Catalyst 9500 ハイ パフォーマンス シリーズ スイッチ Cisco IOS XE Fuji 16.9.2 • Cisco Catalyst 9200 シリーズ スイッチ
シェルとスクリプト化	

機能	リリースとプラットフォーム
ゲスト シェル	Cisco IOS XE Everest 16.5.1a • 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 9500 シリーズ スイッチ Cisco IOS XE Everest 16.5.1b • Cisco 4000 シリーズ サービス統合型ルータ Cisco IOS XE Everest 16.6.2 • Cisco Catalyst 9400 シリーズ スイッチ Cisco IOS XE Fuji 16.7.1 • Cisco ASR 1000 アグリゲーション サービス ルータ（ASR1001-X、ASR1001-HX、ASR1002-X、ASR1002-HX） • Cisco Cloud Services Router 1000V シリーズ Cisco IOS XE Fuji 16.8.1a • Cisco Catalyst 9500 ハイ パフォーマンス シリーズ スイッチ

機能	リリースとプラットフォーム
Python API	Cisco IOS XE Everest 16.5.1a • 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 9500 シリーズ スイッチ Cisco IOS XE Everest 16.5.1b • Cisco 4000 シリーズ サービス統合型ルータ Cisco IOS XE Everest 16.6.2 • Cisco Catalyst 9400 シリーズ スイッチ Cisco IOS XE Fuji 16.7.1 • Cisco ASR 1000 アグリゲーション サービス ルータ（ASR1001-X、ASR1001-HX、ASR1002-X、ASR1002-HX） • Cisco Cloud Services Router 1000V シリーズ Cisco IOS XE Fuji 16.8.1a • Cisco Catalyst 9500 ハイ パフォーマンス シリーズ スイッチ

機能	リリースとプラットフォーム
Python CLI モジュール	Cisco IOS XE Everest 16.5.1a • 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 9500 シリーズ スイッチ Cisco IOS XE Everest 16.5.1b • Cisco 4000 シリーズ サービス統合型ルータ Cisco IOS XE Everest 16.6.2 • Cisco Catalyst 9400 シリーズ スイッチ Cisco IOS XE Fuji 16.7.1 • Cisco ASR 1000 アグリゲーション サービス ルータ（ASR1001-X、ASR1001-HX、ASR1002-X、ASR1002-HX） • Cisco Cloud Services Router 1000V シリーズ Cisco IOS XE Fuji 16.8.1 • 最低 4 GB の RAM を搭載した Cisco 4000 シリーズ サービス統合型ルータ モデル • Cisco ASR 1000 アグリゲーション サービス ルータ（ASR1004、ASR1006、ASR1006-X、ASR1009-X、ASR1013） Cisco IOS XE Fuji 16.8.1a • Cisco Catalyst 9500 ハイ パフォーマンス シリーズ スイッチ

機能	リリースとプラットフォーム
EEM Python モジュール	Cisco IOS XE Everest 16.5.1a • 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 9500 シリーズ スイッチ Cisco IOS XE Everest 16.5.1b • Cisco 4000 シリーズ サービス統合型ルータ Cisco IOS XE Everest 16.6.2 • Cisco Catalyst 9400 シリーズ スイッチ Cisco IOS XE Fuji 16.7.1 • Cisco ASR 1000 アグリゲーション サービス ルータ（ASR1001-X、ASR1001-HX、ASR1002-X、ASR1002-HX） • Cisco Cloud Services Router 1000V シリーズ Cisco IOS XE Fuji 16.8.1a • Cisco Catalyst 9500 ハイ パフォーマンス シリーズ スイッチ
モデル駆動型プログラマビリティ	

機能	リリースとプラットフォーム
NETCONF ネットワーク管理インターフェイス	Cisco IOS XE Denali 16.3.1 • Cisco Catalyst 3650 シリーズ スイッチ • Cisco Catalyst 3850 シリーズ スイッチ • Cisco 4000 シリーズ サービス統合型ルータ Cisco IOS XE Everest 16.5.1a • Cisco Catalyst 9300 シリーズ スイッチ • Cisco Catalyst 9500 シリーズ スイッチ Cisco IOS XE Everest 16.6.2 • Cisco Catalyst 9400 シリーズ スイッチ Cisco IOS XE Fuji 16.7.1 • Cisco ASR 900 シリーズ アグリゲーション サービス ルータ • Cisco Network Convergence System 4200 シリーズ Cisco IOS XE Fuji 16.9.2 • Cisco Catalyst 9200 シリーズ スイッチ Cisco IOS XE ジブラルタル 16.10.1 • Cisco Catalyst 9800 シリーズ ワイヤレス コントローラ • Cisco IR1101 耐環境性能 サービス統合型ルータ • Cisco Network Convergence System 520 シリーズ

機能	リリースとプラットフォーム
NETCONF および RESTCONF IPv6 のサポート	Cisco IOS XE Fuji 16.8.1a • Cisco 1000 シリーズ サービス統合型ルータ • Cisco ASR 1000 シリーズ アグリゲーション サービス 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 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 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 ISR 4000 シリーズ サービス統合型ルータ
gNMI プロトコル	Cisco IOS XE Fuji 16.8.1a • 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

機能	リリースとプラットフォーム
モデルベースの AAA	Cisco IOS XE Fuji 16.8.1 • 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 1000 シリーズ アグリゲーション サービス 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 Cloud Services Router 1000V シリーズ • Cisco Network Convergence System 4200 Cisco IOS XE Fuji 16.8.1a • 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 シリーズ スイッチ

機能	リリースとプラットフォーム
モデル駆動型テレメトリ NETCONF ダイヤルイン	

機能	リリースとプラットフォーム
	Cisco IOS XE Everest 16.6.1 • 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 9500 シリーズ スイッチ Cisco IOS XE Everest 16.6.2 • Cisco Catalyst 9400 シリーズ スイッチ Cisco IOS XE Fuji 16.7.1 • Cisco 4000 シリーズ サービス統合型ルータ • Cisco ASR 1000 シリーズ アグリゲーション サービス ルータ（ASR1001-HX、ASR1001-X、ASR1002-HX、ASR1002-X） Cisco IOS XE Fuji 16.8.1 • Cisco ASR 1000 RP2 および RP3 シリーズ アグリゲーション サービス ルータ Cisco IOS XE Fuji 16.8.1a • Cisco 9500 ハイ パフォーマンス シリーズ スイッチ Cisco IOS XE Fuji 16.9.1 • 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 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 Catalyst 9200 シリーズ スイッチ Cisco IOS XE ジブラルタル 16.10.1 • Cisco IR1101 耐環境性能 サービス統合型ルータ • Cisco クラウド サービス ルータ 1000v • Cisco Network Convergence System 520 シリーズ

機能	リリースとプラットフォーム
モデル駆動型テレメトリ gRPC ダイヤルアウト	Cisco IOS XE Gibraltar 16.10.1 では、gRPC ダイヤルアウト機能が次のプラットフォームで導入されました。 • Cisco 1000 シリーズ サービス統合型ルータ • Cisco IR1101 耐環境性能 サービス統合型ルータ • Cisco ASR 1000 シリーズ アグリゲーション サービス 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 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 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 Cloud Services Router 1000V シリーズ • Cisco Network Convergence System 4200 • Cisco Network Convergence System 520

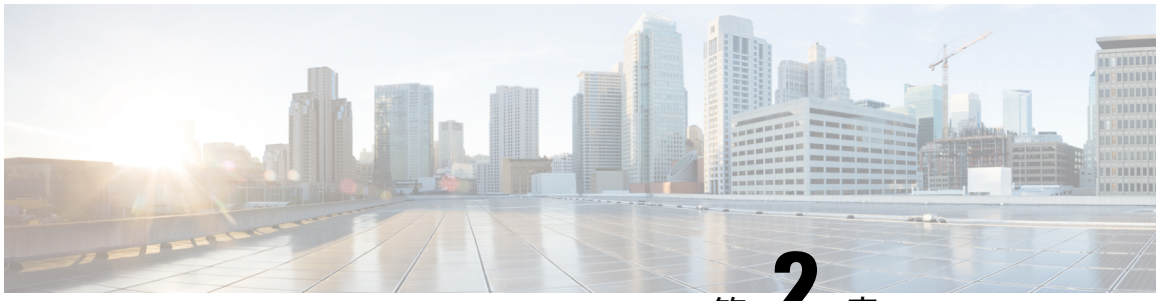
機能	リリースとプラットフォーム
サービス中モデル更新プログラム	Cisco IOS XE Everest 16.5.1a • Cisco Catalyst 9300 シリーズ スイッチ • Cisco Catalyst 9500 シリーズ スイッチ Cisco IOS XE Everest 16.5.1b • Cisco 4000 シリーズ サービス統合型ルータ Cisco IOS XE Everest 16.6.1 • Cisco Catalyst 3650 シリーズ スイッチ • Cisco Catalyst 3850 シリーズ スイッチ Cisco IOS XE Everest 16.6.2 • Cisco Catalyst 9400 シリーズ スイッチ Cisco IOS XE Fuji 16.7.1 • Cisco ASR 1000 アグリゲーション サービス ルータ Cisco IOS XE Fuji 16.8.1a • Cisco Catalyst 9500 ハイ パフォーマンス シリーズ スイッチ

機能	リリースとプラットフォーム
RESTCONF ネットワーク管理インターフェイス	Cisco IOS XE Everest 16.6.1 • 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（ASR1001-HX および ASR1002-HX） • Cisco Cloud Services Router 1000V シリーズ Cisco IOS XE Fuji 16.7.1 • Cisco ASR 9xx シリーズ アグリゲーション サービス ルータ • Cisco Network Convergence System 4200 シリーズ Cisco IOS XE Fuji 16.8.1a • 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 シリーズ スイッチ Cisco IOS XE Fuji 16.9.2 • Cisco Catalyst 9200 シリーズ スイッチ Cisco IOS XE ジブラルタル 16.10.1 • Cisco Network Convergence System 520 シリーズ • Cisco IR1101 耐環境性能 サービス統合型ルータ
アプリケーション ホスティング	
アプリケーション ホスティング	Cisco IOS XE Fuji 16.9.1 • 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
OpenFlow	

機能	リリースとプラットフォーム
OpenFlow	Cisco IOS XE Fuji 16.9.1 • Catalyst 9300 シリーズ スイッチ • Catalyst 9400 シリーズ スイッチ • Catalyst 9500 シリーズ スイッチ • Catalyst 9500 シリーズ ハイ パフォーマンス スイッチ

第 I 部

プロビジョニング



第 2 章

ゼロ タッチ プロビジョニング

ネットワーク プロビジョニングの課題に対応するため、シスコは、ゼロ タッチ プロビジョニング モデルを導入しました。このモジュールでは、ゼロ タッチ プロビジョニング機能について説明します。

（注） ゼロ タッチ プロビジョニング機能は自動的に有効になり、設定は不要です。

ゼロ タッチ プロビジョニングについて

ゼロ タッチ プロビジョニングの概要

ゼロ タッチ プロビジョニングは、異機種混在ネットワーク環境でのネットワーク デバイス プロビジョニングを自動化する、オープン ブートストラップ インターフェイスを提供します。

ゼロ タッチプロビジョニングをサポートするデバイスが起動し、スタートアップ コンフィギュレーションが見つからない場合（初期インストール時）、デバイスはゼロ タッチ プロビジョニング モードに入ります。デバイスは、Dynamic Host Control Protocol（DHCP）サーバを検索し、インターフェイスの IP アドレス、ゲートウェイ、ドメイン ネーム システム（DNS）サーバの IP アドレスをブートストラップして、ゲスト シェルを有効にします。次にデバイスは HTTP/TFTP サーバの IP アドレスまたは URL を取得し、HTTP/TFTP サーバからデバイスを構成する Python スクリプトをダウンロードします。

ゲスト シェルは、Python スクリプトを実行するための環境を提供します。ゲスト シェルは、ダウンロードした Python スクリプトを実行して、初期構成をデバイスに適用します。

初期プロビジョニングが完了したら、ゲスト シェルは有効化されたままになります。詳細については、「ゲスト シェル」の章を参照してください。

（注） ゼロ タッチ プロビジョニングが失敗した場合、デバイスは自動インストールにフォールバックして、コンフィギュレーションファイルをロードします。詳細については、「Using AutoInstall and Setup」を参照してください。

ゼロ タッチ プロビジョニングのための DHCP サーバの設定

ゼロタッチプロビジョニングでは、プロビジョニングされる新しいデバイスと同じネットワークで DHCP サーバを実行する必要があります。ゼロ タッチ プロビジョニングは、管理用ポートとインバンド ポートの両方でサポートされます。

新しいデバイスをオンにすると、そのデバイスは、Python スクリプトが存在する HTTP/TFTP サーバの IP アドレス情報と Python スクリプトのフォルダ パスを DHCP サーバから取得します。Python スクリプトの詳細については、「Python API」および「Python CLI モジュール」の各章を参照してください。

DHCP サーバは、次のオプションで DHCP 検出イベントに応答します。

- オプション 150：（任意）管理ネットワーク上の、実行される Python スクリプトをホストしている HTTP/TFTP サーバを指す IP アドレスの一覧が含まれます。
- オプション 67：HTTP/TFTP サーバ上の Python スクリプトのファイル パスが含まれます。

これらの DHCP オプションを受信すると、デバイスは、HTTP/TFTP サーバに接続して Python スクリプトをダウンロードします。この時点で、デバイスは HTTP/TFTP サーバに到達するルートを持たないため、DHCP サーバによって提供されるデフォルトのルートを使用します。

DHCPv6 のサポート

Cisco IOS XE Fuji 16.9.1 では、Dynamic Host Control Protocol バージョン 6（DHCPv6）のサポートがゼロタッチプロビジョニング機能に追加されました。DHCPv6 はデフォルトで有効になっており、スタートアップ コンフィギュレーションなしでブートするすべてのデバイスで機能します。

（注） DHCPv6 は Catalyst 9300 および 9500 シリーズ スイッチでのみサポートされます。

DHCPv6 は、Python スクリプトの TFTP と HTTP の両方のダウンロードによってサポートされています。Python スクリプトの HTTP または TFTP のダウンロードが失敗した場合、デバイスは開始時点（設定なしの状態）に戻ります。DHCPv4 と DHCPv6 の両方が機能するためには、正しい HTTP ファイル パスが DHCP 設定で使用できる必要があります。

同じインターフェイスに IPv4 と IPv6 の両方のアドレスがあるか、またはネットワーク内に 2 つの異なるインターフェイスがあることが考えられます。つまり、一方は IPv4 トラフィック

を受信でき、他方はIPv6トラフィックを受信できます。導入環境ではDHCPv4またはDHCPv6オプションのいずれかを使用することをお勧めします。

次に、DHCPv4: /etc/dhcp/dhcpd.conf の例を示します。

```
host <hostname> {
  hardware ethernet xx:xx:xx:xx:xx:xx;
  option dhcp-client-identifier "xxxxxxxxxxxxx";
  option host-name "<hostname>".
  option log-servers x.x.x.x;
  fixed-address x.x.x.x;
  if option vendor-class-identifier = "..." {
    option vendor-class-identifier "...";
    if exists user-class and option user-class = "iPXE" {
      filename "http://x.x.x.x/…/<image>";
    } else {
      filename "http://x.x.x.x/…/<script-name>";
    }
  }
}
```

次に、ISC DHCPv6 サーバの設定例を示します。

```
 option dhcp6.bootfile-url  "http://[2001:DB8::21]/sample_day0_script.py";
```

ゼロ タッチ プロビジョニングの構成例

TFTP コピーを使用しての管理ポートにおける DHCP サーバ設定の例

次に、デバイスの管理ポート経由で接続されている場合に TFTP コピーを使用して行う DHCP サーバ設定の例を示します。

```
Device> enable
Device# configure terminal
Device(config)# ip dhcp excluded-address 10.1.1.1
Device(config)# ip dhcp excluded-address vrf Mgmt-vrf 10.1.1.1 10.1.1.10
Device(config)# ip dhcp pool pnp_device_pool
Device(config-dhcp)# vrf Mgmt-vrf
Device(config-dhcp)# network 10.1.1.0 255.255.255.0
Device(config-dhcp)# default-router 10.1.1.1
Device(config-dhcp)# option 150 ip 203.0.113.254
Device(config-dhcp)# option 67 ascii /sample_python_dir/python_script.py
Device(config-dhcp)# exit
Device(config)# interface gigabitethernet 1/0/2
Device(config-if)# no ip dhcp client request tftp-server-address
Device(config-if)# end
```

HTTP コピーを使用しての管理ポートにおける DHCP サーバ設定の例

次に、デバイスの管理ポート経由で接続されている場合に HTTP コピーを使用して行う DHCP サーバ設定の例を示します。

```
Device> enable
Device# configure terminal
Device(config)# ip dhcp pool pnp_device_pool
Device(config-dhcp)# vrf Mgmt-vrf
Device(config-dhcp)# network 10.1.1.0 255.255.255.0
Device(config-dhcp)# default-router 10.1.1.1
Device(config-dhcp)# option 67 ascii http://198.51.100.1:8000/sample_python_2.py
Device(config-dhcp)# end
```

TFTP コピーを使用したインバンド ポートでのサンプル DHCP サーバ構成

次に示すのは、デバイスのインバンド ポート経由で接続されている場合の、TFTP コピーを使用したサンプル DHCP サーバ構成です。

```
Device> enable
Device# configure terminal
Device(config)# ip dhcp excluded-address 10.1.1.1
Device(config)# ip dhcp pool pnp_device_pool
Device(config-dhcp)# network 10.1.1.0 255.255.255.0
Device(config-dhcp)# default-router 10.1.1.1
Device(config-dhcp)# option 150 ip 203.0.113.254
Device(config-dhcp)# option 67 ascii /sample_python_dir/python_script.py
Device(config-dhcp)# exit
Device(config)# interface gigabitethernet 1/0/2
Device(config-if)# no ip dhcp client request tftp-server-address
Device(config-if)# end
```

HTTP コピーを使用したインバンド ポートでのサンプル DHCP サーバ構成

次に示すのは、デバイスのインバンド ポート経由で接続されている場合の、HTTP コピーを使用したサンプル DHCP サーバ構成です。

```
Device> enable
Device# configure terminal
Device(config)# ip dhcp excluded-address 10.1.1.1
Device(config)# ip dhcp pool pnp_device_pool
Device(config-dhcp)# network 10.1.1.0 255.255.255.0
Device(config-dhcp)# default-router 10.1.1.1
Device(config-dhcp)# option 67 ascii http://192.0.2.1:8000/sample_python_2.py
Device(config-dhcp)# end
```

Linux Ubuntu デバイス上でのサンプル DHCP サーバの構成

次の DHCP サーバ構成例は、サーバがデバイスの管理ポートまたはインバンド ポートのどちらかに接続されていることと、Python スクリプトが TFTP サーバからコピーされることを示しています。

```
root@ubuntu-server:/etc/dhcp# more dhcpd.conf
subnet 10.1.1.0 netmask 255.255.255.0 {
range 10.1.1.2  10.1.1.255;
      host 3850 {
          fixed-address                      10.1.1.246 ;
          hardware ethernet                  CC:D8:C1:85:6F:00;
          option bootfile-name !<opt 67>     " /python_dir/python_script.py";
         option tftp-server-name !<opt 150>  "203.0.113.254";
      }
  }
```

次のサンプル DHCP 構成は、Python スクリプトが HTTP サーバからデバイスにコピーされることを示しています。

```
Day0_with_mgmt_port_http
------------------------
subnet 192.168.1.0 netmask 255.255.255.0 {
 range 192.168.1.2 192.168.1.255;
      host C2-3850 {
          fixed-address                   192.168.1.246 ;
          hardware ethernet               CC:D8:C1:85:6F:00;
          option bootfile-name            "http://192.168.1.46/sample_python_2.py";
      }
  }
```

DHCP サーバが実行状態になったら、管理ネットワーク接続デバイスを起動します。これにより構成の残りの部分は自動的に実行されます。

TFTP コピーを使用する管理ポートでの DHCPv6 サーバ設定の例

次に、デバイスの管理ポート経由で接続されている場合に TFTP コピーを使用して行う DHCPv6 サーバ設定の例を示します。

```
Device> enable
Device# configure terminal
Device(config)# ipv6 dhcp pool ztp
Device(config-dhcpv6)# address prefix 2001:DB8::1/64
Device(config-dhcpv6)# domain-name cisco.com
Device(config-dhcpv6)# bootfile-url tftp://[2001:db8::46]/sample_day0_script.py
Device(config-dhcpv6)# exit
Device(config)# interface vlan 20
Device(config-if)# ipv6 dhcp server ztp
Device(config-if)# end
```

サンプルの Python プロビジョニング スクリプト

次に示すのは、HTTP サーバまたは TFTP サーバのいずれかから使用できるサンプル Python スクリプトです。

```
print "\n\n *** Sample ZTP Day0 Python Script *** \n\n"

# Importing cli module
import cli


print "\n\n *** Executing show platform  *** \n\n"
cli_command = "show platform"
cli.executep(cli_command)

print "\n\n *** Executing show version *** \n\n"
cli_command = "show version"
cli.executep(cli_command)

print "\n\n *** Configuring a Loopback Interface *** \n\n"
cli.configurep(["interface loop 100", "ip address 10.10.10.10 255.255.255.255", "end"])


print "\n\n *** Executing show ip interface brief  *** \n\n"
cli_command = "sh ip int brief"
cli.executep(cli_command)

print "\n\n *** ZTP Day0 Python Script Execution Complete *** \n\n"
```

ゼロ タッチ プロビジョニングのブート ログ

次のゼロ タッチ プロビジョニングのブート ログでは、ゲスト シェルが正常に有効にされ、Python スクリプトがゲスト シェルにダウンロードされ、ゲスト シェルがダウンロードした Python スクリプトを実行してデバイスをデイ ゼロに設定していることが示されています。

```
% failed to initialize nvram
! <This message indicates that the startup configuration
is absent on the device. This is the first indication that the Day Zero work flow is
going to start.>

This product contains cryptographic features and is subject to United
States and local country laws governing import, export, transfer and
use. Delivery of Cisco cryptographic products does not imply
third-party authority to import, export, distribute or use encryption.
Importers, exporters, distributors and users are responsible for
compliance with U.S. and local country laws. By using this product you
agree to comply with applicable laws and regulations. If you are unable
to comply with U.S. and local laws, return this product immediately.

A summary of U.S. laws governing Cisco cryptographic products may be found at:
http://www.cisco.com/wwl/export/crypto/tool/stqrg.html

If you require further assistance please contact us by sending email to
export@cisco.com.
```

```
cisco ISR4451-X/K9 (2RU) processor with 7941237K/6147K bytes of memory.
Processor board ID FJC1950D091
4 Gigabit Ethernet interfaces
32768K bytes of non-volatile configuration memory.
16777216K bytes of physical memory.
7341807K bytes of flash memory at bootflash:.
0K bytes of WebUI ODM Files at webui:.

%INIT: waited 0 seconds for NVRAM to be available


         --- System Configuration Dialog ---

Would you like to enter the initial configuration dialog? [yes/no]: %
!!<DO NOT TOUCH. This is Zero-Touch Provisioning>>
Generating 2048 bit RSA keys, keys will be non-exportable...
[OK] (elapsed time was 1 seconds)
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
Guestshell enabled successfully


 *** Sample ZTP Day0 Python Script ***


 *** Configuring a Loopback Interface ***


Line 1 SUCCESS: interface loop 100
Line 2 SUCCESS: ip address 10.10.10.10 255.255.255.255
Line 3 SUCCESS: end


 *** Executing show ip interface brief  ***


Interface              IP-Address      OK? Method Status                Protocol
GigabitEthernet0/0/0   unassigned      YES unset  down                  down
GigabitEthernet0/0/1   unassigned      YES unset  down                  down
GigabitEthernet0/0/2   unassigned      YES unset  down                  down
GigabitEthernet0/0/3   192.168.1.246   YES DHCP   up                    up
GigabitEthernet0       192.168.1.246   YES DHCP   up                    up
Loopback100            10.10.10.10     YES TFTP   up                    up


 *** ZTP Day0 Python Script Execution Complete ***



Press RETURN to get started!
```

デイ ゼロ プロビジョニングが完了すると、IOS プロンプトがアクセス可能になります。

ゼロ タッチ プロビジョニングの機能情報

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。

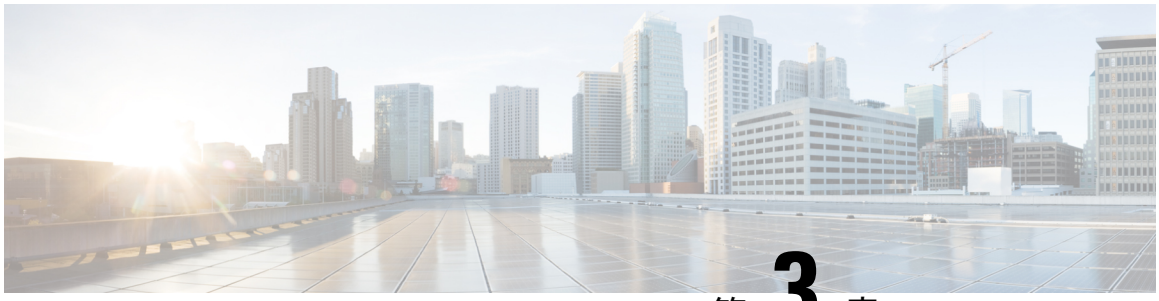
プラットフォームのサポートおよびシスコ ソフトウェア イメージのサポートに関する情報を検索するには、Cisco Feature Navigator を使用します。Cisco Feature Navigator にアクセスするには、www.cisco.com/go/cfn に移動します。Cisco.com のアカウントは必要ありません。

表 *2:* ゼロ タッチ プロビジョニングの機能情報

機能名	リリース	機能情報
ゼロ タッチ プロビジョニング	Cisco IOS XE Everest 16.5.1a Cisco IOS XE Everest 16.5.1b Cisco IOS XE Fuji 16.7.1 Cisco IOS XE Fuji 16.8.2	

機能名	リリース	機能情報
		ネットワーク プロビジョニングの課題に対応するため、シスコは、ゼロ タッチ プロビジョニング モデルを導入しました。 Cisco IOS XE Everest 16.5.1a 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 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 9500 シリーズ スイッチ Cisco IOS XE Everest 16.5.1b 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。 • ゲストシェルをサポートするための、最低 8 GB の RAM を搭載した Cisco 4000 シリーズ サービス統合型ルータ モデル。 Cisco IOS XE Fuji 16.7.1 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 ASR 1000 アグリゲーション サービス ルータ（ASR1001-X、ASR1001-HX、ASR1002-X、ASR1002-HX） Cisco IOS XE Fuji 16.8.2 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 ASR 1000 シリーズ アグリゲーション サービス ルータ（ASR1004、ASR1006、ASR1006-X、ASR1009-X、ASR1013）

機能名	リリース	機能情報
ゼロ タッチ プロビジョニング：HTTP ダウンロード	Cisco IOS XE Fuji 16.8.1	ゼロ タッチ プロビジョニングは、HTTP および TFTP のファイル ダウンロードをサポートします。 Cisco IOS XE Everest 16.8.1 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 4000 シリーズ サービス統合型ルータ • 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 9500 シリーズ スイッチ Cisco IOS XE Fuji 16.8.1a では、この機能は Cisco Catalyst 9500 ハイ パフォーマンス シリーズ スイッチに実装されていました。
ゼロタッチ プロビジョニングのための DHCPv6 のサポート	Cisco IOS XE Fuji 16.9.1	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 Catalyst 9300 シリーズ スイッチ • Cisco Catalyst 9500 シリーズ スイッチ



第 3 章

iPXE

iPXE は、ネットワーク ブーティングのオープン スタンダードである Pre-boot eXecution Environment（PXE）の拡張版です。このモジュールでは、iPXE 機能および設定方法について説明します。

iPXE について

iPXE について

iPXE は、ネットワーク ブーティングのオープン スタンダードである Pre-boot eXecution Environment（PXE）の拡張版です。

iPXE ネットブートは、次を提供します。

- IPv4 および IPv6 プロトコル
- FTP/HTTP/TFTP ブート イメージのダウンロード
- イメージへの埋め込みスクリプト
- Dynamic Host Configuration Protocol バージョン 4（DHCPv4）や DHCPv6 を使用したステートレスおよびステートフル アドレス自動設定（SLAAC）、ブート URI、および IPv6 ルータ アドバタイズメントに応じた DHCPv6 オプションのパラメータ。

ネットブート要件

ネット ブーティングの主な要件は、次のとおりです。

- 適切に設定された DHCP サーバ。
- FTP/HTTP/TFTP サーバ上で使用可能なブート イメージ。
- ネットワーク ベースのソースから起動するように設定されたデバイス。

iPXE の概要

ネットワーク ブートローダは、ネットワーク ベースのソースからのブート処理をサポートします。ブートローダは、HTTP、FTP、またはTFTPサーバにあるイメージを起動します。ネットワーク ブート ソースは、iPXE のようなソリューションを使用して自動検出されます。

iPXE により、オフラインのデバイスのネットワーク ブートが可能になります。ブート モードには次の 3 種類があります。

- iPXE タイムアウト：iPXE ネットワーク ブートを介して起動します。IPXE_TIMEOUT ROMmon 変数を使用して、iPXE ネットワーク ブートのタイムアウトを秒単位で設定します。iPXE タイムアウトを設定するには **boot ipxe timeout** コマンドを使用します。タイムアウト時間を経過すると、デバイス ブートがアクティブになります。
- iPXE 期限なし：iPXE ネットワーク ブートを介して起動します。**boot ipxe forever** コマンドが設定されている場合、デバイスは DHCP 要求を期限なしで送信します。これは iPXE のみを使うブートです（つまり、ブートローダは、有効なDHCP応答を受け取るまでDHCP要求を期限なしで送信するため、デバイス ブートまたはコマンド プロンプトにフォールバックすることはありません）。
- デバイス：設定されているローカル デバイスの BOOT 行を使ってブートします。デバイス ブートが設定された場合、設定されているIPXE_TIMEOUT ROMmon変数は無視されます。次のように指定してデバイス ブートをアクティブ化できます。
  - BOOTMODE=ipxe-forever の場合は、ユーザの介入がなければデバイス ブートがアクティブになりません（ENABLE_BREAK=yes の場合にのみ可能）。
  - BOOTMODE=ipxe-timeout の場合は、IPXE_TIMEOUT 変数で指定した秒数が経過するとデバイス ブートがアクティブになります。
  - BOOTMODE=device の場合は、デバイス ブートがアクティブになります。これはデフォルトのアクティブ モードです。
  - デバイス ブートは CLI を使用してアクティブ化することもできます。

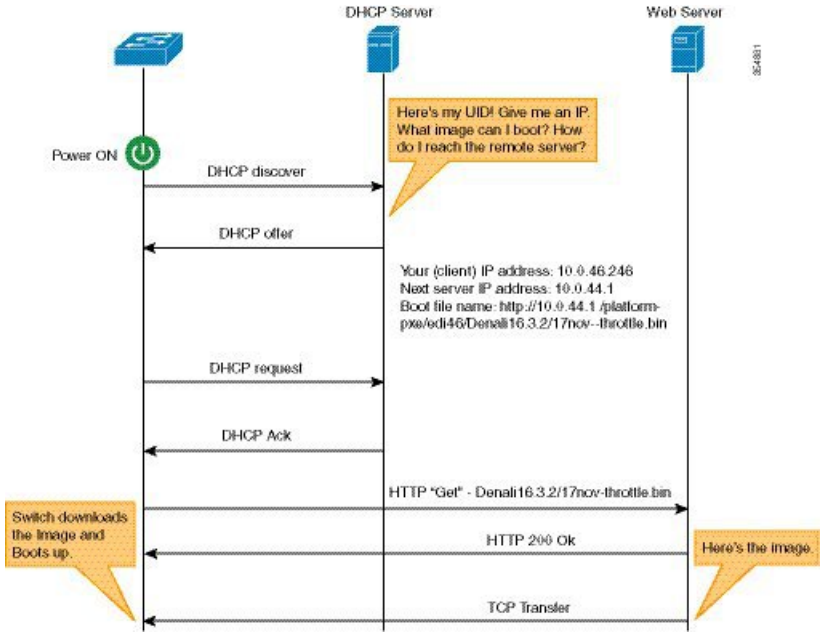
（注） デバイス ブートは、デフォルトのブート モードです。

（注） このマニュアルでは、手動ブートという用語も使われています。手動ブートは、ROMmon のリロードを行うかどうかを決定するフラグです。デバイスが ROMmon モードの場合は、手動で **boot** コマンドを実行する必要があります。

手動ブートを YES に設定した場合は、ROMmon またはデバイス プロンプトがアクティブになります。手動ブートを NO に設定した場合は、autoboot 変数が実行されます。つまり、BOOT 変数で設定された値に従います。

ここでは、iPXE ブートローダの動作について説明します。

図 *1 : iPXE* ブートローダのワークフロー



1. ブートローダは DHCP 検出メッセージを送信し、サーバが応答すると、ブートローダは DHCP 要求を送信します。
2. DHCP 応答には、IP アドレスとのブート ファイル名が含まれています。ブート ファイル名は、ブート イメージが TFTP サーバ（tftp://server/filename）、FTP サーバ（ftp://userid:password@server/filename）、または HTTP サーバ（http://server/filename）から取得されることを示しています。
3. ブートローダがネットワーク ソースからイメージをダウンロードして起動します。
4. DHCP 応答が受信されない場合、ブートローダはブート モードの設定に基づいて、DHCP 要求を期限なしで、または指定された期間の間送信し続けます。タイムアウトが発生すると、ブートローダはデバイスベースのブートに戻ります。設定されたブート モードが **ipxe-forever** の場合のみ、デバイスは DHCP 要求を期限なしで送信します。**ipxe-timeout** ブート モード コマンドが設定されている場合、DHCP 要求は指定された時間にわたって送信され、タイムアウトが経過すると、デバイス ブート モードがアクティブになります。

（注） 現在の iPXE 実装は管理ポート（GigabitEthernet0/0）のみを経由して動作するため、前面パネル ポートを介して送信される DHCP 要求はサポートされていません。

ネットワーク ブートに対して静的なネットワーク設定を使用する場合、ROMmon は次の環境変数を使用します（すべて必須です）。

- BOOT：セミコロン（;）で区切られた起動元の URL。
- IP_ADDRESS：静的に割り当てられたデバイスの IP アドレス。
- DEFAULT_GATEWAY：デバイスのデフォルト ゲートウェイ。
- IP_SUBNET_MASK：IPv4 または IPv6 プレフィックス情報。

  IPv4：WWW.XXX.YYY.ZZZ という形式のデバイスのサブネット マスク（255.255.255.0 など）。

  IPv6：NNN という形式のデバイスのサブネット プレフィックス長（64、112 など）。

手動ブートが無効になっている場合、ブートローダは、設定された ROMmon iPXE 変数の値に基づいて、デバイス ブートを実行するかネットワーク ブートを実行するかを決定します。手動ブートが有効か無効かにかかわらず、ブートローダは BOOTMODE 変数を使用して、デバイス ブートとネットワーク ブートのどちらを実行するかを決定します。手動ブートは、ユーザによって **boot manual switch** コマンドが設定済みであることを意味します。手動ブートが無効になっている場合にデバイスをリロードすると、起動プロセスが自動的に開始されます。

iPXE が無効になっている場合は、デバイスの起動方法の決定に、既存の BOOT 変数の内容が使用されます。BOOT 変数には、ネットワークベースの Uniform Resource Identifier（URI）（たとえば、http://、ftp://、tftp://）が含まれている場合があり、ネットワーク ブートが開始されます。しかし、ネットワーク イメージ パスの取得に DHCP は使用されません。静的なネットワーク設定は、IP_ADDRESS 変数、DEFAULT_GATEWAY 変数、および IP_SUBNET_MASK 変数から取得されます。BOOT 変数には、デバイスのファイル システム ベースのパスが含まれている場合もあり、この場合は、デバイスのファイル システム ベースのブートが開始されます。

起動に使用される DHCP サーバは、製品 ID（PID）（DHCP オプション 60 で判別可能）、シャーシのシリアル番号（DHCP オプション 61 で判別可能）、またはデバイスの MAC アドレスを使用して、デバイスを識別できます。**show inventory** および **show switch** コマンドでもデバイスでこれらの値を表示します。

次に、**show inventory** コマンドの出力例を示します。

```
Device# show inventory

NAME:"c38xx Stack", DESCR:"c38xx Stack"
PID:WS-3850-12X-48U-L, VID:V01 ,  SN: F0C1911V01A

NAME:"Switch 1", DESCR:"WS-C3850-12X48U-L"
PID:WS-C3850-12X48U-L, VID:V01 , SN:F0C1911V01A

NAME:"Switch1 -Power Supply B", DESCR:"Switch1 -Power Supply B"
PID:PWR-C1-1100WAC, VID:V01, SN:LIT1847146Q
```

次に、 **show switch** コマンドの出力例を示します。

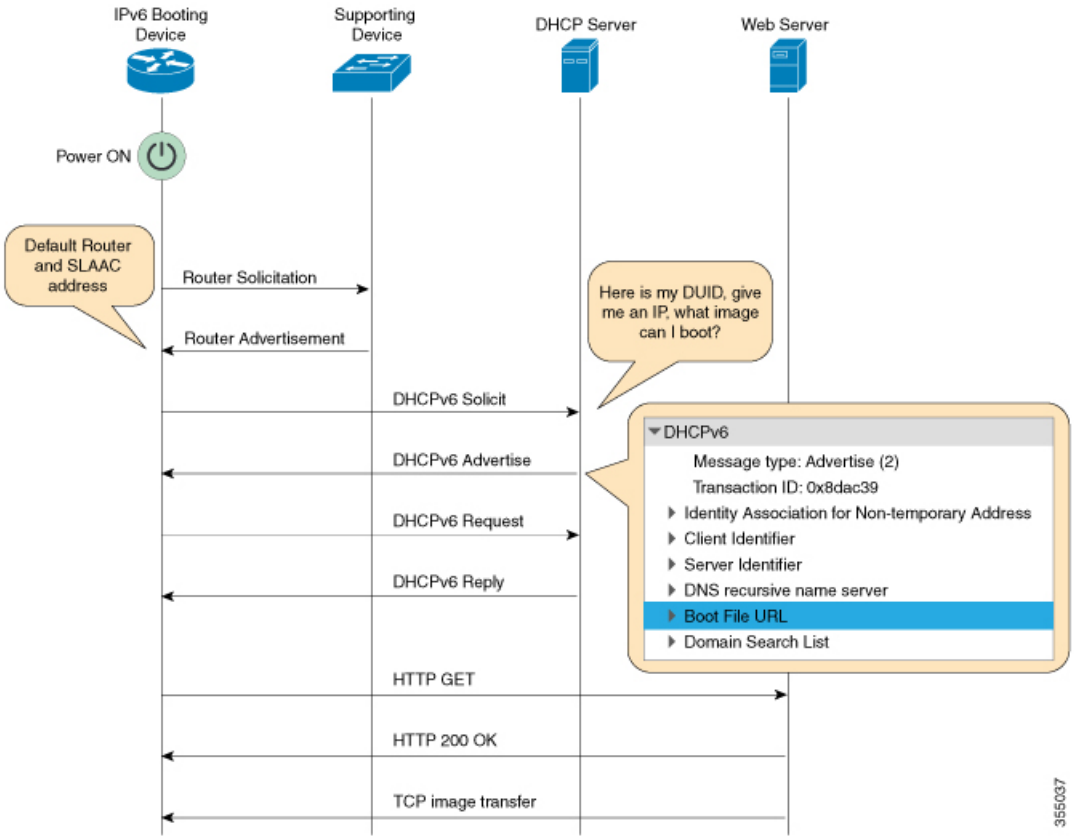
```
Device# show switch

Switch/Stack Mac Address : 046c.9d01.7d80 - Local Mac Address
Mac persistency wait time: Indefinite
                                             H/W   Current
Switch#   Role    Mac Address     Priority Version  State
-----------------------------------------------------------------------------------
 1       Member   046c.9d1e.1a00     1              Ready
 2       Standby  046c.9d01.7d80     1              Ready
*3       Active   f8b7.e24e.9a00     1      P2B     Ready
```

次の ROMmon 変数が iPXE に設定されている必要があります。

- BOOTMODE = ipxe-forever | ipxe-timeout | device
- IPXE_TIMEOUT = seconds

IPv6 iPXE ネットワーク ブート

次の図は、Cisco デバイス上の IPv6 iPXE ネットワーク ブートの動作を表します。



次に、上掲の図の 4 つの要素を説明します。

- IPv6 ブート デバイス：iPXE ブートによって起動するデバイス。

- サポート デバイス：IPv6アドレスで、ルータ アドバタイズメント（RA）メッセージを生成するように設定された Cisco デバイス。

（注） この図では、IPv6 ブート デバイス、サポート デバイス、および DHCP サーバは、同じサブネット上にあります。ただし、サポート デバイスと DHCP サーバが異なるサブネット上にある場合、ネットワーク内にリレーエージェントを設ける必要があります。

- DHCP サーバ：任意の DHCP サーバ。
- Web サーバ：任意の Web サーバ。

この項では、IPv6 iPXE ブート プロセスを説明します。

1. デバイスは、ルータ要請である Internet Control Message Protocol IPv6（ICMPv6）タイプ 133 パケットをローカル サブネット上の IPv6 デバイスに送信します。
2. ローカル サブネット上の IPv6 デバイスは、ルータ アドバタイズメント（RA）メッセージである ICMPv6 タイプ 134 パケットで応答します。ルータ要請メッセージを送信したデバイスは、ステートレス アドレス自動設定（SLAAC）アドレスを完成させるため、RA パケットからデフォルト ルータとプレフィックスの情報を取得します。
3. デバイスは、DHCPv6 要請メッセージを、すべての DHCP エージェントについて、マルチキャスト グループ アドレス ff02::1:2 に送信します。

   次に、iPXE ブートの際の DHCPv6 要請パケットのフィールドの例を示します。

```
DHCPv6
 Message type: Solicit (1)
 Transaction ID: 0x36f5f1
 Client Identifier
 Vendor Class
 Identity Association for Non-Temporary Address
 Option Request
 User Class
 Vendor-specific Information
```

   DHCPv6 要請メッセージには、次の情報が含まれています。

   - DHCP 固有識別子（DUID）：クライアントを識別します。iPXE では、DUID-EN をサポートしています。EN は、エンタープライズ番号（Enterprise Number）の略です。この DUID は、ベンダーに割り当てられた固有の識別子に基づいています。
   - DHCP および DHCPv6 のオプション

4. DHCPv6 サーバが設定されている場合、そのサーバは、128 ビット IPv6 アドレス、ブートファイルの Uniform Resource Identifier（URI）、ドメイン ネーム システム（DNS）サーバおよびドメイン検索リスト、ならびにクライアントとサーバの ID を含む DHCPv6 アドバタイズ パケットで応答します。クライアント ID にはクライアント（この図では IPv6 ブー

トデバイス）の DUID が、サーバ ID には DHCPv6 サーバの DUID が、それぞれ含まれています。

5. それを受け、クライアントは、マルチキャスト グループ アドレス ff02::1:2 に DHCPv6 要求パケットを送信し、アドバタイズされたパラメータを要求します。
6. サーバは、クライアントのリンク ローカル（FE80::）の IPv6 アドレスにユニキャスト DHCPv6 応答を返します。次に、DHCPv6 応答パケットのフィールドの例を示します。

```
DHCPv6
 Message type: Reply (7)
 Transaction ID: 0x790950
 Identity Association for Non-Temporary Address
 Client Identifier
 Server Identifier
 DNS recursive name server
 Boot File URL
 Domain Search List
```

7. 次に、デバイスは、Web サーバに HTTP GET 要求を送信します。
8. 要求されたイメージが指定されたパスで使用可能な場合、Web サーバは、HTTP GET 要求に OK を返します。
9. TCP イメージ転送によりイメージがコピーされ、デバイスが起動します。

ROMmon モードでの IPv6 アドレスの割り当て

DHCP クライアントは、次の優先順位を使用して、ROMmon モードで使用する IPv6 アドレスを決定します。

1. DHCP サーバによって割り当てられたアドレス
2. ステートレス アドレス自動設定（SLAAC）アドレス
3. リンクローカル アドレス
4. サイトローカル アドレス

デバイスは、イメージをブートするのにDHCPサーバによって割り当てられたアドレスを使用します。DHCPv6 サーバがアドレスの割り当てに失敗した場合、デバイスは、SLAAC アドレスの使用を試行します。DHCP サーバによって割り当てられたアドレスと SLAAC アドレスの両方が使用できない場合、デバイスは、リンクローカル アドレスを使用します。ただし、イメージのコピーを正常に行うには、リモート FTP/HTTP/TFTP サーバがデバイスと同じローカル サブネット上にある必要があります。

最初の3つのアドレスが使用できない場合、デバイスは、自動的に生成されるサイトローカル アドレスを使用します。

サポートされる ROMmon 変数

Cisco IOS XE Fuji 16.8.1 では、次の ROMmon 変数がサポートされています。

- BAUD：デバイスのコンソール ボー レートをシスコの標準ボー レート（1200、2400、4800、9600、19200、38400、57600、115200 など）のいずれかに変更します。無効な値はすべて拒否されます。BAUD 変数が設定されていない場合は、デフォルトで 9600 になります。対応する CLI コマンドは、
- ENABLE_BREAK：ROMmon のブレークを有効にします。デフォルト値は NO です。
- MANUAL_BOOT：手動ブートが 1 に設定されている場合、ROMmon またはデバイス プロンプトがアクティブになります。手動ブートが 0 に設定されている場合、デバイスはリロードされますが、ROMmon モードはアクティブになりません。
- SWITCH_IGNORE_STARTUP_CFG：値が 1 の場合は、デバイスでスタートアップ コンフィギュレーションが無視されます。値が設定されていない場合は、値がゼロとみなされます。これは読み取り専用変数であり、IOS のみが変更できます。

iPXE がサポートする DHCP オプション

iPXE ブートは、ROMmon モードで次の DHCPv4 および DHCPv6 オプションをサポートしています。

（注） Catalyst 9000 シリーズ スイッチは、DHCP オプション 60、オプション 77、DHCPv6 オプション 1、オプション 15、およびオプション 16 をサポートしています。DHCP オプション 61 は、Catalyst 9300 および 9500 シリーズ スイッチでのみサポートされています。

- DHCP オプション 60：ベンダー クラス識別子。このオプションには、ROMmon 環境変数 MODEL_NUM の値が設定されます。
- DHCP オプション 61：クライアント識別子。このオプションには、ROMmon 環境変数 SYSTEM_SERIAL_NUM の値が設定されます。

  （注） このオプションは Catalyst 9400 シリーズ スイッチではサポートされていません。

- DHCP オプション 77：ユーザ クラス オプション。このオプションは、DHCP 検出パケットに追加されるもので、iPXE という文字列に等しい値を含んでいます。このオプションは、DHCP サーバからブートするためのイメージを探す iPXE DHCP クライアントを分離する際に使用されます。

次に、ISC DHCP サーバからの DHCPv4 設定で、オプション 77 の使用が示されている例を示します。この例における if 条件は、オプション 77 が存在しており、文字列 iPXE に等しい場合は、イメージのブート ファイルの URI がアドバタイズされることを示します。

```
host Switch2 {
     fixed-address 192.168.1.20 ;
     hardware ethernet CC:D8:C1:85:6F:11 ;
           #user-class = length of string + ASCII code for iPXE
     if exists user-class and option user-class = 04:68:50:58:45 {
         filename "http://192.168.1.146/test-image.bin"
     }
}
```

- DHCPv6 オプション 1：クライアント識別子オプション。このオプションには、RFC 3315 で規定されている ROMmon 環境変数 SYSTEM_SERIAL_NUM の値が設定されます。ROMmon 環境変数で推奨される形式は MAC_ADDR です。
- DHCPv6 オプション 15：ユーザ クラス オプション。このオプションは、DHCPv6 要請メッセージ内の IPv6 ユーザ クラス オプションであり、文字列 iPXE が設定されます。次に、ISC DHCP サーバで定義されているオプション 15 の例を示します。

```
option dhcp6.user-class code 15 = string ;
```

次に、DHCPv6 オプション 15 が使用されている DHCP サーバ設定の例を示します。

```
#Client-specific parameters
host switch1 {
     #assigning a fixed IPv6 address
     fixed-address6 2001:DB8::CAFE ;
     #Client DUID in hexadecimal format contains: DUID-type"2" + "EN=9" + "Chassis
serial number"
    host-identifier option dhcp6.client-id      00:02:00:00:00:09:46:4F:43:31:38:33:
31:58:31:41:53;
    #User class 00:04:69:50:58:45 is len 4 + "iPXE"
    if option dhcp6.user-class = 00:04:69:50:58:45 {
       option dhcp6.bootfile-url
"http://[2001:DB8::461/platform-pxe/edi46/test-image.bin";
    }
}
```

- DHCPv6 オプション 16：ベンダー クラス オプション。デバイスの製品 ID（PID）が含まれています。PID は、**show inventory** コマンドの出力または MODEL_NUM ROMmon 変数から特定できます。オプション 16 は ISC DHCP サーバのデフォルトのオプションではなく、次のように定義することができます。

```
option dhcp6.vendor-class-data code 16 = string;
```

次に、DHCPv6 オプション 16 が使用されている設定例を示します。

```
# Source: dhcpd6ConfigPD

host host1-ipxe6-auto-host1 {
     fixed-address6 2001:DB8::1234;
     host-identifier option dhcp6.client-id 00:02:00:00:00:09:46:4F:
```

```
     43:31:38:33:31:58:31:41:53;
     if option dhcp6.vendor-class-data = 00:00:00:09:00:0E:57:53:2D:
     43:33:38:35:30:2D:32:34:50:2D:4D {
     option dhcp6.bootfile-url
"http://[2001:DB8::46]/platform-pxe/host1/17jan-polaris.bin";
```

次の表で、この出力に表示される重要なフィールドを説明します。

表 *3:* サンプル出力フィールドの説明

フィールド	説明
dhcp6.client-id	クライアントを識別する DHCP 固有識別子（DUID）。
dhcp6.user-class	DHCPv6 オプション 15、ユーザ クラス オプション。
dhcp6.vendor-class-data	DHCPv6 オプション 16、スイッチの製品 ID（PID）を含むベンダー クラス オプション。
dhcp6.bootfile-url	ブート ファイル URI を要求する DHCPv6 オプション 6。

DHCPv6 固有識別子

RFC 3315 によって定義されている DHCPv6 識別子（DUID）には、次の 3 種類があります。

- DUID-LLT：DUID リンク層アドレスと時刻。DHCP デバイスに接続しているネットワーク インターフェイスのリンク層アドレスに、生成された時刻のタイムスタンプが追加されたものです。
- DUID-EN：EN は、エンタープライズ番号（Enterprise Number）の略です。この DUID は、ベンダーに割り当てられた固有の ID に基づいています。
- DUID-LL：DHCP（クライアント/サーバ）デバイスに永久的に接続されているネットワーク インターフェイスのリンク層アドレスを使用して形成される DUID です。

この機能をサポートしているシスコ デバイスは、DHCP クライアント（DHCPv6 要請パケット内のデバイス）を識別するのに DUID-EN（DUID タイプ 2）を使用します。Catalyst 9000 シリーズ スイッチは、DUID-EN だけでなく DUID-LL（DUID タイプ 3）もサポートしています。DUID-EN は優先される型です。ただし、スイッチがこの型を作成できない場合は、DUID-LL が作成されて使用されます。

iPXE の設定方法

iPXE の設定

手順の概要

1. **enable**
2. **configure terminal**
3. • **boot ipxe forever** [*switch number*]
   • **boot ipxe timeout** *seconds* [*switch number*]
4. **boot system** {**switch** *switch-number* | **all**} {**flash:** | **ftp:** | **http:** | **usbflash0** | **tftp:**}
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）。
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	• **boot ipxe forever** [*switch number*]
	• **boot ipxe timeout** *seconds* [*switch number*]
	例：
	`Device(config)# boot ipxe forever switch 2`
	例：
	`Device(config)# boot ipxe timeout 30 switch 2`
		BOOTMODE ROMmon 変数を設定します。
		• **forever** キーワードは、BOOTMODE ROMmon 変数を IPXE-FOREVER として設定します。
		• **timeout** キーワードは、BOOTMODE ROMmon 変数を IPXE-TIMEOUT として設定します。

ステップ 4	**boot system** {**switch** *switch-number*	**all**} {**flash:**	**ftp:**	**http:**	**usbflash0**	**tftp:**}
						例：
						`Device(config)# boot system switch 1 http://192.0.2.42/image-filename`
						または
						`Device(config)# boot system switch 1 http://[2001:db8::1]/image-filename`
							指定した場所からイメージを起動します。
							• リモートの FTP/HTTP/TFTP サーバには、IPv4 または IPv6 アドレスを使用できます。
							• 角かっこ内に IPv6 アドレスを入力する必要があります（RFC 2732 に従って）。そうしない場合、デバイスは起動しません。

	コマンドまたはアクション	目的
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 モードに戻ります。

デバイス ブートの設定

デバイス ブートは、**no boot ipxe** または **default boot ipxe** コマンドのいずれかを使用して設定できます。

手順の概要

1. **enable**
2. **configure terminal**
3. • **no boot ipxe**
   • **default boot ipxe**
4. **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）。
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	• **no boot ipxe**
	• **default boot ipxe**
	例：
	`Device(config)# no boot ipxe`
	例：
	`Device(config)# default boot ipxe`
		デバイス ブートを設定します。デフォルトのブートモードはデバイス ブートです。
		デバイスでデフォルト設定を有効にします。

ステップ 4	**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。

iPXE の設定例

例：iPXE 構成

以下は、デバイスがイメージで起動するまで、DHCP 要求を期限なしで送信するように iPXE を設定する例を示しています。

```
Device# configure terminal
Device(config)# boot ipxe forever switch 2
Device(config)# end
```

以下は、ブート モードを ipxe-timeout に設定する方法の例を示します。設定されているタイムアウト値は 200 秒です。設定されているタイムアウト経過後に iPXE ブート障害が発生する場合、設定されているデバイス ブートがアクティブになります。この例で、設定済みのデバイス ブートは http://[2001:db8::1]/image-filename です。

```
Device# configure terminal
Device(config)# boot ipxe timeout 200 switch 2
Device(config)# boot system http://[2001:db8::1]/image-filename
Device(config)# end
```

サンプルの iPXE ブート ログ

次に示すのは、ROMmon モードのデバイスからのサンプル ブート ログです。ここでは、**ipxe-timeout** コマンドを使用した手動ブートが設定されます。

```
switch: boot

pxemode:(ipxe-timeout) 60s timeout
00267.887 ipxe_get_booturl: Get URL from DHCP; timeout 60s
00267.953 ipxe_get_booturl: trying DHCPv6 (#1) for 10s
IPv4:
           ip addr 192.168.1.246
           netmask 255.255.255.0
           gateway 192.168.1.46
IPv6:
link-local addr fe80::ced8:c1ff:fe85:6f00
site-local addr fec0::ced8:c1ff:fe85:6f00
      DHCP addr 2001:db8::cafe
    router addr fe80::f29e:63ff:fe42:4756
     SLAAC addr 2001:db8::ced8:c1ff:fe85:6f00 /64
Common:
        macaddr cc:d8:c1:85:6f:00
            dns 2001:db8::46
       bootfile
http://[2001:DB8::461/platform-pxe/edi46/17jan-dev.bin--13103--2017-Feb28--13-54-50
         domain cisco.com
00269.321 ipxe_get_booturl: got URL
(http://[2001:DB8::461/platform-pxe/edi46/17jan-dev.bin--13103--2017-Feb-28--13-54-50)
```

```
Reading full image into memory ….…….…….…….…….…….…….…….…….…….…….…….…….…….…….…….…….…...
Bundle Image
-----------------------------------------------
Kernel Address    : 0x5377a7e4
Kernel Size       : 0x365e3c/3563068
Initramfs Address : 0x53ae0620
Initramfs Size    : 0x13a76f0/20608752
Compression Format: mzip
```

iPXE 用のサンプル DHCPv6 サーバ構成

次に示すのは、参考のために Internet Systems Consortium（ISC）の DHCP サーバから取得した DHCPv6 サーバ設定の例です。先頭に文字 # がある行は、続く構成を説明しているコメントです。

```
Default-least-time 600;
max-lease-time-7200;
log-facility local7;

#Global configuration
#domain search list
option dhcp6.domain-search "cisco.com" ;
#User-defined options:new-name code new-code = definition ;
option dhcp6.user-class code 15 = string ;
option dhcp6.vendor-class-data code 16 = string;

subnet6 2001:db8::/64 {
 #subnet range for clients requiring an address
 range6 2001:db8:0000:0000::/64;

#DNS server options
option dhcp6.name-servers 2001:db8::46;

}
#Client-specific parameters
host switch1 {
    #assigning a fixed IPv6 address
    fixed-address6 2001:DB8::CAFE ;
    #Client DUID in hexadecimal that contains: DUID-type "2" + "EN=9" + "Chassis serial
 number"
    host-identifier option dhcp6.client-id 00:02:00:00:00:09:46:4F:43:31:38:33:
31:58:31:41:53;
   option dhcp6.bootfile-url "http://[2001:DB8::461/platform-pxe/edi46/test-image.bin";
}
```

DHCP サーバ コマンドの詳細については、ISC DHCP サーバ の Web サイトを参照してください。

この設定例では、dhcp6.client-id オプションはスイッチを識別し、エンタープライズ クライアント DUID が続きます。クライアント DUID は、16 進形式の 00:02 + 00:00:00:09 + のシャーシ シリアル番号を理解するために分解できます。ここで 2 はエンタープライズ クライアント DUID タイプ、9 はシスコのエンタープライズ DUID の予約済みコードをそれぞれ参照し、16 進形式でのシャーシ シリアル番号の ASCII コードが続きます。このサンプルのスイッチのシャーシ シリアル番号は、FOC1831X1AS です。

ブート ファイル URI は、指定された DUID を使用してのみスイッチにアドバタイズされます。

DHCPv6 ベンダー クラス オプション 16 も、DHCP サーバ上のスイッチを識別するため使用できます。オプション 16 をユーザ定義オプションとして定義するには、次のように設定します。

```
option dhcp6.vendor-class-data code 16 = string;
```

次に示すのは、スイッチ製品 ID を使用して形成された DHCPv6 ベンダー クラス オプション 16 に基づいてスイッチを識別する、DHCP サーバの構成例です。

```
# Source: dhcp6ConfigPID

host edi-46-ipxe6-auto-edi46 {
    fixed-address6 2001:DB8::1234;
    host-identifier option dhcp6.client-id 00:02:00:00:00:09:
    46:4F:43:31:38:33:31:58:31:58:31:41:53;
    if option dhcp6.vendor-class-data = 00:00:00:09:00:0E:57:
    53:2D:43:33:38:35:30:2D:32:34:50:2D:4C {
      option dhcp6.bootfile-url "http://[2001:DB8::461/platform-pxe/edi46/17jan-dev.bin";

    }
}
```

この構成例では、dhcp6.vendor-class-data オプションは、DHCPv6 オプション 16 を参照します。dhcp6.vendor-class-data で、00:00:00:09 はシスコのエンタープライズ DUID、0E は PID の長さ、および残りは 16 進形式の PID です。PID は、**show inventory** コマンドまたは CFG_MODEL_NUM ROMmon 変数の出力から特定することもできます。このサンプル構成で使用される PID は、WS-C3850-24P-L です。

サーバ構成の DHCPv6 オプションおよび DUID は、ISC DHCP サーバのガイドラインに従って、16 進形式で指定する必要があります。

iPXE のトラブルシューティングのヒント

この項では、トラブルシューティングのヒントを説明します。

- 電源投入時に iPXE ブートが有効化されると、デバイスは、最初に DHCPv6 要請メッセージの送信を試行し、その後で、DHCPv4 検出メッセージの送信を試行します。ブート モードが **ipxe-forever** の場合、デバイスは、この 2 つを期限なしで反復し続けます。
- 起動モードが iPXE タイムアウトの場合、デバイスは、最初に DHCPv6 要請メッセージを、次に DHCPv4 検出メッセージを送信した後、タイムアウト時間が経過すると、デバイス ブートにフォールバックします。
- iPXE ブートを中断するには、コンソールにシリアル ブレイクを送信します。

  UNIX Telnet クライアントを使用している場合は、Ctrl キーを押した状態で] キーを押すと、ブレイクが送信されます。その他の Telnet クライアントを使用している場合、または

シリアルポートに直接接続している場合は、ブレイクの送信のトリガーは、別のキーストロークまたはコマンドの場合があります。

- DHCP サーバはイメージで応答するものの DNS サーバがホスト名を解決できない場合、DNS デバッグを有効にします。

（注） ISC の DHCP サーバの使用をお勧めします。IOS の DHCP ではこの機能はまだ検証されていません。

- HTTP サーバの接続をテストするには、HTTP コピーを使用して、HTTP サーバから少量のサンプル ファイルをデバイスにコピーします。たとえば ROMmon プロンプトで、**copy http://192.168.1.1/test null:**（フラッシュが通常はロックされており、テストに Null デバイスを使用する必要がある場合）または **http://[2001:db8::99]/test** と入力します。
- 手動ブートが有効化されており、ブート モードが iPXE タイムアウトである場合、デバイスが電源投入時に自動的に起動することはありません。ROMmon モードで **boot** コマンドを実行します。ブートプロセスが電源投入時に自動で発生するようにするには、手動ブートを無効にします。
- ROMmon モード の IPv6 アドレスやデフォルト ルータを含む現在の IPv6 パラメータを表示するには、**net6-show** コマンドを使用します。

（注） Catalyst 9000 シリーズ スイッチでは、 **net-show** show コマンドを使用します。

- 設定に基づいて、**net-dhcp** または **net6-dhcp** コマンドを使用します。**net-dhcp** コマンドは DHCPv4 用のテスト コマンド、**net6-dhcp** コマンドは DHCPv6 用のテスト コマンドです。

（注） Catalyst 9000 シリーズ スイッチでは、DHCPv6 に **net-dhcp -6** コマンドを使用します。

- 名前を解決するには、**dig** コマンドを使用します。

（注） Catalyst 9000 シリーズ スイッチでは、**dns-lookup** コマンドを使用して名前を解決します。

- Web サーバからの HTTP 応答コードを表示するには、HTTP デバッグ ログを有効にします。

- ステートレス アドレス自動設定（SLAAC）アドレスが生成されない場合、IPv6 RA メッセージを提供するルータがありません。この場合、IPv6 での iPXE ブートは、リンクローカルまたはサイトローカルのアドレスでのみ使用できます。

iPXE に関する追加情報

関連資料

関連項目	マニュアル タイトル
プログラマビリティ コマンド	『Programmability Command Reference, Cisco IOS XE Everest 16.6.1』

標準および RFC

標準/RFC	タイトル
RFC 3315	『*Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*』
RFC 3986	『*Uniform Resource Identifier (URI): Generic Syntax*』

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

iPXE 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。

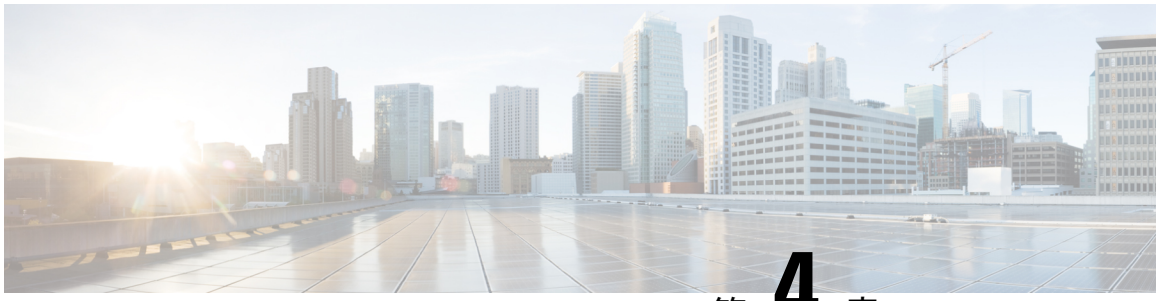
プラットフォームのサポートおよびシスコ ソフトウェア イメージのサポートに関する情報を検索するには、Cisco Feature Navigator を使用します。Cisco Feature Navigator にアクセスするには、www.cisco.com/go/cfn に移動します。Cisco.com のアカウントは必要ありません。

表 *4 : iPXE* の機能情報

機能名	リリース	機能情報
iPXE	Cisco IOS XE Denali 16.5.1a	ネットワーク ブートローダは、IPv4/IPv6 デバイス ベースまたはネットワーク ベースの送信元からのブート処理をサポートします。ネットワーク ブート ソースは、iPXE のようなソリューションを使用して自動的に検出される必要があります。 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。 • Catalyst 3650 シリーズ スイッチ • Catalyst 3850 シリーズ スイッチ
	Cisco IOS XE Denali 16.6.1	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。 • Catalyst 9300 シリーズ スイッチ • Catalyst 9500 シリーズ スイッチ
	Cisco IOS XE Everest 16.6.2	この機能は、Cisco IOS XE Everest 16.6.2 で、Cisco Catalyst 9400 シリーズ スイッチに実装されました。
	Cisco IOS XE Fuji 16.9.2	この機能は、Cisco IOS XE Everest 16.9.2 で、Cisco Catalyst 9200 シリーズ スイッチに実装されました。
IPXE IPv6 のサポート	Cisco IOS XE 16.8.1a	IPXE は IPv6 プロトコル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。 • Catalyst 9300 シリーズ スイッチ • Catalyst 9400 シリーズ スイッチ • Catalyst 9500 シリーズ スイッチ

第 II 部

シェルとスクリプト化



第 4 章

ゲスト シェル

ゲスト シェルは仮想化された Linux ベースの環境で、Python などの カスタム Linux アプリケーションを実行して Cisco デバイスを自動で制御および管理するために設計されています。システムの自動プロビジョニング（デイ ゼロ）も含まれます。このコンテナ シェルは、ホスト デバイスから分離された安全な環境を提供します。ユーザはそこで、スクリプトまたはソフトウェア パッケージをインストールし、実行することができます。

このモジュールでは、ゲスト シェルとそれを有効にする方法について説明します。

ゲスト シェルについて

ゲスト シェルの概要

ゲスト シェルは、仮想化された Linux ベースの環境であり、Cisco デバイスの自動制御と管理のための Python アプリケーションを含む、カスタム Linux アプリケーションを実行するように設計されています。ゲスト シェルを使用して、サードパーティ製 Linux アプリケーションをインストール、更新、および操作することもできます。ゲスト シェルはシステム イメージとともにバンドルされており、Cisco IOS コマンド **guestshell enable** を使用してインストールできます。

ゲスト シェル環境は、ネットワーキングではなく、ツール、Linux ユーティリティ、および管理性を意図したものです。

ゲスト シェルは、ホスト（Cisco スイッチおよびルータ）システムとカーネルを共有します。ユーザは、ゲスト シェルの Linux シェルにアクセスし、コンテナの rootfs にあるスクリプトおよびソフトウェア パッケージを更新することができます。ただし、ゲスト シェル内のユーザは、ホストのファイル システムおよびプロセスを変更することはできません。

ゲスト シェル コンテナは、IOx を使用して管理されます。IOx は、Cisco IOS XE デバイスのためのシスコのアプリケーション ホスティング インフラストラクチャです。IOx は、シスコ、パートナー、およびサードパーティの開発者によって開発されたアプリケーションおよびサービスをネットワーク エッジ デバイスでシームレスにホスティングすることを、各種の多様なハードウェア プラットフォームにおいて可能にします。

次の表は、ゲスト シェルのさまざまな機能とサポート対象のプラットフォームに関する情報を提供します。

表 *5 : Cisco* ゲスト シェルの機能

	ゲスト シェル Lite（限定的な LXC コンテナ）	ゲスト シェル（LXC コンテナ）
オペレーティング システム	Cisco IOS XE	Cisco IOS XE
サポートされるプラットフォーム	• Cisco Catalyst 3650 シリーズ スイッチ（全モデル） • Cisco Catalyst 3850 シリーズ スイッチ（全モデル）	• Cisco ISR 4000 シリーズ サービス統合型ルータ（最低 8 GB の RAM を有するモデル）
ゲスト シェル環境	Montavista CGE7	CentOS 7
Python 2.7	サポート対象（Python V2.7.11）	サポート対象（Python V2.7.5）
カスタムの Python ライブラリ	• Cisco 組込イベント マネージャ • Cisco IOS XE CLI • Ncclient	• Cisco 組込イベント マネージャ • Cisco IOS XE CLI
サポートされる rootfs	Busybox、SSH、および Python PIP のインストール	SSH、Yum のインストール、および Python PIP のインストール
GNU C コンパイラ	サポート対象外	サポート対象外
RPM のインストール	サポート対象外	サポートあり
アーキテクチャ	MIPS	x86

ゲスト シェルとゲスト シェル Lite

ゲストシェル コンテナを使用すると、ユーザは、システム上で自分のスクリプトやアプリケーションを実行できるようになります。Intel x86 プラットフォーム上のゲスト シェル コンテナは、CentOS 7.0 の最小限の rootfs を持つ Linux コンテナ（LXC）になります。ランタイム中に、CentOS 7.0 で Yum ユーティリティを使用して、Python バージョン 3.0 などの他の Python ライ

ブラリをインストールすることができます。また、PIP を使用して Python パッケージをインストールまたは更新することもできます。

Catalyst 3650 や Catalyst 3850 シリーズ スイッチなどの MIPS プラットフォーム上のゲスト シェル Lite コンテナには、Carrier Grade Edition（CGE）7.0 の rootfs があります。ゲスト シェル Lite では、スクリプトのインストールまたは実行のみ可能です。これらのデバイスでは、Yum のインストールはサポートされていません。

ゲスト シェルのセキュリティ

シスコは、ゲスト シェル内のユーザまたはアプリケーションによってホスト システムが攻撃されることがないよう、セキュリティを提供しています。ゲスト シェルは、ホスト カーネルから分離され、非特権コンテナとして動作します。

ゲスト シェルのハードウェア要件

この項では、サポート対象のプラットフォームにおけるハードウェア要件に関する情報を提供します。Cisco CSR 1000v と Cisco ISRv（仮想プラットフォーム）は、これらの要件をソフトウェアで実装します。

表 *6 : Catalyst* スイッチでのゲスト シェルのサポート

プラットフォーム	デフォルトの **DRAM**	ゲスト シェルのサポート
WS-3650-xxx（すべて）	4 GB	サポート対象
WS-3850-xxx（すべて）	4 GB	サポート対象
C9300-xx-x（すべて）	8 GB	サポート対象
C9500-24Q-x（すべて）	16 GB	サポート対象

Catalyst 3850 シリーズ スイッチの最小システム要件は、4 GB の DRAM です。

表 *7 : ISR 4000* シリーズ サービス統合型ルータでのゲスト シェルのサポート

プラットフォーム	デフォルトの **DRAM**	ゲスト シェルのサポート
ISR 4221	4GB	未サポート
ISR 4321	4 GB	未サポート
	8 GB	サポート対象
ISR 4331	8 GB	サポート対象
	16 GB	サポート対象
ISR 4351	8 GB	サポート対象
	16 GB	サポート対象

プラットフォーム	デフォルトの **DRAM**	ゲスト シェルのサポート
ISR 4431	8 GB	サポート対象
	16 GB	サポート対象
ISR 4451	8 GB	サポート対象
	16 GB	サポート対象

ISR 4000 シリーズ サービス統合型ルータの最小システム要件は、8 GB の DRAM です。

（注） 仮想サービスがインストールされているアプリケーションとゲスト シェル コンテナを同時に使用することはできません。

CSR 1000 v と ISRv の最小システム要件は、4 GB の RAM です。

ゲスト シェルのストレージ要件

Catalyst 3650 および Catalyst 3850 シリーズ スイッチでは、ゲスト シェルは、フラッシュのファイル システムにのみインストールできます。Catalyst 3850 シリーズ スイッチのブートフラッシュでは、ゲスト シェルを正常にインストールするには 75 MB のディスク空き容量が必要です。

Cisco 4000 シリーズ統合型サービス ルータでは、ゲスト シェルは、ネットワーク インターフェイス モジュール（NIM）のサービス セット識別子（SSID）（ハード ディスク）がある場合、そこにインストールされます。ハード ディスク ドライブが使用可能な場合、ゲスト シェルのインストールにブートフラッシュを選択することはできません。Cisco 4000 シリーズ サービス統合型ルータでは、ゲスト シェルを正常にインストールするには 1100 MB のハード ディスク（NIM SSID）空き容量が必要です。

Cisco 4000 シリーズ統合型サービス ルータおよび ASR 1000 ルータ（オプションのハード ディスクがそのルータに追加されている場合）では、ゲスト シェルをハードディスクにインストールしており、そのハードディスクがルータに挿入されている場合にのみリソースのサイズ変更を実行できます。

（注） ブートフラッシュを介してインストールしたゲスト シェルでは、アプリケーション ホスティング設定コマンドを使用したリソースのサイズ変更はできません。

ゲスト シェルのインストール中にハード ディスク容量が不足した場合、エラー メッセージが表示されます。

次に、ISR 4000 シリーズ ルータでのエラー メッセージの例を示します。

```
% Error:guestshell_setup.sh returned error:255, message:
```

```
Not enough storage for installing guestshell. Need 1100 MB free space.
```

ブートフラッシュまたはハード ディスクの空き領域は、ゲスト シェルが追加データを格納するために使用されることがあります。Cisco Catalyst 3850 シリーズ スイッチでは、ゲスト シェルが使用できるストレージ 容量は 18 MB です。Cisco 4000 シリーズ サービス統合型ルータでは、ゲスト シェルが使用できるストレージ容量は 800 MB です。ゲスト シェルはブートフラッシュにアクセスするため、その空き領域の全体を使用できます。

表 *8 :* ゲスト シェルおよびゲスト シェル *Lite* が使用できるリソース

リソース	デフォルト	最小/最大
CPU	1 % （注） 1 % は非標準。800 CPU ユニット/システム CPU ユニットの全体	1/100 %
メモリ	256 MB 512 MB（Cisco CSR 1000v）	256/256 MB 512/512 MB（Cisco CSR 1000v）

デバイスでのゲスト シェルへのアクセス

ネットワーク管理者は、IOS コマンドを使用して、ゲスト シェル内のファイルおよびユーティリティを管理することができます。

ゲスト シェルのインストール中に、SSH アクセスがキー ベースの認証でセットアップされます。ゲスト シェルへのアクセスは、IOS の最も高い特権（15）を持つユーザに制限されます。このユーザは、sudo の実行者である guestshell Linux ユーザとして Linux コンテナへのアクセスを許可され、すべてのルート操作を実行できます。ゲスト シェルから実行されるコマンドは、ユーザが IOS 端末にログインしたときと同じ特権で実行されます。

ゲスト シェル プロンプトでは、標準的な Linux コマンドを実行できます。

管理ポートを介してのゲスト シェルへのアクセス

ゲスト シェルは、デフォルトで、アプリケーションによる管理ネットワークへのアクセスを許可します。ユーザは、ゲスト シェル内から管理 VRF のネットワーキング設定を変更することはできません。

（注） 管理ポートがないプラットフォームの場合、VirtualPortGroup を IOS 設定内のゲスト シェルに関連付けることができます。詳細については、「VirtualPortGroup の設定例」の項を参照してください。

ゲスト シェルでのスタッキング

ゲスト シェルがインストールされている場合、フラッシュのファイルシステムには、gs_script ディレクトリが自動的に作成されます。このディレクトリは、スタック メンバー間で同期されます。切り替え時には、gs_scriptディレクトリの内容のみが、すべてのスタック メンバー間で同期されます。ハイ アベイラビリティでの切り替えの際にデータを保持するには、このディレクトリにデータを格納します。

ハイ アベイラビリティでの切り替えの際には、新しいアクティブ デバイスは、それぞれのゲスト シェル インストールを作成します。古いファイル システムは維持されません。ゲスト シェルの状態は、切り替え時に維持されます。

IOx の概要

IOx は Cisco が開発したエンド ツー エンド アプリケーション フレームワークであり、Cisco ネットワーク プラットフォーム上のさまざまなタイプのアプリケーションに対し、アプリケーション ホスティング機能を提供します。Cisco ゲスト シェルは特殊なコンテナ展開であり、システムの開発および使用に役立つアプリケーションの 1 つです。

IOxは、構築済みアプリケーションをパッケージ化し、それらをターゲットデバイス上にホストする開発者の作業を支援する一連のサービスを提供することにより、アプリケーションのライフサイクル管理とデータ交換を容易化します。IOxのライフサイクル管理には、アプリケーションおよびデータの配布、展開、ホスティング、開始、停止（管理）、およびモニタが含まれます。IOx サービスにはアプリケーションの配布および管理ツールも含まれており、ユーザがアプリケーションを発見して IOx フレームワークに展開するのに役立ちます。

アプリケーション ホスティングは、次の機能を提供します。

- ネットワークの不均質性の遮蔽。
- デバイス上にホストされているアプリケーションのライフサイクルをリモートで管理する IOx アプリケーション プログラミング インターフェイス（API）。
- 一元的なアプリケーション ライフ サイクル管理。
- クラウド ベースの開発。

IOx のトレースとロギングの概要

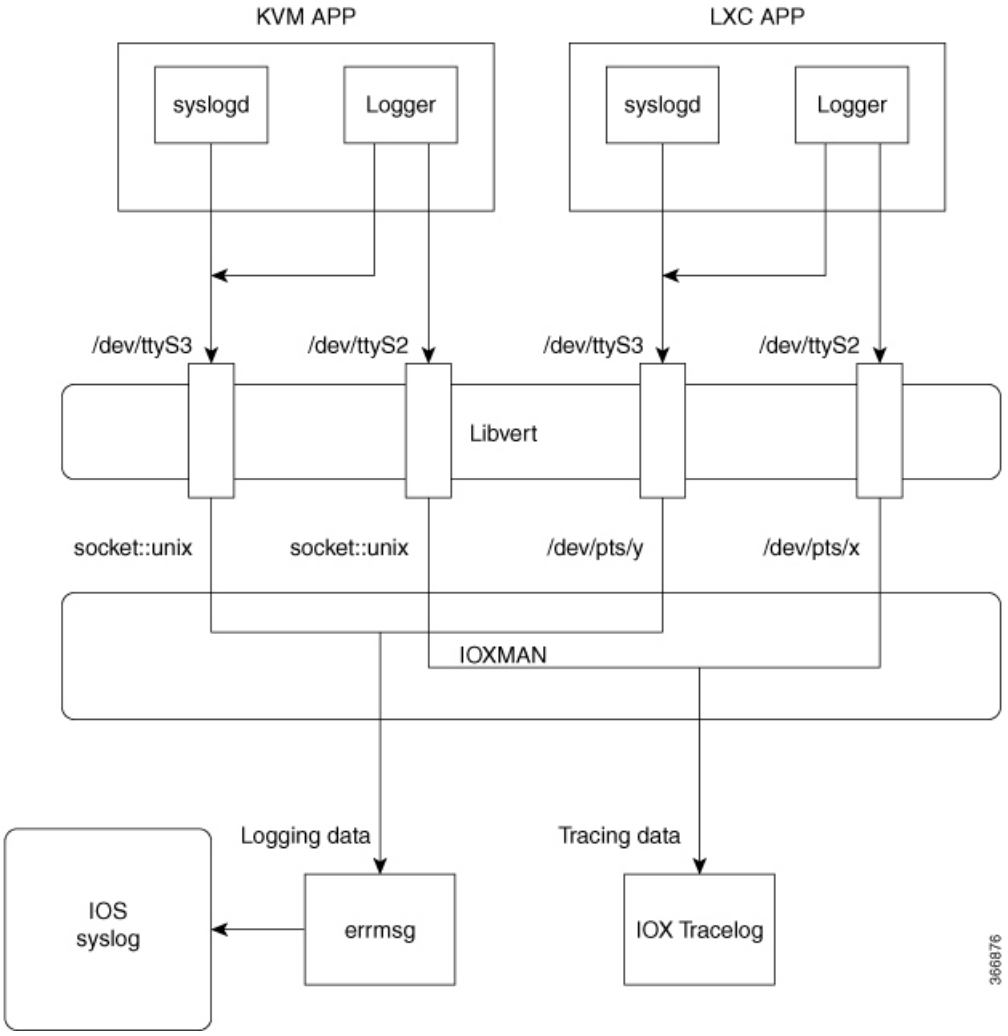
IOx のトレースとロギングの機能を使用すると、ホスト デバイスでゲスト アプリケーションを個別に実行できます。これにより、ホストへのデータのロギングとトレースをレポートするのに役立ちます。トレース データは IOx トレースログに保存され、ロギング データはホスト デバイスの IOS syslog に保存されます。

トレース データをホスト デバイス上の適切なストレージ デバイスにリダイレクトすると、ゲスト アプリケーションのデバッグに役立ちます。

IOXMAN 構造体

ゲスト アプリケーション、システム LXC、または KVM インスタンスはそれぞれ独自の syslog およびログファイルを使用して設定されます。これらのファイルは表示可能なファイル システム内に保存され、ホスト デバイスからはアクセスできません。IOS syslog へのデータのロギングとホスト上の IOx トレース ログへのデータのトレースをサポートするため、次の図に示すように、ホストにデータを配信するための 2 つのシリアル デバイス（**/dev/ttyS2** と **/dev/ttyS3**）がゲスト アプリケーションで指定されています。

図 *2 : IOXMAN* 構造体



IOXMAN は、トレース インフラストラクチャを確立してロギング サービスまたはトレース サービス（シリアル デバイスをエミュレートする Libvert を除く）を提供するプロセスです。IOXMAN は、ゲスト アプリケーションのライフサイクルに基づいて、トレース サービスを有効または無効にし、ロギング データを IOS syslog に送信し、トレース データを IOx トレース ログに保存し、各ゲスト アプリケーションの IOx トレースロ グを維持します。

ロギングとトレースのシステム フロー

ここでは、IOx のロギングとトレースの仕組みについて説明します。

LXC のロギング

1. ゲスト OS が、ゲスト アプリケーションで **/dev/ttyS2** を有効にします。
2. ゲスト アプリケーションが、**/dev/ttyS2** にデータを書き込みます。
3. Libvert が、ホストで **/dev/pts/x** への **/dev/ttyS2** をエミュレートします。
4. IOXMAN が、エミュレートされたシリアル デバイス **/dev/pts/x** を XML ファイルから取得します。
5. IOXMAN が、使用可能なデータを **/dev/pts/x** からリッスンして読み取り、メッセージの重大度を設定して、メッセージをフィルタ処理し、解析してキューに格納します。
6. **errmsg** を使用してホストの **/dev/log** デバイスにメッセージを送信するタイマーが開始されます。
7. データが IOS syslog に保存されます。

KVM のロギング

1. ゲスト OS が、ゲスト アプリケーションで **/dev/ttyS2** を有効にします。
2. ゲスト アプリケーションが、**/dev/ttyS2** にデータを書き込みます。
3. Libvert が、ホストで **/dev/pts/x** への **/dev/ttyS2** をエミュレートします。
4. IOXMAN が、エミュレートされた TCP パスを XML ファイルから取得します。
5. IOXMAN が、UNIX ソケットを開き、リモート ソケットに接続します。
6. IOXMAN が、使用可能なデータをソケットから読み取り、メッセージの重大度を設定して、メッセージをフィルタ処理し、解析して、キューに格納します。
7. **errmsg** を使用してホストの **/dev/log** デバイスにメッセージを送信するタイマーが開始されます。
8. データが IOS syslog に保存されます。

LXC のトレース

1. ゲスト OS が、ゲスト アプリケーションで **/dev/ttyS3** を有効にします。
2. メッセージを **/dev/ttyS3** にコピーするように **syslogd** を設定します。
3. ゲスト アプリケーションが、**/dev/ttyS3** にデータを書き込みます。
4. Libvert が、ホストで **/dev/pts/y** への **/dev/ttyS3** をエミュレートします。

5. IOXMAN が、エミュレートされたシリアル デバイス **/dev/pts/y** を XML ファイルから取得します。
6. IOXMAN が、使用可能なデータを **/dev/pts/y** からリッスンして読み取り、フィルタ処理し、解析して、メッセージを IOx トレースログに保存します。
7. IOx トレースログが満杯の場合は、IOXMAN がトレースログ ファイルを **/bootflash/tracelogs** にローテーションします。

KVM のトレース

1. ゲスト OS が、ゲスト アプリケーションで **/dev/ttyS3** を有効にします。
2. メッセージを **/dev/ttyS3** にコピーするように syslogd を設定します。
3. ゲスト アプリケーションが、**/dev/ttyS3** にデータを書き込みます。
4. Libvert が、ホストで TCP パスへの **/dev/ttyS3** をエミュレートします。
5. IOXMAN が、エミュレートされた TCP パスを XML ファイルから取得します。
6. IOXMAN が、UNIX ソケットを開き、リモート ソケットに接続します。
7. IOXMAN が、使用可能なデータをソケットから読み取り、メッセージの重大度を設定して、メッセージをフィルタ処理し、解析して、IOx トレースログに格納します。
8. IOx トレースログが満杯の場合は、IOXMAN がトレースログ ファイルを **/bootflash/tracelogs** にローテーションします。

メッセージのロギングとトレース

ここでは、IOS syslog へのメッセージのロギングとトレースについて説明します。

IOS syslog でのメッセージのロギング

ゲスト アプリケーションから受信したどのロギング メッセージでも、IOXMAN はメッセージの重大度をデフォルトで NOTICE に設定してから IOS syslog に送信します。IOSd で受信されたメッセージはコンソールに表示され、次のメッセージ形式で IOS syslog に保存されます。

***Apr 7 00:48:21.911: %IM-5-IOX_INST_NOTICE:ioxman: IOX SERVICE guestshell LOG: Guestshell test**

IOS syslog に準拠するために、IOXMAN はロギング メッセージの重大度をサポートしています。重大度のあるロギング メッセージを報告するには、ゲスト アプリケーションでメッセージの先頭にヘッダーを追加する必要があります。

```
[a123b234,version,severity]

a123b234 is magic number.
Version:        severity support version.  Current version is 1.
Severity:       CRIT is 2
                ERR is 3
                WARN is 4
                NOTICE is 5
```

```
INFO is 6
DEBUG is 7
```

次に、メッセージ ログの例を示します。

```
echo “[a123b234,1,2]Guestshell failed” > /dev/ttyS2
```

ゲスト アプリケーションから IOS syslog にロギング データを報告するには、次の手順を実行します。

1. Cプログラミングを使用している場合は、**write()** を使用してロギング データをホストに送信します。

```
#define SYSLOG_TEST       “syslog test”
     int fd;
     fd = open("/dev/ttyS2", O_WRONLY);
     write(fd, SYSLOG_TEST, strlen(SYSLOG_TEST));
     close(fd);
```

2. シェル コンソールを使用している場合は、**echo** を使用してロギング データをホストに送信します。

```
echo “syslog test” > /dev/ttyS2
```

IOx トレースログへのメッセージのトレース

ゲスト アプリケーションから IOx トレースログにトレース メッセージを報告するには、次の手順を実行します。

1. Cプログラミングを使用している場合は、**write()** を使用してトレース メッセージをホストに送信します。

```
#define SYSLOG_TEST       “tracelog test”
     int fd;
     fd = open("/dev/ttyS3", O_WRONLY);
     write(fd, SYSLOG_TEST, strlen(SYSLOG_TEST));
     close(fd);
```

2. C プログラミングを使用している場合は、**syslog()** を使用してトレース メッセージをホストに送信します。

```
#define SYSLOG_TEST       “tracelog test”

syslog(LOG_INFO, “%s\n”, SYSLOG_TEST);
```

3. シェル コンソールを使用している場合は、**echo** を使用してトレース データをホストに送信します。

```
echo “tracelog test” > /dev/ttyS3
   or
logger “tracelog test”
```

例：ゲスト シェルのネットワーキング設定

ゲスト シェルのネットワーキングでは、次の設定が必要です。

- ドメイン ネーム システム（DNS）の設定
- プロキシの設定
- プロキシの設定を使用するための YUM または PIP の設定

ゲスト シェルを有効にする方法

IOx の管理

始める前に

IOx は開始まで最長で2分かかります。ゲスト シェルを正常に有効にするには、CAF、IOXman、および Libirtd サービスが実行している必要があります。

手順の概要

1. **enable**
2. **configure terminal**
3. **iox**
4. **exit**
5. **show iox-service**
6. **show app-hosting list**

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）。
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	**iox** 例： `Device(config)# iox`	IOx サービスを設定します。

	コマンドまたはアクション	目的
ステップ 4	**exit** 例： `Device(config)# exit`	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 5	**show iox-service** 例： `Device# show iox-service`	IOx サービスのステータスを表示します。
ステップ 6	**show app-hosting list** 例： `Device# show app-hosting list`	デバイスに対して有効になっている app-hosting サービスのリストを表示します。

次のタスク

次に、ISR 4000 シリーズ ルータでの **show iox-service** コマンドの出力例を示します。

```
Device# show iox-service

Virtual Service Global State and Virtualization Limits:

Infrastructure version : 1.7
Total virtual services installed : 0
Total virtual services activated : 0

Machine types supported   : KVM, LXC
Machine types disabled    : none

Maximum VCPUs per virtual service : 6
Resource virtualization limits:
Name                         Quota     Committed     Available
------------------------------------------------------------------
system CPU (%)                  75             0            75
memory (MB)                  10240             0         10240
bootflash (MB)                1000             0          1000
harddisk (MB)                20000             0         18109
volume-group (MB)           190768             0        170288


IOx Infrastructure Summary:
---------------------------
IOx service (CAF)    : Running
IOx service (HA)     : Not Running
IOx service (IOxman) : Running
Libvirtd             : Running
```

次に示すのは、Catalyst 3850 シリーズ スイッチでの **show iox-service** コマンドの短縮された出力例です。

```
Device# show iox-service

IOx Infrastructure Summary:
---------------------------
IOx service (CAF)    : Running
```

```
IOx service (HA)     : Running
IOx service (IOxman) : Running
Libvirtd             : Running
```

次に、**show app-hosting list** コマンドの出力例を示します。

```
Device# show app-hosting list

App id                                   State
---------------------------------------------------------
guestshell                               RUNNING
```

ゲスト シェルの管理

始める前に

ゲスト シェルのアクセスが機能するには、IOx が構成されて実行している必要があります。IOx が構成されていない場合は、IOx の構成を求めるメッセージが表示されます。IOx を削除すると、ゲスト シェルにもアクセスできなくなります。ただし rootfs は影響を受けません。

手順の概要

1. **enable**
2. • **guestshell enable**
   • **guestshell enable** [**VirtualPortGroup** *port-number* **guest-ip** *ip-address* **gateway** *gateway-ip* **netmask** *netmask* [**name-server** *ip-address*]]
3. **guestshell run** *linux-executable*
4. **guestshell run bash**
5. **guestshell disable**
6. **guestshell destroy**

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）。
ステップ 2	• **guestshell enable** • **guestshell enable** [**VirtualPortGroup** *port-number* **guest-ip** *ip-address* **gateway** *gateway-ip* **netmask** *netmask* [**name-server** *ip-address*]] 例：	ゲスト シェル サービスの有効化。 または フロント パネル ポートへの接続を有効にします。

	コマンドまたはアクション	目的

	`Device# guestshell enable`
	例：
	`Device# guestshell enable VirtualPortGroup 0 guest-ip 192.168.35.2 gateway 192.168.35.1 netmask 255.255.255.0 name-server 10.1.1.1`
		（注）
		• 引数を指定しない **guestshell enable** コマンドは、ネットワーキングに管理 Virtual Routing and Forwarding（VRF）インスタンスを使用します。
		• **guestshell enable** コマンドと引数は、Cisco IOS XE 16.6.x 以前でのみサポートされます。
		• フロント パネル ネットワーキングに VirtualPortGroups（VPG）を使用している場合は、まず VPG を構成する必要があります。
		• ゲスト IP アドレスとゲートウェイ IP アドレスは同じサブネット内にある必要があります。
		• 前面パネルのネットワーキングは、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 9500 シリーズ スイッチではサポートされていません。これは、**guestshell enable** コマンドと引数は入力できますが、これらのプラットフォーム上では NAT を設定できず、ネットワーキングが動作しないためです。管理モードだけがサポートされています。

ステップ 3	**guestshell run** *linux-executable* 例： `Device# guestshell run python`	ゲスト シェルで Linux プログラムを実行します。 • Python バージョン 2.7.11 は Catalyst 3650 および Catalyst 3850 シリーズ スイッチにプリインストールされており、Python バージョン 2.7.5 は ISR 4000 シリーズ ルータにプリインストールされています。
ステップ 4	**guestshell run bash** 例： `Device# guestshell run bash`	Bash シェルを開始して、ゲスト シェルにアクセスします。
ステップ 5	**guestshell disable** 例：	ゲスト シェル サービスを無効化します。

	コマンドまたはアクション	目的
	`Device# guestshell disable`	
ステップ 6	**guestshell destroy** 例： `Device# guestshell destroy`	ゲスト シェル サービスを非アクティブ化して、アンインストールします。

ゲスト シェルの有効化と実行

guestshell enable コマンドは、ゲスト シェルをインストールします。このコマンドは、無効化されているゲスト シェルを再アクティブ化する際にも使用されます。

ゲスト シェルが有効化された状態でシステムをリロードすると、ゲスト シェルは有効化されたままになります。

（注） **guestshell enable** コマンドを使用する前に、IOx を設定しておく必要があります。

guestshell run bash コマンドは、ゲスト シェルの bash プロンプトを開きます。このコマンドを動作させるには、ゲスト シェルが事前に有効化されていることが必要です。

（注） 次のメッセージがコンソールに表示される場合、IOx が有効化されていません。「**show iox-service** コマンドの出力をチェックして、IOx の状態を確認してください」

```
The process for the command is not responding or is otherwise unavailable
```

ゲスト シェルの無効化と破棄

guestshell disable コマンドを使用することで、ゲスト シェルを終了して無効化できます。ゲスト シェルが無効化された状態でシステムをリロードすると、ゲスト シェルは無効化されたままになります。

guestshell destroy コマンドは、フラッシュのファイル システムから rootfs を削除します。すべてのファイル、データ、インストールされている Linux アプリケーション、およびカスタムの Python ツールとユーティリティが削除され、回復できなくなります。

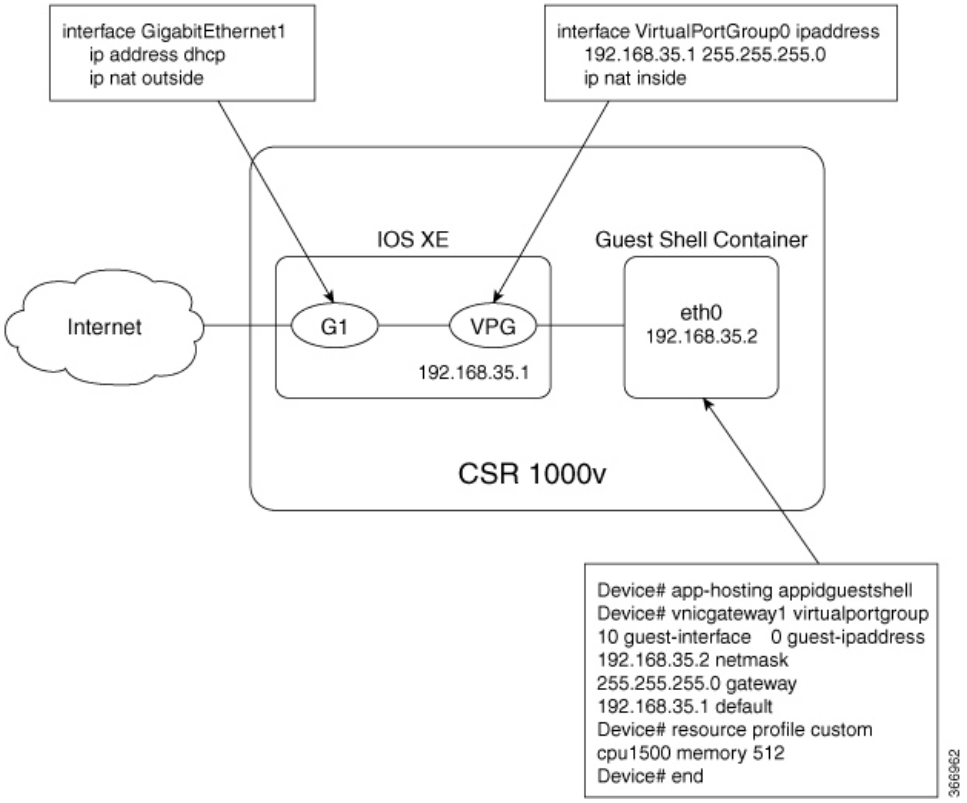
アプリケーション ホスティングを使用したゲスト シェルの管理

（注） ゲスト シェルのアクセスが機能するには、IOx が構成されて実行している必要があります。IOx が構成されていない場合は、IOx の構成を求めるメッセージが表示されます。IOx を削除すると、ゲスト シェルにもアクセスできなくなります。ただし rootfs は影響を受けません。

（注） この手順（アプリケーションホスティングを使用したゲストシェルの管理）を使用して、Cisco IOS XE Fuji 16.7.1 以降のゲスト シェルを有効にします。Cisco IOS XE Everest 16.6.x 以前では、ゲスト シェルの管理 （65 ページ）の手順を使用します。

```
Router(config)# interface GigabitEthernet1
Router(config-if)# ip address dhcp
Router(config-if)# ip nat outside
Router(config-if)# exit

Router(config-if)# interface VirtualPortGroup0
Router(config-if)# ip address 192.168.35.1 255.255.255.0
Router(config-if)# ip nat inside
Router(config-if)# exit

Router(config)# ip nat inside source list GS_NAT_ACL interface GigabitEthernet1 overload
Router(config)# ip access-list standard GS_NAT_ACL
Router(config)# permit 192.168.0.0 0.0.255.255

Router(config)# app-hosting appid guestshell
Router(config-app-hosting)# vnic gateway1 virtualportgroup 0 guest-interface 0
guest-ipaddress 192.168.35.2 netmask 255.255.255.0 gateway 192.168.35.1 name-server
8.8.8.8 default
Router(config-app-hosting)# resource profile custom cpu 1500 memory 512
Router(config-app-hosting)# end

Router# guestshell enable
Router# guestshell run python
```

図 *3:* アプリケーション ホスティングを使用したゲスト シェルの管理



前面パネルのネットワーキングでは、GigabitEthernet インターフェイスと VirtualPortGroup インターフェイスを上の図に示すように設定する必要があります。ゲスト シェルは virtualportgroup を送信元インターフェイスとして使用し、NAT を通じて外部ネットワークに接続します。

内部 NAT の設定には、次のコマンドを使用します。これにより、ゲスト シェルがインターネットに到達し、たとえば、Linux ソフトウェア更新プログラムを取得できるようになります。

```
ip nat inside source list
ip access-list standard
permit
```

上の例の `guestshell run` コマンドは Python 実行可能ファイルを実行します。また、`guestshell run` コマンドを使用して他の Linux 実行可能ファイルを実行することもできます。たとえば、`guestshell run bash` コマンドは bash シェルを起動し、`guestshell disable` コマンドはゲスト シェルをシャットダウンして無効にします。後でシステムをリロードしても、ゲスト シェルは無効のままになります。

Python インタープリタのアクセス

Python はインタラクティブに使用できますが、Python スクリプトをゲスト シェルで実行することもできます。**guestshell run python** コマンドを使用してゲスト シェルで Python インタープリタを起動し、Python 端末を開きます。

（注） **guestshell run** コマンドは、Linux 実行可能ファイルの実行に相当する IOS であり、IOS からの Python スクリプトの実行時に絶対パスを指定します。次の例は、コマンドの絶対パスを指定する方法を示しています。

```
Guestshell run python /flash/sample_script.py parameter1 parameter2
```

ゲスト シェルの設定例

例：ゲスト シェルの管理

次の例では、Catalyst 3850 シリーズ スイッチ上でゲスト シェルを有効にする方法を示しています。

```
Device> enable
Device# guestshell enable

Management Interface will be selected if configured
Please wait for completion
Guestshell enabled successfully

Device# guestshell run python

Python 2.7.11 (default, Feb 21 2017, 03:39:40)
[GCC 5.3.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.

Device# guestshell run bash

[guestshell@guestshell ~]$

Device# guestshell disable

Guestshell disabled successfully

Device# guestshell destroy

Guestshell destroyed successfully
```

VirtualPortGroup 設定の例

ゲスト シェル ネットワーキングに VirtualPortGroup インターフェイスを使用する場合、VirtualPortGroup インターフェイスには設定済みの静的 IP アドレスが必要です。フロントポートインターフェイスはインターネットに接続されている必要があり、ネット

ワーク アドレス変換（NAT）は VirtualPortGroup とフロント パネル ポートの間で設定されている必要があります。

次に示すのは、VirtualPortGroup の設定例です。

```
Device> enable
Device# configure terminal
Device(config)# interface VirtualPortGroup 0
Device(config-if)# ip address 192.168.35.1 255.255.255.0
Device(config-if)# ip nat inside
Device(config-if)# no mop enabled
Device(config-if)# no mop sysid
Device(config-if)# exit
Device(config)# interface GigabitEthernet 0/0/3
Device(config-if)# ip address 10.0.12.19 255.255.0.0
Device(config-if)# ip nat outside
Device(config-if)# negotiation auto
Device(config-if)# exit
Device(config)# ip route 0.0.0.0 0.0.0.0 10.0.0.1
Device(config)# ip route 10.0.0.0 255.0.0.0 10.0.0.1
!Port forwarding to use ports for SSH and so on.
Device(config)# ip nat inside source static tcp 192.168.35.2 7023 10.0.12.19 7023
extendable
Device(config)# ip nat outside source list NAT_ACL interface GigabitEthernet 0/0/3
overload
Device(config)# ip access-list standard NAT_ACL
Device(config-std-nacl)# permit 192.168.0.0 0.0.255.255
Device(config-std-nacl)# exit
Device(config)# exit
Device#
```

例：ゲスト シェルの使用

ゲスト シェル プロンプトから Linux のコマンドを実行できます。次の例は、一部の Linux コマンドの使用法を示しています。

```
[guestshell@guestshell~]$ pwd
/home/guestshell

[guestshell@guestshell~]$ whoami
guestshell

[guestshell@guestshell~]$ uname -a
Linux guestshell 3.10.101.cge-rt110 #1 SMP Sat Feb 11 00:33:02
PST 2017 mips64 GNU/Linux
```

Catalyst 3650 および Catalyst 3850 シリーズ スイッチには、BusyBox が提供する定義された一連の Linux 実行可能ファイルがあり、Cisco 4000 シリーズ サービス統合型ルータには、CentOS Linux リリース 7.1.1503 が提供するコマンドがあります。

次の例は、Catalyst 3850 シリーズ スイッチ上での **dohost** コマンドの使用を示しています。

```
[guestshell@guestshell ~]$ dohost "show version"

Cisco IOS Software [Everest], Catalyst L3 Switch Software [CAT3K_CAA-UNIVERSALK9-M),
Experimental Version 16.5.2017200014[v165_throttle-BLD-
 BLD_V165_THROTTLE_LATEST_20170531_192849 132]
```

（注） **dohost** コマンドには、**ip http server** コマンドがデバイス上で設定されていることが必要です。

例：ゲスト シェルのネットワーキング設定

ゲスト シェルのネットワーキングでは、次の設定が必要です。

- ドメイン ネーム システム（DNS）の設定
- プロキシの設定
- プロキシの設定を使用するための YUM または PIP の設定

ゲスト シェルの DNS 設定の例

ゲスト シェルのサンプル DNS 構成は次のとおりです。

```
[guestshell@guestshell ~]$ cat/etc/resolv.conf
nameserver 192.0.2.1

Other Options:
[guestshell@guestshell ~]$ cat/etc/resolv.conf
domain cisco.com
search cisco.com
nameserver 192.0.2.1
search cisco.com
nameserver 198.51.100.1
nameserver 172.16.0.6
domain cisco.com
nameserver 192.0.2.1
nameserver 172.16.0.6
nameserver 192.168.255.254
```

例：プロキシ環境変数の設定

ネットワークがプロキシの背後にある場合は、Linux でプロキシ変数を設定します。必要な場合は、環境にこれらの変数を追加します。

次の例は、プロキシ変数を設定する方法を示しています。

```
[guestshell@guestshell ~]$cat /bootflash/proxy_vars.sh
export http_proxy=http://proxy.example.com:80/
export https_proxy=http://proxy.example.com:80/
export ftp_proxy=http://proxy.example.com:80/
export no_proxy=example.com
export HTTP_PROXY=http://proxy.example.com:80/
export HTTPS_PROXY=http://proxy.example.com:80/
export FTP_PROXY=http://proxy.example.com:80/
guestshell ~] source /bootflash/proxy_vars.sh
```

例：プロキシ設定用の Yum および PIP の構成

次の例は、プロキシ環境変数の設定に Yum を使用する方法を示しています。

```
cat /etc/yum.conf | grep proxy
[guestshell@guestshell~]$ cat/bootflash/yum.conf | grep proxy
proxy=http://proxy.example.com:80/
```

PIP のインストールでは、プロキシ設定に使用される環境変数が選択されます。PIP インストールには -E オプションを指定した sudo を使用します。環境変数が設定されていない場合は、次の例に示すように PIP コマンドでそれらを明示的に定義します。

```
sudo pip --proxy http://proxy.example.com:80/install requests
sudo pip install --trusted-bost pypi.example.com --index-url
http://pypi.example.com/simple requests
```

次の例では、Python の PIP インストールを使用する方法を示します。

```
Sudo -E pip install requests
[guestshell@guestshell ~]$ python
Python 2.17.11 (default, Feb 3 2017, 19:43:44)
[GCC 4.7.0] on linux2
Type "help", "copyright", "credits" or "license" for more information
>>>import requests
```

ゲスト シェルに関するその他の参考資料

関連資料

関連項目	マニュアル タイトル
	『Programmability Command Reference, Cisco IOS XE Everest 16.6.1』
Python モジュール	『CLI Python モジュール』
ゼロ タッチ プロビジョニング	『ゼロ タッチ プロビジョニング』

MIB

MB	MIB のリンク
	選択したプラットフォーム、Cisco IOS リリース、およびフィーチャ セットに関する MIB を探してダウンロードするには、次の URL にある Cisco MIB Locator を使用します。 http://www.cisco.com/go/mibs

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

ゲスト シェルの機能情報

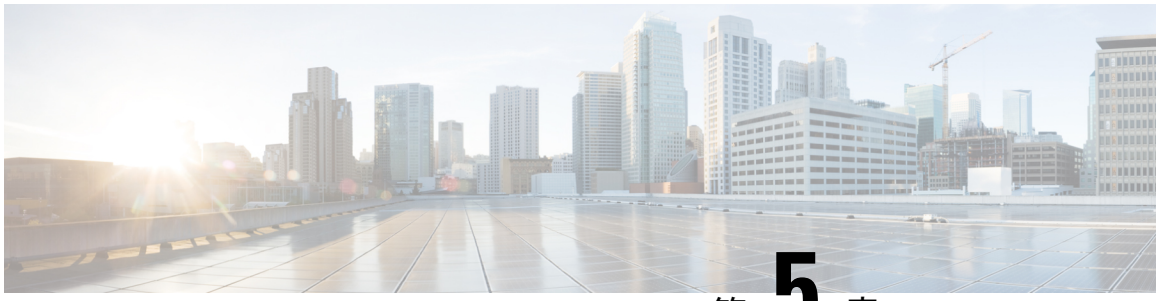
次の表に、このモジュールで説明した機能に関するリリース情報を示します。この表は、ソフトウェア リリース トレインで各機能のサポートが導入されたときのソフトウェア リリースだけを示しています。その機能は、特に断りがない限り、それ以降の一連のソフトウェア リリースでもサポートされます。

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 *9:* ゲスト シェルの機能情報

機能名	リリース	機能情報
ゲスト シェル	Cisco IOS XE Everest 16.5.1a Cisco IOS XE Everest 16.5.1b	ゲスト シェルは、お客様がシスコ スイッチの自動制御および管理のためのカスタム Python アプリケーションを実行できる、埋め込み Linux 環境であるセキュア コンテナです。システムの自動化されたプロビジョニングも含まれます。このコンテナ シェルは、ホスト デバイスから分離された安全な環境を提供します。ユーザはそこで、スクリプトまたはソフトウェア パッケージをインストールし、実行することができます。 Cisco IOS XE Everest 16.5.1a 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 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 9500 シリーズ スイッチ Cisco IOS Everest 16.5.1b 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 4000 シリーズ サービス統合型ルータ
	Cisco IOS XE Everest 16.6.2	この機能は、Cisco IOS XE Everest 16.6.2 で、Cisco Catalyst 9400 シリーズ スイッチに実装されました。

機能名	リリース	機能情報
	Cisco IOS XE Fuji 16.7.1	この機能は、Cisco IOS XE Fuji 16.7.1 で、Cisco CSR 1000v シリーズに実装されました。 Cisco IOS XE Fuji 16.7.1 では、ゲスト シェル機能の場合、ロギングとトレーシング サポートが Cisco ASR 1000 アグリゲーション サービス ルータに実装されました。
	Cisco IOS XE Fuji 16.8.1a	Cisco IOS XE Fuji 16.8.1a では、この機能は Cisco Catalyst 9500 ハイ パフォーマンス シリーズ スイッチに実装されていました。



第 5 章

Python API

Python プログラマビリティは、Python API をサポートしています。

• Python の使用 （79 ページ）

Python の使用

Cisco Python モジュール

シスコが提供する Python モジュールでは、EXEC および設定コマンドを実行するアクセス権が提供されます。 **help()** コマンドを入力すると、Cisco Python モジュールの詳細が表示されます。 **help()** コマンドは Cisco CLI モジュールのプロパティを表示します。

次の例は、Cisco Python モジュールに関する情報を示します。

```
Device# guestshell run python

Python 2.7.5 (default, Jun 17 2014, 18:11:42)
[GCC 4.8.2 20140120 (Red Hat 4.8.2-16)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> >>> from cli import cli,clip,configure,configurep, execute, executep
>>> help(configure)
Help on function configure in module cli:

configure(configuration)
Apply a configuration (set of Cisco IOS CLI config-mode commands) to the device
and return a list of results.

configuration = '''interface gigabitEthernet 0/0
no shutdown'''

# push it through the Cisco IOS CLI.
try:
results = cli.configure(configuration)
print "Success!"
except CLIConfigurationError as e:
print "Failed configurations:"
for failure in e.failed:
print failure
```

```
Args:
configuration (str or iterable): Configuration commands, separated by newlines.

Returns:
list(ConfigResult): A list of results, one for each line.

Raises:
CLISyntaxError: If there is a syntax error in the configuration.
```

```
>>> help(configurep)
Help on function configurep in module cli:

configurep(configuration)
Apply a configuration (set of Cisco IOS CLI config-mode commands) to the device
and prints the result.

configuration = '''interface gigabitEthernet 0/0
no shutdown'''

# push it through the Cisco IOS CLI.
configurep(configuration)

Args:
configuration (str or iterable): Configuration commands, separated by newlines.
```

```
>>> help(execute)
Help on function execute in module cli:

execute(command)
Execute Cisco IOS CLI exec-mode command and return the result.

command_output = execute("show version")

Args:
command (str): The exec-mode command to run.

Returns:
str: The output of the command.

Raises:
CLISyntaxError: If there is a syntax error in the command.
```

```
>>> help(executep)
Help on function executep in module cli:

executep(command)
Execute Cisco IOS CLI exec-mode command and print the result.

executep("show version")

Args:
command (str): The exec-mode command to run.
```

```
>>> help(cli)
Help on function cli in module cli:

cli(command)
    Execute Cisco IOS CLI command(s) and return the result.

    A single command or a delimited batch of commands may be run. The
    delimiter is a space and a semicolon, " ;". Configuration commands must be
    in fully qualified form.
```

```
    output = cli("show version")
    output = cli("show version ; show ip interface brief")
    output = cli("configure terminal ; interface gigabitEthernet 0/0 ; no shutdown")

    Args:
        command (str): The exec or config CLI command(s) to be run.

    Returns:
        string: CLI output for show commands and an empty string for
            configuration commands.

    Raises:
        errors.cli_syntax_error: if the command is not valid.
        errors.cli_exec_error: if the execution of command is not successful.


>>> help(clip)
Help on function clip in module cli:

clip(command)
    Execute Cisco IOS CLI command(s) and print the result.

    A single command or a delimited batch of commands may be run. The
    delimiter is a space and a semicolon, " ;". Configuration commands must be
    in fully qualified form.

    clip("show version")
    clip("show version ; show ip interface brief")
    clip("configure terminal ; interface gigabitEthernet 0/0 ; no shutdown")

    Args:
        command (str): The exec or config CLI command(s) to be run.
```

IOS CLI コマンドを実行するための Cisco Python モジュール

（注） Python を実行するには、ゲスト シェルが有効である必要があります。詳細については、「ゲスト シェル」の章を参照してください。

Python プログラミング言語は CLI コマンドを実行できる 6 つの関数を使用します。これらの関数は、Python CLI モジュールから利用できます。これらの関数を使用するには、**import cli** コマンドを実行します。これらの関数が機能するには、**ip http server** コマンドが有効になっている必要があります。

これらの関数の引数は CLI コマンドの文字列です。Python インタープリタ経由で CLI コマンドを実行するには、次の 6 つの関数のいずれかの引数文字列として CLI コマンドを入力します。

- **cli.cli(command)**：この関数は IOS コマンドを引数として取り、IOS パーサーからコマンドを実行し、結果のテキストを返します。このコマンドの形式が正しくない場合、Python の例外が発生します。次に、**cli.cli(command)** 関数の出力例を示します。

```
>>> import cli
>>> cli.clip('configure terminal; interface loopback 10; ip address
```

```
 10.10.10.10 255.255.255.255')
*Mar 13 18:39:48.518: %LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback10,
changed state to up
>>> cli.clip('show clock')
'\n*18:11:53.989 UTC Mon Mar 13 2017\n'
>>> output=cli.cli('show clock')
>>> print(output)
*18:12:04.705 UTC Mon Mar 13 2017
```

- **cli.clip(command)**：この関数は **cli.cli(command)** 関数と機能はまったく同じです。ただし結果のテキストを（返すのではなく）*stdout* に出力する点が異なります。次に、**cli.clip(command)** 関数の出力例を示します。

```
>>> cli
>>> cli.clip('configure terminal; interface loopback 11; ip address
 10.11.11.11 255.255.255.255')
*Mar 13 18:42:35.954: %LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback11,
changed state to up
*Mar 13 18:42:35.954: %LINK-3-UPDOWN: Interface Loopback11, changed state to up
>>> cli.clip('show clock')
*18:13:35.313 UTC Mon Mar 13 2017
>>> output=cli.clip('show clock')
*18:19:26.824 UTC Mon Mar 13 2017
>>> print (output)
None
```

- **cli.execute(command)**：この関数は単一の EXEC コマンドを実行して出力を返します。ただし結果のテキストは出力しません。このコマンドの一部としてセミコロンまたは改行を使用することは許可されません。この関数を複数回実行するには、for-loop が指定された Python リストを使用します。次に、 **cli.execute(command)**

  関数の出力例を示します。

```
>>> cli.execute("show clock")
'15:11:20.816 UTC Thu Jun 8 2017'
>>>
>>> cli.execute('show clock'; 'show ip interface brief')
  File "<stdin>", line 1
    cli.execute('show clock'; 'show ip interface brief')
                            ^
SyntaxError: invalid syntax
>>>
```

- **cli.executep(command)**：この関数は単一のコマンドを実行して、結果のテキストを（返すのではなく）stdout に出力します。次に、**cli.executep(command)** 関数の出力例を示します。

```
>>> cli.executep('show clock')
*18:46:28.796 UTC Mon Mar 13 2017
>>> output=cli.executep('show clock')
*18:46:36.399 UTC Mon Mar 13 2017
>>> print(output)
```

```
None
```

- **cli.configure(command)**：この関数は、コマンドで使用できる設定によりデバイスを設定します。これは次に示すように、コマンドとその結果が含まれる名前付きタプルのリストを返します。

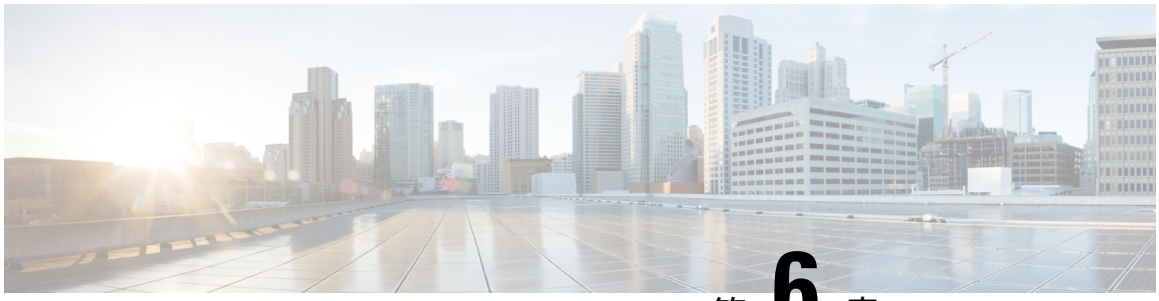
```
[Think: result = (bool(success), original_command, error_information)]
```

コマンド パラメータは複数行に入力することができ、**show running-config** コマンドの出力に表示されているのと同じ形式にすることができます。次に、**cli.configure(command)** 関数の出力例を示します。

```
>>>cli.configure(["interface GigabitEthernet1/0/7", "no shutdown",
"end"])
[ConfigResult(success=True, command='interface GigabitEthernet1/0/7',
line=1, output='', notes=None), ConfigResult(success=True, command='no shutdown',
line=2, output='', notes=None), ConfigResult(success=True, command='end',
line=3, output='', notes=None)]
```

- **cli.configurep(command)**：この関数は **cli.configure(command)** 関数と機能はまったく同じです。ただし結果のテキストを（返すのではなく）stdout に出力する点が異なります。次に、**cli.configurep(command)** 関数の出力例を示します。

```
>>> cli.configurep(["interface GigabitEthernet1/0/7", "no shutdown",
 "end"])
Line 1 SUCCESS: interface GigabitEthernet1/0/7
Line 2 SUCCESS: no shut
Line 3 SUCCESS: end
```



第 6 章

CLI Python モジュール

Python プログラマビリティでは、CLI を使用して IOS と対話できる Python モジュールを提供しています。

Python CLI モジュールについて

Python について

Cisco IOS XE デバイスは、ゲスト シェル内でインタラクティブおよび非インタラクティブ（スクリプト）の両方のモードで Python バージョン 2.7 をサポートします。Python スクリプト機能により、デバイスの CLI にプログラムを使用してアクセスして、さまざまなタスク、およびゼロ タッチ プロビジョニングまたは Embedded Event Manager（EEM）アクションを実行することができます。

Python スクリプトの概要

Python は、仮想化された Linux ベースの環境であるゲスト シェルで実行されます。詳細については、「ゲスト シェル」の章を参照してください。シスコが提供する Python モジュールは、ユーザの Python スクリプトがホスト デバイス上で IOS CLI コマンドを実行することを可能にします。

対話形式の Python プロンプト

デバイス上で **guestshell run python** コマンドを実行すると、ゲスト シェル内で、対話形式の Python プロンプトが開きます。Python の対話モードでは、Cisco Python CLI モジュールから Python 機能を実行してデバイスを設定することができます。

次の例は、対話形式の Python プロンプトを有効にする方法を示しています。

```
Device# guestshell run python

Python 2.7.5 (default, Jun 17 2014, 18:11:42)
[GCC 4.8.2 20140120 (Red Hat 4.8.2-16)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>

Device#
```

Python スクリプト

Python スクリプト名を引数として Python コマンドで使用することで、Python スクリプトを非インタラクティブ モードで実行できます。Python スクリプトは、ゲスト シェル内からアクセス可能である必要があります。ゲスト シェルから Python スクリプトにアクセスするには、ゲスト シェル内にマウントされているブートフラッシュまたはフラッシュにスクリプトを保存します。

次の Python スクリプトの例は、さまざまな CLI 関数を使用して **show** コマンドを設定および出力します。

```
Device# more flash:sample_script.py


import sys
import cli

intf= sys.argv[1:]
intf = ''.join(intf[0])

print "\n\n *** Configuring interface %s with 'configurep' function  *** \n\n" %intf
cli.configurep(["interface loopback55","ip address 10.55.55.55 255.255.255.0","no
shut","end"])

print "\n\n *** Configuring interface %s with 'configure' function  *** \n\n"
cmd='interface %s,logging event link-status ,end' % intf
cli.configure(cmd.split(','))

print "\n\n *** Printing show cmd with 'executep' function  *** \n\n"
cli.executep('show ip interface brief')

print "\n\n *** Printing show cmd with 'execute' function  *** \n\n"
output= cli.execute('show run interface %s' %intf)
print (output)

print "\n\n *** Configuring interface %s with 'cli' function  *** \n\n"
cli.cli('config terminal; interface %s; spanning-tree portfast edge default' %intf)

print "\n\n *** Printing show cmd with 'clip' function  *** \n\n"
cli.clip('show run interface %s' %intf)


To run a Python script from the Guest Shell, execute the guestshell run python
/flash/script.py command
at the device prompt.
The following example shows how to run a Python script from the Guest Shell:
```

次の例は、ゲスト シェルから Python スクリプトを実行する方法を示しています。

```
Device# guestshell run python /flash/sample_script.py loop55

 *** Configuring interface loop55 with 'configurep' function  ***

Line 1 SUCCESS: interface loopback55
Line 2 SUCCESS: ip address 10.55.55.55 255.255.255.0
Line 3 SUCCESS: no shut
Line 4 SUCCESS: end

 *** Configuring interface %s with 'configure' function  ***


 *** Printing show cmd with 'executep' function  ***

Interface              IP-Address      OK? Method Status                Protocol
Vlan1                  unassigned      YES NVRAM  administratively down down
GigabitEthernet0/0     192.0.2.1       YES NVRAM  up                    up
GigabitEthernet1/0/1   unassigned      YES unset  down                  down
GigabitEthernet1/0/2   unassigned      YES unset  down                  down
GigabitEthernet1/0/3   unassigned      YES unset  down                  down
    :
         :
         :
Te1/1/4                unassigned      YES unset  down                  down
Loopback55             10.55.55.55     YES TFTP   up                    up
Loopback66             unassigned      YES manual up                    up


 *** Printing show cmd with 'execute' function  ***

Building configuration...
Current configuration : 93 bytes
!
interface Loopback55
 ip address 10.55.55.55 255.255.255.0
 logging event link-status
end

 *** Configuring interface %s with 'cli' function  ***

 *** Printing show cmd with 'clip' function  ***

Building configuration...
Current configuration : 93 bytes
!
interface Loopback55
 ip address 10.55.55.55 255.255.255.0
 logging event link-status
end
```

サポートされる Python のバージョン

ゲスト シェルは、Python バージョン 2.7 をプリインストールしています。ゲスト シェルは、仮想化された Linux ベースの環境であり、Cisco デバイスの自動制御と管理のための Python アプリケーションを含む、カスタム Linux アプリケーションを実行するように設計されています。Montavista CGE7 がインストールされたプラットフォームは Python バージョン 2.7.11 をサポートし、CentOS 7 がインストールされたプラットフォームは Python バージョン 2.7.5 をサポートします。

次の表は、Pythonの各バージョンおよびサポート対象のプラットフォームに関する情報を示しています。

表 *10 : Python* バージョン サポート

Python のバージョン	プラットフォーム

CentOS 7がインストールされたプラットフォームは、オープン ソース リポジトリからのRedhat Package Manager（RPM）のインストールをサポートします。

Cisco CLI Python モジュールの更新

Cisco CLI Pythonモジュールおよび EEMモジュールは、デバイスにインストール済みです。ただし、Yumまたは事前にパッケージ化されているバイナリのいずれかを使用してPythonのバージョンを更新する場合は、シスコが提供する CLI モジュールも更新する必要があります。

（注） Python バージョン 2 がすでにあるデバイスで Python バージョン 3 への更新を行うと、デバイス上には両方のバージョンの Python が存在するようになります。Python を実行するには、次の IOS コマンドのいずれかを使用します。

- **guestshell run python2** コマンドは、Python バージョン 2 を有効化します。
- **guestshell run python3** コマンドは、Python バージョン 3 を有効化します。
- **guestshell run python** コマンドは、Python バージョン 2 を有効化します。

Python のバージョンを更新するには、次の方法のいずれかを使用します。

- スタンドアロン tarball のインストール
- CLI モジュールのための PIP のインストール

CLI Python モジュールに関するその他の参考資料

関連資料

関連項目	マニュアル タイトル
ゲスト シェル	ゲスト シェル
EEM Python モジュール	EEM の Python スクリプト

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

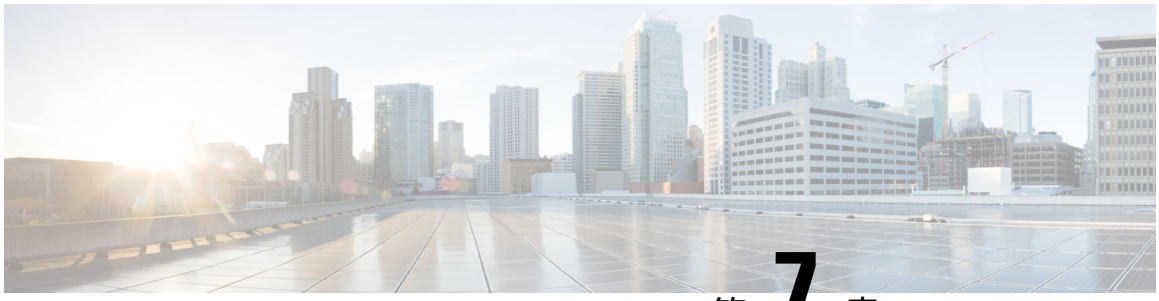
CLI Python モジュールの機能情報

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 *11 : CLI Python* モジュールの機能情報

機能名	リリース	機能情報
CLI Python モジュール	Cisco IOS XE Everest 16.5.1a	Python プログラマビリティでは、ユーザが CLI を使用して IOS と対話できるようにする Python モジュールが提供されます。 Cisco IOS XE Everest 16.5.1a 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 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 9500 シリーズ スイッチ Cisco IOS XE Everest 16.5.1b 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 4000 シリーズ サービス統合型ルータ
	Cisco IOS XE Everest 16.6.2	この機能は、Cisco Catalyst 9400 シリーズ スイッチに実装されました。
	Cisco IOS XE Fuji 16.7.1	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 ASR 1000 アグリゲーション サービス ルータ • Cisco CSR 1000v シリーズ クラウド サービス ルータ



第 7 章

EEM Python モジュール

組み込みイベント マネージャ（EEM）ポリシーは、Python スクリプトをサポートします。Python スクリプトは、EEM アプレットで EEM アクションの一部として実行できます。

EEM Python モジュールの前提条件

ゲスト シェルは、コンテナ内で機能する必要があります。ゲスト シェルは、デフォルトでは有効になっていません。詳細については、ゲスト シェル機能の説明を参照してください。

EEM Python モジュールについて

EEM の Python スクリプト

組み込みイベント マネージャ（EEM）ポリシーは、Python スクリプトをサポートします。Python スクリプトを EEM ポリシーとして登録し、対応するイベントが発生したときに、登録済みの Python スクリプトを実行することができます。EEM Python スクリプトには、EEM TCL ポリシーと同じイベント仕様の構文があります。

設定済みの EEM ポリシーは、ゲスト シェル内で実行します。ゲスト シェルは、仮想化された Linux ベースの環境であり、Cisco デバイスの自動制御と管理のための Python アプリケーションを含む、カスタム Linux アプリケーションを実行するように設計されています。ゲスト シェル コンテナは、Python インタープリタを提供します。

EEM Python パッケージ

EEM Python パッケージを Python スクリプトにインポートすると、EEM に固有の拡張機能を実行できます。

（注） EEM Python パッケージは、EEM Python スクリプト内でのみ使用できます（パッケージは EEM に登録でき、スクリプトの最初の行に EEM イベント仕様が記載されます）。標準的な Python スクリプト（Python スクリプト名を使用して実行される）では使用できません。

Python パッケージには、次のアプリケーションプログラミングインターフェイス（API）が含まれています。

- アクション API：EEM アクションを実行するもので、デフォルトのパラメータがあります。
- CLI 実行 API：IOS コマンドを実行し、出力を返します。次に、CLI 実行 API のリストを示します。
  - eem_cli_open()
  - eem_cli_exec()
  - eem_cli_read()
  - eem_cli_read_line()
  - eem_cli_run()
  - eem_cli_run_interactive()
  - eem_cli_read_pattern()
  - eem_cli_write()
  - eem_cli_close()
- 環境変数にアクセスする API：組み込みまたはユーザ定義の変数のリストを取得します。次に、環境変数にアクセスする API を示します。
  - eem_event_reqinfo ()：組み込み変数のリストを返します。
  - eem_user_variables()：引数の現在の値を返します。

Python がサポートする EEM アクション

Python パッケージ（EEM スクリプト内でのみ使用可能で、標準的な Python スクリプトでは使用不可）では、次の EEM アクションをサポートしています。

- Syslog メッセージの印刷
- SNMP トラップの送信

- ボックスのリロード
- スタンバイ デバイスへの切り替え
- ポリシーの実行
- トラック オブジェクトの読み取り
- トラック オブジェクト セット
- Cisco ネットワーキング サービスのイベントの生成

EEM Python パッケージは、EEM アクションを実行するため、インターフェイスを公開します。これらのアクションは Python スクリプトを使用して呼び出すことができ、Cisco Plug N Play（PnP）経由で Python パッケージからアクション ハンドラに転送されます。

EEM 変数

EEM ポリシーは、次の種類の変数を持つことができます。

- イベント固有の組み込み変数：ポリシーをトリガーしたイベントの詳細が設定される事前定義の変数のセット。eem_event_reqinfo () API は、組み込み変数のリストを返します。これらの変数は、ローカルマシンに保存してローカル変数として使用することができます。ローカル変数に対する変更は、組み込み変数に反映されません。
- ユーザ定義の変数：定義およびポリシーでの使用が可能な変数。これらの変数の値は、Python スクリプト内で参照できます。スクリプトを実行する際に、変数の最新の値が使用可能であることを確認してください。eem_user_variables() API は、API で入力された引数の現在の値を返します。

EEM CLI ライブラリのコマンド拡張

EEM 内では、Python スクリプトを動作させるため、次の CLI ライブラリ コマンドを使用できます。

- eem_cli_close()：EXEC プロセスをクローズし、コマンドに接続された、VTY および指定されたチャネル ハンドラをリリースします。
- eem_cli_exec：指定されたチャネル ハンドラにコマンドを記述し、コマンドを実行します。次に、チャネルからコマンドの出力を読み取り、出力を返します。
- eem_cli_open：VTY を割り当て、EXEC CLI セッションを作成し、VTY をチャネル ハンドラに接続します。チャネル ハンドラを含む配列を返します。
- eem_cli_read()：読み取られている内容でデバイス プロンプトのパターンが発生するまで、指定された CLI のチャネル ハンドラからコマンド出力を読み取ります。一致するまで、読み取られたすべての内容を返します。
- eem_cli_read_line()：指定された CLI のチャネル ハンドラから、コマンド出力の 1 行を読み取ります。読み取られた行を返します。

- eem_cli_read_pattern()：読み取られている内容でパターンが発生するまで、指定された CLI のチャネル ハンドラからコマンド出力を読み取ります。一致するまで、読み取られたすべての内容を返します。
- eem_cli_run()：clist にある項目を繰り返し、それぞれが、イネーブル モードで実行されるコマンドであることを前提とします。正常に実行されると、実行されたすべてのコマンドの出力を返します。失敗すると、エラーを返します。
- eem_cli_run_interactive()：3 つの項目がある clist のサブリストを用意します。正常に実行されると、実行されたすべてのコマンドの出力を返します。失敗すると、エラーを返します。可能な場合には、配列も使用します。予測と応答を別々に保持することによって、より簡単に後で読み取ることができます。
- eem_cli_write()：指定された CLI チャネル ハンドラに対して実行されるコマンドを書き込みます。CLI チャネル ハンドラによって、コマンドが実行されます。

EEM Python ポリシーの設定方法

Python スクリプトが動作できるようにするには、ゲスト シェルを有効化する必要があります。詳細については、「ゲスト シェル」の章を参照してください。

Python ポリシーの登録

手順の概要

1. **enable**
2. **configure terminal**
3. **event manager directory user policy** *path*
4. **event manager policy** *policy-filename*
5. **exit**
6. **show event manager policy registered**
7. **show event manager history events**

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）。
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。

	コマンドまたはアクション	目的
ステップ 3	**event manager directory user policy** *path* 例： `Device(config)# event manager directory user policy flash:/user_library`	ユーザ ライブラリ ファイルまたはユーザ定義 EEM ポリシーの保存に使用するディレクトリを指定します。 （注） 指定されたパスにポリシーが必要です。たとえば、この手順では、eem_script.py ポリシーが flash:/user_library フォルダーまたはパスで使用できます。
ステップ 4	**event manager policy** *policy-filename* 例： `Device(config)# event manager policy eem_script.py`	EEM ポリシーを EEM に登録します。 • ポリシーは、ファイル拡張子に基づいて解析されます。ファイル拡張子は .py で、ポリシーは Python ポリシーとして登録されます。 • EEMは、ポリシーそのものに含まれるイベント仕様に基づいてポリシーをスケジューリングし、実行します。**event manager policy** コマンドが呼び出されると、EEMはポリシーを確認し、指定されたイベントが発生した場合に実行されるように登録します。
ステップ 5	**exit** 例： `Device(config)# exit`	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 6	**show event manager policy registered** 例： `Device# show event manager policy registered`	保留 EEM ポリシーを表示します。
ステップ 7	**show event manager history events** 例： `Device# show event manager history events`	トリガーされた EEM イベントを表示します。

例

次に、**show event manager policy registered** コマンドの出力例を示します。

```
Device# show event manager policy registered

No.  Class     Type    Event Type          Trap  Time Registered           Name
1    script    user    multiple            Off   Tue Aug 2 22:12:15 2016   multi_1.py
 1: syslog: pattern {COUNTER}
 2: none: policyname {multi_1.py} sync {yes}
 trigger delay 10.000
  correlate event 1 or event 2
  attribute tag 1 occurs 1
```

```
 nice 0 queue-priority normal maxrun 100.000 scheduler rp_primary Secu none


2    script    user    multiple            Off   Tue Aug 2 22:12:20 2016   multi_2.py
 1: syslog: pattern {COUNTER}
 2: none: policyname {multi_2.py} sync {yes}
 trigger
  correlate event 1 or event 2
 nice 0 queue-priority normal maxrun 100.000 scheduler rp_primary Secu none


3    script    user    multiple            Off   Tue Aug 2 22:13:31 2016   multi.tcl
 1: syslog: pattern {COUNTER}
 2: none: policyname {multi.tcl} sync {yes}
 trigger
  correlate event 1 or event 2
  attribute tag 1 occurs 1
 nice 0 queue-priority normal maxrun 100.000 scheduler rp_primary Secu none
```

EEM アプレット アクションの一部としての Python スクリプトの実行

Python スクリプト：eem_script.py

アクション コマンドを使用することで、EEM アプレットに Python スクリプトを含めることができます。この例では、ユーザは標準 Python スクリプトを EEM アクションの一部として実行しようとしています。ただし、EEM Python パッケージは標準 Python スクリプトでは使用できません。IOS の標準 Python スクリプトには from cli import cli,clip という名前のパッケージがあり、そのパッケージは IOS コマンドを実行するために使用できます。

```
import sys
from cli import cli,clip,execute,executep,configure,configurep

intf= sys.argv[1:]
intf = ''.join(intf[0])


print ('This script is going to unshut interface %s and then print show ip interface
brief'%intf)

if intf == 'loopback55':
configurep(["interface loopback55","no shutdown","end"])
else :
cmd='int %s,no shut ,end' % intf
configurep(cmd.split(','))

executep('show ip interface brief')
```

次に、**guestshell run python** コマンドの出力例を示します。

```
Device#  guestshell run python /flash/eem_script.py loop55

This script is going to unshut interface loop55 and then print show ip interface brief
Line 1 SUCCESS: int loop55
Line 2 SUCCESS: no shut
```

```
Line 3 SUCCESS: end
Interface IP-Address OK? Method Status Protocol
Vlan1 unassigned YES NVRAM administratively down down
GigabitEthernet0/0 5.30.15.37 YES NVRAM up up
GigabitEthernet1/0/1 unassigned YES unset down down
GigabitEthernet1/0/2 unassigned YES unset down down
GigabitEthernet1/0/3 unassigned YES unset down down
GigabitEthernet1/0/4 unassigned YES unset up up
GigabitEthernet1/0/5 unassigned YES unset down down
GigabitEthernet1/0/6 unassigned YES unset down down
GigabitEthernet1/0/7 unassigned YES unset down down
GigabitEthernet1/0/8 unassigned YES unset down down
GigabitEthernet1/0/9 unassigned YES unset down down
GigabitEthernet1/0/10 unassigned YES unset down down
GigabitEthernet1/0/11 unassigned YES unset down down
GigabitEthernet1/0/12 unassigned YES unset down down
GigabitEthernet1/0/13 unassigned YES unset down down
GigabitEthernet1/0/14 unassigned YES unset down down
GigabitEthernet1/0/15 unassigned YES unset down down
GigabitEthernet1/0/16 unassigned YES unset down down
GigabitEthernet1/0/17 unassigned YES unset down down
GigabitEthernet1/0/18 unassigned YES unset down down
GigabitEthernet1/0/19 unassigned YES unset down down
GigabitEthernet1/0/20 unassigned YES unset down down
GigabitEthernet1/0/21 unassigned YES unset down down
GigabitEthernet1/0/22 unassigned YES unset down down
GigabitEthernet1/0/23 unassigned YES unset up up
GigabitEthernet1/0/24 unassigned YES unset down down
GigabitEthernet1/1/1 unassigned YES unset down down
GigabitEthernet1/1/2 unassigned YES unset down down
GigabitEthernet1/1/3 unassigned YES unset down down
GigabitEthernet1/1/4 unassigned YES unset down down
Te1/1/1 unassigned YES unset down down
Te1/1/2 unassigned YES unset down down
Te1/1/3 unassigned YES unset down down
Te1/1/4 unassigned YES unset down down
Loopback55 10.55.55.55 YES manual up up

Device#
Jun 7 12:51:20.549: %LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback55,
changed state to up
Jun 7 12:51:20.549: %LINK-3-UPDOWN: Interface Loopback55, changed state to up
```

次に示すのは、syslog へのメッセージ出力のサンプル スクリプトです。このスクリプトは、ファイルに保存され、デバイス上のファイル システムにコピーされ、イベント マネージャのポリシー ファイルを使用して登録される必要があります。

```
::cisco::eem::event_register_syslog tag "1"    pattern     COUNTER    maxrun 200

import eem
import time

eem.action_syslog("SAMPLE SYSLOG MESSAGE","6","TEST")
```

次に示すのは、EEM 環境変数を出力するサンプル スクリプトです。このスクリプトは、ファイルに保存され、デバイス上のファイル システムにコピーされ、イベント マネージャのポリシー ファイルを使用して登録される必要があります。

```
::cisco::eem::event_register_syslog tag "1"    pattern     COUNTER    maxrun 200
```

```
import eem
import time

c = eem.env_reqinfo()


print "EEM Environment Variables"
for k,v in c.iteritems():
      print "KEY : " + k + str(" ---> ") + v

print "Built in Variables"
for i,j in a.iteritems() :
      print "KEY : " + i + str(" ---> ") + j
```

EEM アプレットでの Python スクリプトの追加

手順の概要

1. **enable**
2. **configure terminal**
3. **event manager applet** *applet-name*
4. **event** [**tag** *event-tag*] **syslog pattern** *regular-expression*
5. **action** *label* **cli command** *cli-string*
6. **action** *label* **cli command** *cli-string* [**pattern** *pattern-string*]
7. **end**
8. **show event manager policy active**
9. **show event manager history events**

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）。
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	**event manager applet** *applet-name* 例： `Device(config)# event manager applet interface_Shutdown`	Embedded Event Manager（EEM）にアプレットを登録し、アプレット コンフィギュレーション モードを開始します。
ステップ 4	**event** [**tag** *event-tag*] **syslog pattern** *regular-expression* 例：	syslog メッセージのパターン一致を実行する正規表現を指定します。

	コマンドまたはアクション	目的
	`Device(config-applet)# event syslog pattern "Interface Loopback55, changed state to administratively down"`	
ステップ 5	**action** *label* **cli command** *cli-string* 例： `Device(config-applet)# action 0.0 cli command "en"`	EEMアプレットがトリガーされたときに実行されるIOS コマンドを指定します。
ステップ 6	**action** *label* **cli command** *cli-string* [**pattern** *pattern-string*] 例： `Device(config-applet)# action 1.0 cli command "guestshell run python3 /bootflash/eem_script.py loop55"`	**pattern** キーワードで指定されるアクションを指定します。 • 次の要請プロンプトに一致する正規表現パターン文字列を指定します。
ステップ 7	**end** 例： `Device(config-applet)# end`	アプレット 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 8	**show event manager policy active** 例： `Device# show event manager policy active`	実行している EEM ポリシーを表示します。
ステップ 9	**show event manager history events** 例： `Device# show event manager history events`	トリガーされた EEM イベントを表示します。

次のタスク

次の例では、タスクに設定されているPythonスクリプトをトリガーする方法を示しています。

```
Device(config)# interface loopback 55
Device(config-if)# shutdown
Device(config-if)# end
Device#

Mar 13 10:53:22.358 EDT: %SYS-5-CONFIG_I: Configured from console by console
Mar 13 10:53:24.156 EDT: %LINK-5-CHANGED:  Line protocol on Interface Loopback55, changed
 state to down
Mar 13 10:53:27.319 EDT: %LINK-3-UPDOWN:  Interface Loopback55, changed state to
administratively down
Enter configuration commands, one per line.  End with CNTL/Z.
Mar 13 10:53:35.38 EDT:  %LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback55,
changed state to up
*Mar 13 10:53:35.39 EDT %LINK-3-UPDOWN: Interface Loopback55, changed state to up
+++ 10:54:33 edi37(default) exec +++
show ip interface br
Interface              IP-Address      OK? Method Status                Protocol
GigabitEthernet0/0/0   unassigned      YES unset  down                  down
GigabitEthernet0/0/1   unassigned      YES unset  down                  down
GigabitEthernet0/0/2   10.1.1.31       YES DHCP   up                    up
```

```
GigabitEthernet0/0/3   unassigned      YES unset  down                  down
GigabitEthernet0       192.0.2.1       YES manual up                    up
Loopback55             198.51.100.1    YES manual up                    up
Loopback66             172.16.0.1      YES manual up                    up
Loopback77             192.168.0.1     YES manual up                    up
Loopback88             203.0.113.1     YES manual up                    up
```

EEM Python モジュールに関するその他の参考資料

関連資料

関連項目	マニュアル タイトル
Cisco IOS コマンド	『*Cisco IOS Master Command List, All Releases*』
EEM 設定	『*Embedded Event Manager Configuration Guide*』
EEM コマンド	『*Embedded Event Manager Command Reference*』
ゲスト シェル設定	『ゲスト シェル』

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

EEM Python モジュールの機能情報

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。

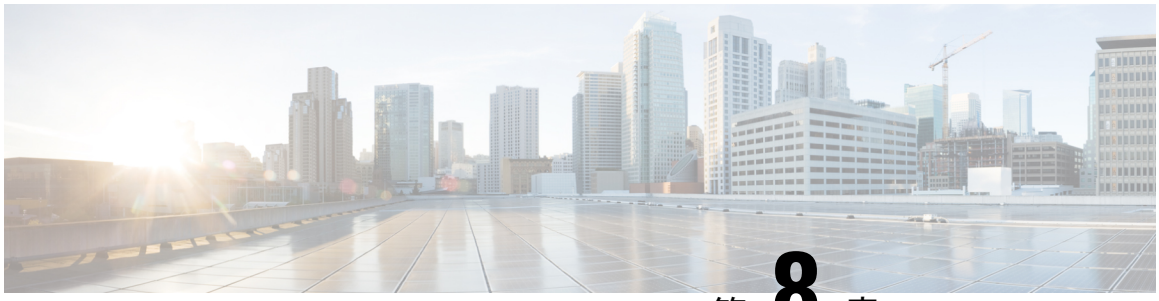
プラットフォームのサポートおよびシスコ ソフトウェア イメージのサポートに関する情報を検索するには、Cisco Feature Navigator を使用します。Cisco Feature Navigator にアクセスするには、www.cisco.com/go/cfn に移動します。Cisco.com のアカウントは必要ありません。

表 *12 : EEM Python* モジュールの機能情報

機能名	リリース	機能情報
EEM Python モジュール	Cisco IOS XE Everest 16.5.1a Cisco IOS XE Everest 16.5.1b	この機能は、EEM ポリシーとして Python スクリプトをサポートします。 追加された新規コマンドはありません。 Cisco IOS XE Everest 16.5.1a 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 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 IOS XE Everest 16.5.1b 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 ISR 4000 シリーズ サービス統合型ルータ
	Cisco IOS XE Everest 16.6.2	この機能は、Cisco IOS XE Everest 16.6.2 で、Cisco Catalyst 9400 シリーズ スイッチに実装されました。
	Cisco IOS XE Fuji 16.8.1a	Cisco IOS XE Fuji 16.8.1a では、この機能は Cisco Catalyst 9500 ハイ パフォーマンス シリーズ スイッチに実装されていました。

第 III 部

モデル駆動型プログラマビリティ



第 8 章

NETCONF プロトコル

NETCONF プロトコルの概要

データモデルの概要：プログラムによる設定と各種の標準規格に準拠した設定

ネットワーク デバイスを管理する従来の方法は、階層的データ（設定コマンド）および運用データ（showコマンド）用のコマンドラインインターフェイス（CLI）を使用することです。ネットワーク管理の場合、特にさまざまなネットワーク デバイス間で管理情報を交換するために、Simple Network Management Protocol（SNMP）が広く使用されています。頻繁に使用されているCLIとSNMPですが、これにはいくつかの制約事項があります。CLIは非常に独自的であり、テキストベースの仕様を理解し、解釈するには人間の介入が必要です。SNMPは、階層的データと運用データを区別しません。

これを解決するには、手作業で設定作業を行うのではなく、プログラムを使用したり、各種の標準規格に準拠してネットワーク デバイスの設定を記述します。Cisco IOS XEで動作するネットワーク デバイスは、データ モデルを使用するネットワーク上の複数のデバイスの設定の自動化をサポートしています。データ モデルは、業界で定義された標準的な言語で開発され、ネットワークの設定とステータス情報を定義できます。

Cisco IOS XEは、Yet Another Next Generation（YANG）データ モデリング言語をサポートしています。YANGをネットワーク設定プロトコル（NETCONF）で使用すると、自動化されたプログラミング可能なネットワーク操作の望ましいソリューションが実現します。NETCONF（RFC 6241）は、クライアント アプリケーションがデバイスからの情報を要求してデバイスに設定変更を加えるために使用するXMLベースのプロトコルです。YANGは主に、NETCONF操作で使用される設定とステート データをモデル化するために使用されます。

Cisco IOS XE では、モデル ベースのインターフェイスは、既存のデバイス CLI、Syslog、およびSNMP インターフェイスと相互運用します。必要に応じて、これらのインターフェイスは、ネットワーク デバイスからノースバウンドに公開されます。YANG は、RFC 6020 に基づいて各プロトコルをモデル化するために使用されます。

（注） 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。

NETCONF

NETCONF は、ネットワーク デバイスの設定をインストール、操作、削除するためのメカニズムです。

コンフィギュレーション データとプロトコル メッセージに Extensible Markup Language（XML）ベースのデータ符号化を使用します。

NETCONF はシンプルなリモート プロシージャ コール（RPC）ベースのメカニズムを使用してクライアントとサーバ間の通信を促進します。クライアントはネットワーク マネージャの一部として実行されているスクリプトやアプリケーションです。通常、サーバはネットワーク デバイス（スイッチまたはルータ）です。サーバは、ネットワーク デバイス全体のトランスポート層としてセキュア シェル（SSH）を使用します。SSH ポート番号 830 をデフォルトのポートとして使用します。ポート番号は、設定可能なオプションです。

NETCONF は、機能の検出およびモデルのダウンロードもサポートしています。サポート対象のモデルは、ietf-netconf-monitoring モデルを使用して検出されます。各モデルに対する改定日付は、機能の応答に示されています。データ モデルは、get-schema RPC を使用して、デバイスからオプションのダウンロードとして入手できます。これらの YANG モデルを使用して、データ モデルを理解したりエクスポートしたりできます。NETCONF の詳細については、RFC 6241 を参照してください。

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。

NETCONF RESTCONF IPv6 のサポート

データ モデル インターフェイス（DMI）は IPv6 プロトコルの使用をサポートしています。DMI による IPv6 のサポートは、クライアント アプリケーションが、IPv6 アドレスを使用するサービスと通信する場合に役に立ちます。外部向けインターフェイスは、IPv4 と IPv6 の両方についてデュアルスタックをサポートします。

DMI は、ネットワーク要素の管理を容易にする一連のサービスです。NETCONF や RESTCONF などのアプリケーション層プロトコルは、ネットワークを介してこれらの DMI にアクセスします。

IPv6 アドレスが設定されていない場合でも、外部向けアプリケーションは IPv6 ソケットをリッスンし続けますが、これらのソケットは到達不能になります。

NETCONF グローバル セッションのロック

NETCONF プロトコルは、デバイス設定を管理し、デバイスの状態情報を取得するための一連の操作を提供します。NETCONF はグローバル ロックをサポートしており、NETCONF では応答しなくなったセッションを kill する機能が導入されています。

複数の同時セッションの全体にわたって一貫性を確保し、設定の競合を防ぐために、セッションのオーナーは NETCONF セッションをロックできます。NETCONF lock RPC は、コンフィギュレーション パーサーと実行コンフィギュレーション データベースをロックします。その他のすべての NETCONF セッション（ロックを所有していない）は、編集操作を実行できません。ただし、読み取り操作は実行できます。これらのロックは存続時間が短いことを意図しており、オーナーは、他の NETCONF クライアント、NETCONF 以外のクライアント（SNMP、CLI スクリプトなど）、および人間のユーザとやり取りをせずに変更を加えることができます。

アクティブ セッションによって保持されているグローバル ロックは、関連付けられたセッションが kill されたときに無効になります。ロックによって、ロックを保持しているセッションが、設定に対して排他的な書き込みアクセスを行えるようになります。グローバル ロックにより設定の変更が拒否された場合は、エラー メッセージによって、NETCONF グローバル ロックが原因で設定の変更が拒否されたことが示されます。

<lock> 操作は必須パラメータ <target> を受け取ります。これは、ロックしようとするコンフィギュレーション データストアの名前です。ロックがアクティブな場合、<edit-config> 操作と <copy-config> 操作は許可されません。

NETCONF のグローバル ロックの保持中に **clear configuration lock** コマンドが指定された場合は、設定の完全な同期がスケジュールされ、警告の syslog メッセージが生成されます。このコマンドは、パーサー コンフィギュレーション ロックのみをクリアします。

次に、<lock> 操作を示す RPC の例を示します。

```
<rpc message-id="101"
          xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
       <lock>
         <target>
           <running/>
         </target>
       </lock>
</rpc>
```

NETCONF Kill セッション

セッションの競合時、またはクライアントによるグローバル ロックの誤用が生じたときは、**show netconf-yang sessions** コマンドを使用して NETCONF セッションをモニタできます。また、**clear netconf-yang session** コマンドを使用して応答しなくなったセッションをクリアすることもできます。**clear netconf-yang session** コマンドは、NETCONF ロックとコンフィギュレーション ロックの両方をクリアします。

<kill-session> 要求は、NETCONF セッションを強制的に終了します。NETCONF エンティティは、オープン セッションの <kill-session> 要求を受信すると、プロセス内のすべての操作を停止し、セッションに関連付けられているすべてのロックとリソースを解放して、関連付けられた接続をすべて閉じます。

<kill-session> 要求には、終了する NETCONF セッションのセッション ID が必要です。セッション ID の値が現在のセッション ID と同じ場合は、無効な値を示すエラーが返されます。NETCONF セッションのトランザクションがまだ進行中に NETCONF セッションが終了した場合は、データ モデル インフラストラクチャによってロールバックが要求され、ネットワーク要素にロールバックが適用されて、すべての YANG モデルの同期がトリガーされます。

セッションの kill が失敗し、グローバル ロックが保持されている場合は、コンソールまたは vty を使用して **clear configuration lock** コマンドを入力します。この時点で、データ モデルを停止して再起動することができます。

候補コンフィギュレーション サポート

候補コンフィギュレーション サポート機能を使用すると、シンプルなコミット オプションを使用して RFC 6241 を実装することによって、候補機能をサポートできます。

候補データストアは、デバイスの実行コンフィギュレーションのコピーを保存する一時的な作業領域となります。実行コンフィギュレーションをデバイスにコミットする前に、実行コンフィギュレーションを作成して変更することができます。候補機能は、NETCONF 機能 urn:ietf:params:netconf:capability:candidate:1.0 により示されます。この NETCONF 機能は、デバイスが候補データストアをサポートしていることを示します。ユーザはこの共有データストアを使用して、デバイスの実行コンフィギュレーションに影響を与えることなく、デバイスのコンフィギュレーションを作成、追加、削除、変更できます。コミット操作では、デバイスのコンフィギュレーションが「候補」から「実行」のコンフィギュレーションにプッシュされます。「候補」データストアが有効になっていると、「実行」のデータ ストアには NETCONF セッションを介して書き込むことができず、すべてのコンフィギュレーションは候補を通じてのみコミットされます。つまり、稼働中の設定を直接変更できる NETCONF 機能は、候補では有効になりません。

（注） 候補データストアは共有データ ストアであることに留意してください。複数の NETCONF セッションが内容を同時に変更する可能性があります。したがって、内容を変更する前にデータストアをロックして、コミットが競合しないようにし、最終的にコンフィギュレーションの変更が失われる可能性を防ぐことが重要になります。

候補の NETCONF 操作

候補データ ストアでは次の操作を実行できます。

（注） この項の情報はRFC6241 の 8.3.4 項を参考にしています。詳細と正確な RPC については、RFC を参照してください。

ロック

<lock> RPC は、ターゲットのデータ ストアを「ロック」するために使用します。これにより、他のユーザは「ロックされた」データ ストアのコンフィギュレーションを変更できなくなります。ロック操作では候補データと実行データの両方をロックできます。

（注） 「ロック中」の候補データストアは、Cisco IOS のコンフィギュレーションのロックや実行コンフィギュレーションのロックに影響を与えません。逆も同様です。

コミット

<commit> RPC は、候補コンフィギュレーションをデバイスの実行コンフィギュレーションにコピーします。「コミット」操作は、候補コンフィギュレーションを更新してコンフィギュレーションをデバイスにプッシュした後に実行する必要があります。

「実行」または「候補」のデータストアのいずれかが別の NETCONF セッションによってロックされている場合、commit RPC は RPC エラー応答で失敗します。<error-tag> は <in-use> となり、<error-info> にはロックを保持している NETCONF セッションの「セッション ID」が示されます。「conf t lock」モードに移行して「グローバル」ロックを使用し、「実行」コンフィギュレーションをロックすることもできますが、コミット操作は RPC エラー応答で失敗し、error-tag の値は <in-use>、セッション ID は「0」になります。

コンフィギュレーション編集

候補コンフィギュレーションは、コンフィギュレーションを変更するための「edit-config（コンフィギュレーション編集）」操作のターゲットとして使用できます。これにより、デバイスの実行コンフィギュレーションに影響を与えることなく、候補内のコンフィギュレーションの変更を準備できるようになります。

廃棄

候補コンフィギュレーションに加えられた変更を削除するには、discard（廃棄）操作を実行して候補コンフィギュレーションを実行コンフィギュレーションに戻します。

たとえば、NETCONF セッション A によってすでに候補データストアの内容が変更されている場合、セッション B が候補をロックしようとするとロックは失敗します。NETCONF セッショ

ンBでは候補をロックする前に、他のNETCONFセッションからの未解決のコンフィギュレーションを削除するために <discard> 操作を実行する必要があります。

ロック解除

ロック、edit-config（コンフィギュレーション編集）、コミットなどで候補コンフィギュレーションを操作した後、Unlock RPC でターゲットとして「candidate」を指定することによって、データストアを「ロック解除」できます。これで、他のセッションでのすべての操作に候補を使用できるようになります。

候補データストアに対する未解決の変更で不具合が発生した場合、コンフィギュレーションの回復が困難になり、他のセッションで問題が生じる可能性があります。問題を回避するため、未解決の変更は、「NETCONFセッションの障害」で暗黙的にロックが解除されたとき、または「Unlock」操作で明示的にロックが解除されたときに廃棄されます。

コンフィギュレーション取得、コンフィギュレーションコピー、コンフィギュレーション検証

候補は、get-config（コンフィギュレーション取得）、copy-config（コンフィギュレーション コピー）、または validate （コンフィギュレーション検証）のどの操作でも、ソースまたはターゲットとして使用できます。候補の変更をデバイスにコミットせずに、コンフィギュレーションの検証のみを行う場合は、<validate> RPC の後に「discard」を付けることで実行できます。

候補の使用方法

次の図は、候補データストアを介してデバイス コンフィギュレーションを変更する場合に推奨されるベスト プラクティスを示しています。

図 *4:* 候補データストアの変更手順

1. 実行データストアをロックします。
2. 候補データストアをロックします。
3. edit-config RPC とターゲットの候補を使用して、候補コンフィギュレーションを変更します。
4. 候補コンフィギュレーションを「実行」にコミットします。
5. 候補と実行をロック解除します。

候補サポートの設定

候補データストア機能は、 **netconf-yang feature candidate-datastore** コマンドを使用して有効にすることができます。データストアの状態が「実行」から「候補」、またはその逆に変わると、変更を有効にするために netconf-yang または restconf の再起動が行われることをユーザに通知する警告メッセージが表示されます。候補が有効になっていると、実行のデータストアにはNETCONFセッションを介して書き込むことができず、すべてのコンフィギュレーションは候補を通じてのみコミットされます。つまり、稼働中の設定を直接変更できるNetconf機能は、候補では有効になりません。

netconf-yang または restconf confd プロセスの開始時に候補または実行のデータストアの選択がコンフィギュレーションで指定されている場合は、次のような警告が表示されます。

```
Device(config)# netconf-yang feature candidate-datastore
netconf-yang initialization in progress - datastore transition not allowed, please try
again after 30 seconds
```

netconf-yang または restconf confd プロセスの開始後に候補または実行の選択が行われた場合は、次のように適用されます。

- **netconf-yang feature candidate-datastore** コマンドが設定されている場合は、コマンドによって「候補」データストアが有効になり、次の警告が出力されます。

  ```
  “netconf-yang and/or restconf is transitioning from running to candidate netconf-yang
   and/or restconf will now be restarted,
  and any sessions in progress will be terminated”.
  ```

- **netconf-yang feature candidate-datastore** コマンドが削除された場合は、コマンドによって「候補」データストアが無効になり、「実行」データストアが有効になり、次の警告が出力されます。

  ```
  netconf-yang and/or restconf is transitioning from candidate to running netconf-yang
   and/or restconf will now be restarted,
  and any sessions in progress will be terminated”.
  ```

- netconf-yang または restconf が再起動すると、進行中のセッションは失われます。

（注） 候補データ ストアは共有データ ストアです。つまり、複数の NETCONF セッションで内容を同時に変更できます。したがって、内容を変更する前にユーザがデータストアをロックして、コミットが競合しないようにし、最終的にコンフィギュレーションの変更が失われる（つまり、他のユーザによってコンフィギュレーションが変更され、コミットが発行されて、コンフィギュレーションが上書きされる）可能性を防ぐことが重要になります。

NETCONF プロトコルの設定方法

NETCONF-YANG は、デバイスのプライマリ トラストポイントを使用します。トラストポイントが存在しない場合に NETCONF-YANG が設定されると、自己署名トラストポイントが作成されます。詳細については、『公開キー インフラストラクチャ コンフィギュレーション ガイド（Cisco IOS XE Gibraltar 16.10.x 向け）』を参照してください。

NETCONF を使用するための権限アクセスの提供

NETCONF API の使用を開始するには、権限レベル 15 を持つユーザである必要があります。そのようにするには、次の作業を行います。

手順の概要

1. **enable**
2. **configure terminal**
3. **username** *name* **privilege** *level* **password** *password*
4. **aaa authentication login default local** および **aaa authorization exec default local**
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）。
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	**username** *name* **privilege** *level* **password** *password* 例： `Device(config)# username example-name privilege 15 password example_password`	ユーザ名をベースとした認証システムを確立します。次のキーワードを設定します。 • **privilege** *level*：ユーザの権限レベルを設定します。プログラマビリティ機能の場合は、15にする必要があります。 • **password** *password*：CLI ビューにアクセスするためのパスワードを設定します。

ステップ 4	**aaa authentication login default local** および **aaa authorization exec default local**
	例：
	`Device (config)# aaa authentication login default local`
	`Device (config)# aaa authorization exec default local`
		（任意）**aaa new-model** を設定する場合は、AAA 認証および許可が必要です。リモート AAA サーバの場合は、local を AAA サーバに置き換えます。

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 モードを終了します。

NETCONF-YANG の設定

レガシー NETCONF プロトコルがデバイスで有効になっている場合、RFC 準拠の NETCONF プロトコルは機能しません。**no netconf legacy** コマンドを使用してレガシー NETCONF プロトコルを無効にしてください。

手順の概要

1. **enable**
2. **configure terminal**
3. **netconf-yang**
4. **netconf-yang feature candidate-datastore**
5. **exit**

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）。
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	**netconf-yang** 例： `Device (config)# netconf-yang`	ネットワークデバイスでNETCONFインターフェイスを有効にします。 （注） CLIによる最初のイネーブル化の後、ネットワーク デバイスをモデル ベースのインターフェイスを通じて管理できるようになります。モデル ベースのインターフェイス プロセスの完全なアクティベーションには、最大 90 秒かかることがあります。
ステップ 4	**netconf-yang feature candidate-datastore** 例： `Device(config)# netconf-yang feature candidate-datastore`	候補データストアを有効にします。
ステップ 5	**exit** 例： `Device (config)# exit`	グローバル コンフィギュレーション モードを終了します。

NETCONF オプションの設定

SNMP の設定

NETCONF を有効にして、サポートされている MIB から生成された YANG モデルを使用して SNMP MIB データにアクセスしたり、IOS でサポートされている SNMP トラップを有効にして、サポートされているトラップから NETCONF 通知を受信するには、IOS で SNMP サーバを有効にします。

次の操作を行ってください。

手順の概要

1. IOS で SNMP 機能を有効にします。
2. NETCONF-YANG が起動した後、次の RPC <edit-config> メッセージを NETCONF-YANG ポートに送信して、SNMP トラップのサポートを有効にします。
3. 次の RPC メッセージを NETCONF-YANG ポートに送信して、実行コンフィギュレーションをスタートアップ コンフィギュレーションに保存します。

手順の詳細

ステップ 1 IOS で SNMP 機能を有効にします。

例：

```
configure terminal
logging history debugging
logging snmp-trap emergencies
logging snmp-trap alerts
logging snmp-trap critical
logging snmp-trap errors
logging snmp-trap warnings
logging snmp-trap notifications
logging snmp-trap informational
logging snmp-trap debugging
!
snmp-server community public RW
snmp-server trap link ietf
snmp-server enable traps snmp authentication linkdown linkup
snmp-server enable traps syslog
snmp-server manager
exit
```

ステップ 2 NETCONF-YANG が起動した後、次の RPC <edit-config> メッセージを NETCONF-YANG ポートに送信して、SNMP トラップのサポートを有効にします。

例：

```
<?xml version="1.0" encoding="utf-8"?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
```

```
      <netconf-yang xmlns="http://cisco.com/yang/cisco-self-mgmt">
        <cisco-ia xmlns="http://cisco.com/yang/cisco-ia">
          <snmp-trap-control>
            <trap-list>
              <trap-oid>1.3.6.1.4.1.9.9.41.2.0.1</trap-oid>
            </trap-list>
            <trap-list>
              <trap-oid>1.3.6.1.6.3.1.1.5.3</trap-oid>
            </trap-list>
            <trap-list>
              <trap-oid>1.3.6.1.6.3.1.1.5.4</trap-oid>
            </trap-list>
          </snmp-trap-control>
        </cisco-ia>
      </netconf-yang>
    </config>
  </edit-config>
</rpc>
```

ステップ 3 次の RPC メッセージを NETCONF-YANG ポートに送信して、実行コンフィギュレーションをスタートアップ コンフィギュレーションに保存します。

例：

```
<?xml version="1.0" encoding="utf-8"?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="">
  <cisco-ia:save-config xmlns:cisco-ia="http://cisco.com/yang/cisco-ia"/>
</rpc>
```

NETCONF プロトコルのコンフィギュレーションの確認

NETCONF コンフィギュレーションを確認するには次のコマンドを使用します。

手順の概要

1. **show netconf-yang datastores**
2. **show netconf-yang sessions**
3. **show netconf-yang sessions detail**
4. **show netconf-yang statistics**
5. **show platform software yang-management process**

手順の詳細

ステップ 1 **show netconf-yang datastores**

NETCONF-YANG データストアに関する情報を表示します。

例：

```
Device# show netconf-yang datastores

Device# show netconf-yang datastores
Datastore Name : running
```

```
Globally Locked By Session : 42
Globally Locked Time : 2018-01-15T14:25:14-05:00
```

ステップ 2 show netconf-yang sessions

NETCONF-YANG セッションに関する情報を表示します。

例：

```
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 : 10
session-id transport username source-host global-lock
--------------------------------------------------------------------------------
40 netconf-ssh admin 10.85.70.224 None
42 netconf-ssh admin 10.85.70.224 None
44 netconf-ssh admin 10.85.70.224 None
46 netconf-ssh admin 10.85.70.224 None
48 netconf-ssh admin 10.85.70.224 None
50 netconf-ssh admin 10.85.70.224 None
52 netconf-ssh admin 10.85.70.224 None
54 netconf-ssh admin 10.85.70.224 None
56 netconf-ssh admin 10.85.70.224 None
58 netconf-ssh admin 10.85.70.224 None
```

ステップ 3 show netconf-yang sessions detail

NETCONF-YANG セッションに関する詳細情報を表示します。

例：

```
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
```

ステップ 4 show netconf-yang statistics

NETCONF-YANG 統計に関する情報を表示します。

例：

```
Device# show netconf-yang statistics
```

```
netconf-start-time : 2018-01-15T12:51:14-05:00
in-rpcs : 0
in-bad-rpcs : 0
out-rpc-errors : 0
out-notifications : 0
in-sessions : 10
dropped-sessions : 0
in-bad-hellos : 0
```

ステップ 5 show platform software yang-management process

NETCONF-YANG のサポートに必要なソフトウェア プロセスのステータスを表示します。

例：

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

confd            : Running
nesd             : Running
syncfd           : Running
ncsshd           : Running
dmiauthd         : Running
vtyserverutild   : Running
opdatamgrd       : Running
nginx            : Running
ndbmand          : Running
```

（注） プロセス nginx は、**ip http secure-server** または **ip http server** がデバイスで設定されている場合に実行されます。このプロセスが「実行」状態でなくても NETCONF は正常に機能します。ただし、RESTCONF には nginx プロセスが必要です。

表 *13 : show platform software yang-management process* のフィールドの説明

フィールド	説明
confd	コンフィギュレーション デーモン
nesd	ネットワーク要素シンクロナイザ デーモン
syncfd	デーモンからの同期
ncsshd	NETCONF セキュア シェル（SSH）デーモン
dmiauthd	デバイス管理インターフェイス（DMI）認証デーモン
vtyserverutild	VTY サーバ ユーティリティ デーモン
opdatamgrd	運用データ マネージャ デーモン
nginx	NGINX Web サーバ
ndbmand	NETCONF データベース マネージャ

NETCONF 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 6241	ネットワーク設定プロトコル（*NETCONF*）
RFC 6536	ネットワーク設定プロトコル（*NETCONF*）アクセス制御モデル
RFC 8040	*RESTCONF* プロトコル

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

NETCONF 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。

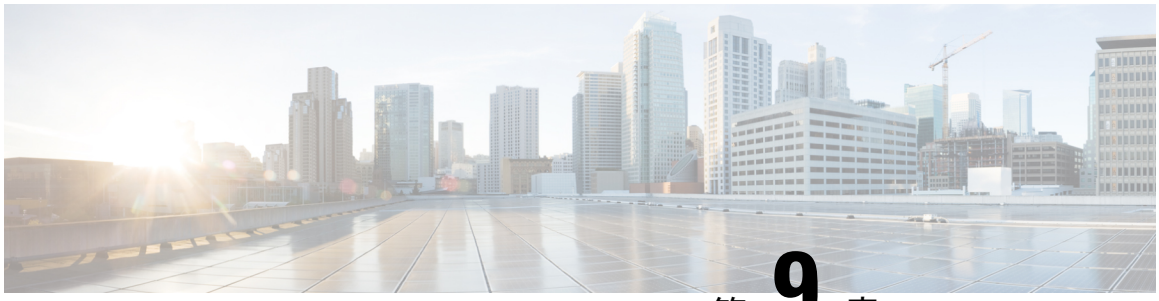
表 *14 : NETCONF* プロトコルの機能情報

機能名	リリース	機能情報
候補コンフィギュレーション サポート	Cisco IOS XE Fuji 16.9.1	候補コンフィギュレーション サポート機能を使用すると、シンプルなコミットオプションを使用して RFC 6241 を実装することによって、候補機能をサポートできます。 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 ASR 1000 シリーズ アグリゲーション サービス 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 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 シリーズ スイッチ • Cisco CBR-8 シリーズ ルータ • Cisco Cloud Services Router 1000V シリーズ • Cisco ISR 4000 シリーズ サービス統合型ルータ 次のコマンドが導入されました： **netconf-yang feature candidate-datastore**

機能名	リリース	機能情報
NETCONF ネットワーク管理インターフェイス	Cisco IOS XE Denali 16.3.1	NETCONF プロトコル機能によって、プログラムによる各種の標準規格に準拠した方法で、設定の記述やネットワーク デバイスからの運用データの読み取りが容易になります。 次のコマンドが導入されました：**netconf-yang**
	Cisco IOS XE Everest 16.5.1a	この機能は、Cisco Catalyst 9300 シリーズ スイッチと Cisco Catalyst 9500 シリーズ スイッチに実装されました。
	Cisco IOS XE Everest 16.6.2	この機能は、Cisco Catalyst 9400 シリーズ スイッチに実装されました。
	Cisco IOS XE Fuji 16.7.1	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 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 Network Convergence System 4200 シリーズ
	Cisco IOS XE Fuji 16.8.1a	この機能は、Cisco Catalyst 9500 ハイ パフォーマンス シリーズ スイッチに実装されていました。
	Cisco IOS XE Fuji 16.9.2	この機能は、Cisco Catalyst 9200 シリーズ スイッチに実装されました。

機能名	リリース	機能情報
NETCONF および RESTCONF IPv6 のサポート	Cisco IOS XE Fuji 16.8.1a	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 ASR 1000 シリーズ アグリゲーション サービス 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 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 シリーズ スイッチ • Cisco CBR-8 シリーズ ルータ • Cisco CSR 1000v スイッチ • Cisco IR1101 耐環境性能 サービス統合型ルータ • Cisco ISR 4000 シリーズ サービス統合型ルータ

機能名	リリース	機能情報
NETCONF グローバル ロックおよびセッションの kill	Cisco IOS XE Fuji 16.8.1a	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 ASR 1000 シリーズ アグリゲーション サービス 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 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 シリーズ スイッチ • Cisco CBR-8 シリーズ ルータ • Cisco CSR 1000v スイッチ • Cisco ISR 1100 シリーズ サービス統合型ルータ • Cisco ISR 4000 シリーズ サービス統合型ルータ



第 9 章

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 プログラマブル インターフェイスについて

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、https://github.com/YangModels/yang/tree/master/vendor/cisco/xe/1681GitHub 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

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 プログラマブル インターフェイスの設定方法

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）。
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。

	コマンドまたはアクション	目的
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： `Device(config)# aaa session-id common`	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。
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。

	コマンドまたはアクション	目的
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）。
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： `Device(config)# ip http secure-server`	セキュア HTTP（HTTPS）サーバをイネーブルにします。
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 プログラマブル インターフェイスの設定例

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作成）

```
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	開発者にわかりやすい方法で 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	Representational State Transfer Configuration Protocol（RESTCONF）

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

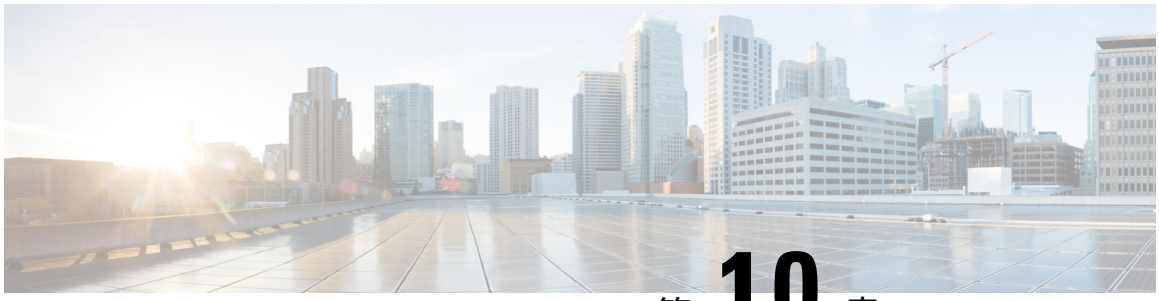
RESTCONF プロトコルの機能情報

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 *15 : RESTCONF* ネットワーク管理インターフェイスの機能情報

機能名	リリース	機能情報
RESTCONF ネットワーク管理インターフェイス	Cisco IOS XE Everest 16.6.1	この章では、HTTP ベースのプロトコルである Representational State Transfer Configuration Protocol（RESTCONF）の設定および構成方法を説明します。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 モデル固有のリモート プロシージャ コール（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。 この機能は、ASR 1000 アグリゲーション サービス ルータの ASR1001-HX および ASR1002-HX、CSR 1000v シリーズ クラウド サービス ルータ、および Cisco 4000 シリーズ サービス統合型ルータ（ISR）に導入されました。 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	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 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 シリーズ スイッチ
	Cisco IOS XE Fuji 16.9.2	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 シリーズ スイッチ



第 10 章

gNMI プロトコル

この機能では、gRPC ネットワーク管理インターフェイス（gNMI）の機能を使用したモデル駆動型の設定と運用データの取得、およびリモート プロシージャ コール（RPC）の取得と設定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

機能には、次のような制約事項が適用されます。

- Subscribe RPC サービスはサポートされていません。
- JSON、BYTES、PROTO、および ASCI エンコーディング オプションはサポートされていません。

  JSON IETF 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：
  - 運用データのフィルタリングはサポートされていません。
  - モデルの使用はサポートされていません。これらは、Get RPC コールへの応答として返す必要があるデータ要素を定義するスキーマ定義モジュールを示す一連のモデル データ メッセージです。
- GetResponse：

- エイリアスはサポートされていません。これは、通知メッセージの中で指定されたプレフィックスのエイリアスを提供する文字列です。
- 削除はサポートされていません。これは、データ ツリーから削除する一連のパスです。

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 ネットワーク管理インターフェイスです。gNMI はネットワーク デバイスの設定をインストール、操作、および削除し、また、運用データの表示も実行するメカニズムです。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。

gNMI 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。

YANG データ ツリーの JSON IETF エンコーディング

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。gNMI は、コンテンツ層でのデータのエンコードに JSON を使用します。

JSON タイプは、値が JSON 文字列としてエンコードされていることを示します。JSON_IETF でエンコードされたデータは、RFC 7951 で規定されている JSON シリアル化のルールに準拠している必要があります。クライアントとターゲットの両方が JSON エンコーディングをサポートしている必要があります。

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。この値には、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 ノードの場合（子が存在せず、関連付けられた値を持つ）、そのリーフの値が直接エンコードされます（そのままの JSON 値が含まれています。JSON オブジェクトは必要ありません）。

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。

表 *16 : GetRequest* の *JSON* 構造

GetRequest	GetResponse

The following is a path for the openconfig-interfaces model
++++++++ Sending get request: ++++++++
path {
  elem {
  name: "interfaces"
  }
  elem {
  name: "interface"
  key {
  key: "name"
  value: "Loopback111"
  }
  }
}	

GetRequest	GetResponse

	encoding: JSON_IETF
	++++++++ Received get response: ++++++++
	notification {
	  timestamp: 1521699434792345469
	  update {
	  path {
	  elem {
	  name: "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\

GetRequest	GetResponse
	"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\t\ "openconfig-interfaces:config\":\t{\n\t\t\t\t\ "openconfig-interfaces:index\":\t0,\n\t\t\t\t\ "openconfig-interfaces:name\":\t\"Loopback111\",\n\t\t\t\t\ "openconfig-interfaces:enabled\":\t\"true\"\n\t\t},\n\t\t\

GetRequest	GetResponse
	"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\

GetRequest	GetResponse
	"openconfig-interfaces:last-clear\":\ t2018\n\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}" } } }

表 *17 :* リーフ値に対する *GetRequest*

GetRequest	GetResponse

++++++++ Sending get request: ++++++++
path {
elem {
name: "interfaces"
}
elem {
name: "interface"
key {
key: "name"
value: "Loopback111"
}
}
elem {
name: "state"
}
elem {
name: "oper-status"
}
}
	encoding: JSON_IETF
	++++++++ Received get response: ++++++++
	notification {
	timestamp: 1521699326012374332
	update {
	path {
	elem {
	name: "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 は JSON キーもサポートしており、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。

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。

表 *18 : SetRequest* の *JSON* 構造の例

SetRequest	SetResponse

`++++++++ Sending set request: ++++++++`
`update {`
`  path {`
`  elem {`
`  name: "interfaces"`
`  }`
`  elem {`
`  name: "interface"`
`  key {`
`  key: "name"`
`  value: "Loopback111"`
`  }`
`  }`
`  elem {`
`  name: "config"`
`  }`
`  }`
`  val {`
`  json_ietf_val:`
`"{\"openconfig-interfaces:enabled\":\"false\"}"`
`  }`
`}`
	`++++++++ Received set response: ++++++++`
	`response {`
	`  path {`
	`  elem {`
	`  name: "interfaces"`
	`  }`
	`  elem {`
	`  name: "interface"`
	`  key {`
	`  key: "name"`
	`  value: "Loopback111"`
	`  }`
	`  }`
	`  elem {`
	`  name: "config"`
	`  }`
	`  }`
	`  op: UPDATE`
	`}`
	`timestamp: 1521699342123890045`

表 *19 :* リーフ値での *SetRequest* の例

SetRequest	SetResponse

++++++++ Sending set request: ++++++++
update {
  path {
  elem {
  name: "interfaces"
  }
  elem {
  name: "interface"
  key {
  key: "name"
  value: "Loopback111"
  }
  }
  elem {
  name: "config"
  }
  elem {
  name: "description"
  }
  }
  val {
  json_ietf_val: "\"UPDATE DESCRIPTION\""
  }
}
	++++++++ Received set response: ++++++++
	response {
	  path {
	  elem {
	  name: "interfaces"
	  }
	  elem {
	  name: "interface"
	  key {
	  key: "name"
	  value: "Loopback111"
	  }
	  }
	  elem {
	  name: "config"
	  }
	  elem {
	  name: "description"
	  }
	  }
	  op: UPDATE
	}
	timestamp: 1521699342123890045

gNMI の名前空間

名前空間は、メッセージの origin フィールドで使用されるパスプレフィックスを指定します。

ここでは、Cisco IOS XE Gibraltar 16.10.1 以降のリリースで使用される名前空間について説明します。

- RFC 7951 で指定された名前空間：パス プレフィックスは、RFC 7951 で定義されている YANG モジュール名を使用します。

  RFC 7951 で指定された値のプレフィックスは、YANG モジュール名を使用します。

  値のプレフィックスは、選択されたパスプレフィックスの名前空間の影響を受けません。次に、RFC 7951 で指定された値のプレフィックスの例を示します。

  ```
  val {
     json_ietf_val:"{
       "openconfig-interfaces:config": {
         "openconfig-interfaces:description":
           "DESCRIPTION"
       }
     }"
   }
  ```

  RFC 7951 で指定された名前空間プレフィックスは、YANG モジュール名も使用します。たとえば、ループバック インターフェイスへの openconfig パスは次のようになります。

  ```
  /openconfig-interfaces:interfaces/interface[name=Loopback111]/
  ```

次の例は、RFC 7951 の名前空間指定を使用した gNMI パスを示しています。

```
path {
  origin: "rfc7951"
  elem {
    name: "openconfig-interface:interfaces"
  }
  elem {
    name: "interface"
    key {
      key: "name"
      value: "Loopback111"
    }
  }
}
```

- openconfig：パスプレフィックスを使用しません。これらは openconfig モデルへのパスでのみ使用できます。

  openconfig 名前空間プレフィックスの動作は、発信元または名前空間が指定されていない場合と同じです。たとえば、ループバック インターフェイスへの openconfig パスは次のようになります。

  ```
  /interfaces/interface[name=Loopback111]/
  ```

  次の例は、openconfig 名前空間指定を使用した gNMI パスを示しています。

  ```
  path {
    origin: "openconfig"
    elem {
      name: "interfaces"
    }
    elem {
      name: "interface"
      key {
        key: "name"
        value: "Loopback111"
      }
    }
  }
  ```

- 空：openconfig プレフィックスと同じです。これがデフォルトです。

  次の例は、空の openconfig 名前空間指定を使用した gNMI パスを示しています。

  ```
  path {
  elem {
      name: "interfaces"
    }
    elem {
      name: "interface"
      key {
        key: "name"
        value: "Loopback111"
      }
    }
  }
  ```

ここでは、Cisco IOS XE Gibraltar 16.10.1 より前のリリースで使用されるパス プレフィックスについて説明します。

ここでは、パス プレフィックスは、YANG モジュール定義で定義されている YANG モジュール プレフィックスを使用します。たとえば、ループバック インターフェイスへの openconfig パスは次のようになります。

```
/oc-if:interfaces/interface[name=Loopback111]/
```

次の例は、従来の名前空間指定を使用した gNMI パスを示しています。

```
path {
  origin: "legacy"
  elem {
    name: "oc-if:interfaces"
  }
  elem {
    name: "interface"
    key {
      key: "name"
      value: "Loopback111"
    }
  }
}
```

gNMI のワイルドカード

gNMI プロトコルは、Get パスのワイルドカードをサポートしています。これは、複数の要素を一致させるためにパス内でワイルドカードを使用する機能です。これらのワイルドカードは、スキーマ内の指定されたサブツリーにあるすべての要素を示します。

elem は要素であり、xpath 内の / 文字の間の値です。elem は gNMI パスでも使用できます。たとえば、 elem 名を基準とするワイルドカードの位置は、ワイルドカードがインターフェイスを表し、すべてのインターフェイスとして解釈されることを暗に意味します。

ワイルドカードには暗黙的と明示的の2つのタイプがあり、どちらもサポートされています。Get パスは、パス ワイルドカードのすべてのタイプと組み合わせをサポートします。

- 暗黙的なワイルドカード：これらは、要素ツリー内の要素のリストを展開します。暗黙的なワイルドカードは、リストの要素にキー値が指定されていない場合に出現します。

  次に、パスの暗黙的なワイルドカードの例を示します。このワイルドカードは、デバイスにあるすべてのインターフェイスの説明を返します。

  ```
  path {
    elem {
      name: "interfaces"
    }
    elem {
      name: "interface"
    }
    elem {
      name: "config"
    }
    elem {
      name: "description"
    }
  }
  ```

- 明示的なワイルドカード：下記の指定によって同じ機能を提供します。

- パス要素名またはキー名のいずれかにアスタリスク（*）を指定します。

  次に、パスのアスタリスク ワイルドカードをキー名として使用する例を示します。このワイルドカードは、デバイスにあるすべてのインターフェイスの説明を返します。

```
path {
  elem {
    name: "interfaces"
  }
  elem {
    name: "interface"
    key {
     key: "name"
      value: "*"
    }
  }
  elem {
    name: "config"
  }
  elem {
    name: "description"
  }
}
```

  次に、パスのアスタリスク ワイルドカードをパス名として使用する例を示します。このワイルドカードは、Loopback111 インターフェイスで使用可能なすべての要素の説明を返します。

```
path {
  elem {
    name: "interfaces"
  }
  elem {
    name: "interface"
    key {
      key: "name"
      value: "Loopback111"
    }
  }
   elem {
    name: "*"
  }
  elem {
    name: "description"
  }
}
```

- 要素名として省略記号（...）または空のエントリを指定します。これらのワイルドカードは、パス内の複数の要素に展開できます。

  次に、パスの省略記号ワイルドカードの例を示します。このワイルドカードは、/interfaces 配下で使用可能なすべての説明フィールドを返します。

```
path {
  elem {
    name: "interfaces"
  }
  elem {
    name: "..."
  }
  elem {
    name: "description"
```

```
          }
        }
```

次に、暗黙的なワイルドカードを使用した GetRequest の例を示します。この GetRequest は、デバイスにあるすべてのインターフェイスの oper-status を返します。

```
path {
  elem {
    name: "interfaces"
    }
  elem {
    name: "interface"
  }
  elem {
    name: "state"
  }
  elem {
    name: "oper-status"
  }
},
type: 0,
encoding: 4
```

次に、暗黙的なワイルドカードを使用した GetResponse の例を示します。

```
notification {
  timestamp: 1520627877608777450
  update {
    path {
      elem {
        name: "interfaces"
      }
      elem {
        name: "interface"
        key {
          key: "name"
          value: "\"FortyGigabitEthernet1/1/1\""
        }
      }
      elem {
        name: "state"
}
      elem {
        name: "oper-status"
      }
    }
    val {
      json_ietf_val: "\"LOWER_LAYER_DOWN\""
    }
  },

<snip>
…
</snip>

update {
    path {
      elem {
        name: "interfaces"
      }
```

```
      elem {
        name: "interface"
        key {
          key: "name"
          value: "\"Vlan1\""
        }
      }
      elem {
        name: "state"
      }
      elem {
        name: "oper-status"
      }
    }
    val {
      json_ietf_val: "\"DOWN\""
    }
  }
}
```

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.ABORTED,
   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

gNMI プロトコル を有効にするには、次の手順を実行します。

1. gNMIクライアントと、認証局（CA）によって署名されたデバイス用に一連の証明書を作成します。
    1. Linux で OpenSSL を使用して証明書を作成します。

2. デバイスに証明書をインストールします。
3. デバイスで gNMI を設定します。
4. gNMI が有効になっていて実行されているかどうかを確認します。

2. 前の手順で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 クライアントを接続します。

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

[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）。
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`	gNMIb 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 50052 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。
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

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）。
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`	gNMIb 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。

	コマンドまたはアクション	目的
ステップ 6	**gnmi-yang secure-client-auth** 例： `Device(config)# gnmi-yang secure-client-auth`	（任意）gNMIb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 50051 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 プロトコルの設定例

例：gNMI プロトコルの有効化

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 50052.>
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 50051.>
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	Title
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

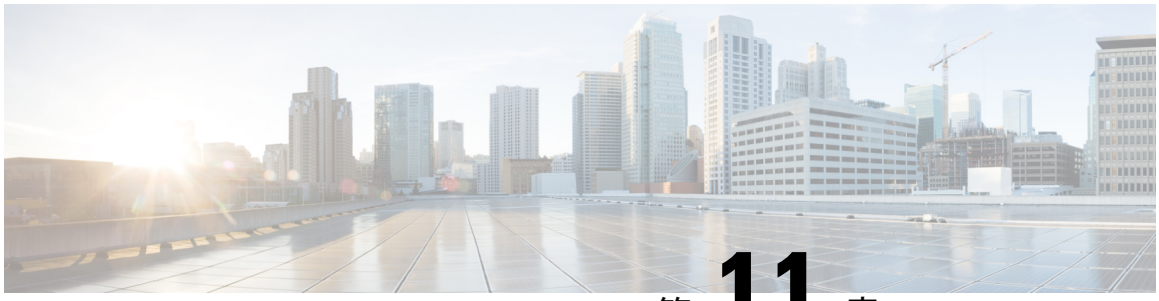
gNMI プロトコルの機能情報

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 *20 : 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 ジブラルタル 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



第 11 章

モデルベースの AAA

NETCONF インターフェイスと RESTCONFイ ンターフェイスは、NETCONF アクセス制御モデル（NACM）を実装しています。NACM は、RFC 6536 で規定されたロールベース アクセス コントロール（RBAC）の形式の 1 つです。

モデルベースの AAA

モデルベースの AAA の前提条件

モデルベースの AAA 機能を使用するには、次の内容について事前に理解しておく必要があります。

- NETCONF-YANG
- NETCONF-YANG kill セッション
- RFC 6536：ネットワーク設定プロトコル（NETCONF）アクセス制御モデル

初期操作

NETCONF サービスや RESTCONF サービスが有効になると、/nacm サブツリーが事前に設定されていないデバイスは、特権レベル 15 のユーザ以外のすべての操作とデータへの読み取り/書き込み/実行アクセスを拒否します。これについては、/nacm サブツリーの次の設定に記述されています。

```
<nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
      <enable-nacm>true</enable-nacm>
      <read-default>deny</read-default>
      <write-default>deny</write-default>
      <exec-default>deny</exec-default>
      <enable-external-groups>true</enable-external-groups>
      <rule-list>
```

```
      <name>admin</name>
      <group>PRIV15</group>
      <rule>
        <name>permit-all</name>
        <module-name>*</module-name>
        <access-operations>*</access-operations>
        <action>permit</action>
      </rule>
    </rule-list>
  </nacm>
```

グループ メンバーシップ

ユーザのグループ メンバーシップは2つのソースから取得できます。1つ目は、認証に使用するAAAサーバで設定されているユーザの権限レベルです。2つ目は、/nacm/groupsサブツリーで設定されている権限レベルです。各権限レベルに対応するグループの名前は次のとおりです。

権限レベル	NACM グループ名
0	PRIV00
1	PRIV01
2	PRIV02
3	PRIV03
4	PRIV04
5	PRIV05
6	PRIV06
7	PRIV07
8	PRIV08
9	PRIV09
10	PRIV10
11	PRIV11
12	PRIV12
13	PRIV13
14	PRIV14
15	PRIV15

（注） 従来の IOS コマンド許可（権限レベルに基づくものなど）は、NETCONF または RESTCONF には適用されません。

（注） 権限レベルに基づいて NACM グループに付与されたアクセスは、権限レベルが高い NACM グループには本来適用されません。たとえば、PRIV10 に適用されるルールは、PRIV11、PRIV12、PRIV13、PRIV14、および PRIV15 にも自動的に適用されるわけではありません。

NACM 権限レベルの依存関係

AAA 設定が **no aaa new-model** で設定されている場合は、ユーザに対してローカルに設定された権限レベルが使用されます。AAA 設定が **aaa new-model** で設定されている場合、権限レベルは、メソッドリスト **aaa authorization exec default** に関連付けられている AAA サーバによって決まります。

NACM の設定の管理と保守

NACM 設定は、NETCONF または RESTCONF を使用して変更できます。ユーザが NCAM 設定にアクセスできるようにするには、そのための明示的な権限を持たせる必要があります。つまり、NACM ルールを使用します。/nacm サブツリーの下の設定は、**copy running-config startup-config** EXEC コマンドが発行されるとき、または **cisco-ia:save-config** RPC が発行されるときは持続します。

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <save-config xmlns="http://cisco.com/yang/cisco-ia"/>
</rpc>
```

（注） NETCONF セッションに適用される NACM ルールは、セッションの確立時に /nacm サブツリーで設定されているものです。/nacm サブツリーに変更を加えても、NETCONF セッションはすでに確立されているため影響を受けません。<kill-session> RPC または **clear netconf-yang session** EXEC コマンドを使用して、不要な NETCONF セッションを強制的に終了することができます。NETCONF Kill セッション （108 ページ）を参照してください。

（注） 特定のデータへのアクセスを拒否するルールを作成する場合は、同じデータが複数の YANG モジュールとデータ ノードのパスを介して公開される可能性があるため、注意が必要です。たとえば、インターフェイス コンフィギュレーションは **Cisco-IOS-XE-native** と **ietf-interface** の両方を介して公開されます。同じ元データの 1 つの表現に適用される可能性があるルールは、そのデータの他の表現には適用されない場合があります。

NACM 設定のリセット

/nacm サブツリーの設定を初期設定にリセットするには、次のコマンドを使用します（初期操作を参照）。

```
Router#request platform software yang-management nacm reset-config
```

NACM の設定例

（注） ここで挙げている例は説明のみを目的とするものです。

次に、グループ設定の例を示します。

```
<nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
    <groups>
      <group>
        <name>administrators</name>
        <user-name>admin</user-name>
        <user-name>root</user-name>
      </group>

      <group>
        <name>limited-permission</name>
        <user-name>alice</user-name>
        <user-name>bob</user-name>
      </group>
    </groups>
  </nacm>
```

表 *21* : グループ設定の設定パラメータの説明

パラメータ	説明
<name>administrators</name>	グループ名
<user-name>admin</user-name>	ユーザ名
<user-name>root</user-name>	ユーザ名

次に、モジュール ルールを作成する例を示します。

```
  <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
    <rule-list>
      <name>only-ietf-interfaces</name>
      <group>limited-permission</group>
      <rule>
        <name>deny-native</name>
        <module-name>Cisco-IOS-XE-native</module-name>
        <access-operations>*</access-operations>
        <action>deny</action>
      </rule>
      <rule>
        <name>allow-ietf-interfaces</name>
        <module-name>ietf-interfaces</module-name>
```

```
      <access-operations>*</access-operations>
      <action>permit</action>
    </rule>
  </rule-list>
</nacm>
```

表 *22 :* モジュール ルールを作成するための設定パラメータの説明

パラメータ	説明
<name>only-ietf-interfaces</name>	固有のルールリスト名
< group > permission </group >	ルールリストが適用されるグループ
<name>deny-native</name>	固有のルール名
<module-name>Cisco-IOS-XE-native</module-name>	YANG モジュールの名前
<access-operations>*</access-operations>	CRUDx の動作タイプ
<action>deny</action>	許可/拒否

次に、プロトコル操作ルールを作成する例を示します。

```
<nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
  <rule-list>
    <name>only-get</name>
    <group>limited-permission</group>

    <rule>
      <name>deny-edit-config</name>
      <module-name>ietf-netconf</module-name>
      <rpc-name>edit-config</rpc-name>
      <access-operations>exec</access-operations>
      <action>deny</action>
    </rule>
    <rule>
      <name>allow-get</name>
      <module-name>ietf-netconf</module-name>
      <rpc-name>get</rpc-name>
      <access-operations>exec</access-operations>
      <action>permit</action>
    </rule>
  </rule-list>
</nacm>
```

表 *23 :* プロトコル操作ルールを作成するための設定パラメータの説明

パラメータ	説明
<name>only-get</name>	固有のルールリスト名
<group>limited-permission</group>	ルールリストが適用されるグループ
<name>deny-edit-config</name>	固有のルール名

パラメータ	説明
<module-name>ietf-netconf</module-name>	RPC を含むモジュールの名前
<rpc-name>edit-config</rpc-name>	RPC の名前
<access-operations>exec</access-operations>	RPC の実行権限
<action>deny</action>	許可/拒否

次に、データ ノード ルールを作成する例を示します。

```
<nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
  <rule-list>
    <name>hide-enable-passwords</name>
    <group>limited-permission</group>

    <rule>
      <name>deny-enable-passwords</name>
     <path xmlns:ios="http://cisco.com/ns/yang/Cisco-IOS-XE-native">/ios:native/enable

      </path>
      <access-operations>*</access-operations>
      <action>deny</action>
    </rule>
  </rule-list>
</nacm>
```

表 *24*：データ ノード ルールを作成するための設定パラメータの説明

パラメータ	説明
<name>hide-enable-passwords</name>	固有のルールリスト名
<group>limited-permission</group>	ルールリストが適用されるグループ
<name>deny-enable-passwords</name>	固有のルール名
<path xmlns:ios="http://cisco.com/ns/yang/Cisco-IOS-XE-native">/ios:native/enable</path>	許可または拒否されるデータ ノードへのパス
<access-operations>*</access-operations>	CRUDx の動作タイプ
<action>deny</action>	許可/拒否

次に、すべてのグループに対して、標準の NETCONF RPC <get> および <get-config> の使用、スキーマ ダウンロード RPC <get-schema>、およびモジュール **ietf-interfaces** にあるデータへの読み取り専用アクセスを許可する NACM 設定の例を示します。

```
<nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
      <rule-list>
        <name>readonly-protocol</name>
        <group>*</group>
        <rule>
          <name>get-permit</name>
          <module-name>ietf-netconf</module-name>
```

```
      <rpc-name>get</rpc-name>
      <access-operations>exec</access-operations>
      <action>permit</action>
    </rule>
    <rule>
      <name>get-config-permit</name>
      <module-name>ietf-netconf</module-name>
      <rpc-name>get-config</rpc-name>
      <access-operations>exec</access-operations>
      <action>permit</action>
    </rule>
    <rule>
      <name>get-schema-permit</name>
      <module-name>ietf-netconf-monitoring</module-name>
      <rpc-name>get-schema</rpc-name>
      <access-operations>exec</access-operations>
      <action>permit</action>
    </rule>
  </rule-list>
  <rule-list>
    <name>readonly-data</name>
    <group>*</group>
    <rule>
      <name>ietf-interfaces-permit</name>
      <module-name>ietf-interfaces</module-name>
      <access-operations>read</access-operations>
      <action>permit</action>
    </rule>
  </rule-list>
</nacm>
```

モデルベースの AAA に関するその他の参考資料

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 6241	ネットワーク設定プロトコル（*NETCONF*）
RFC 6536	ネットワーク設定プロトコル（*NETCONF*）アクセス制御モデル

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

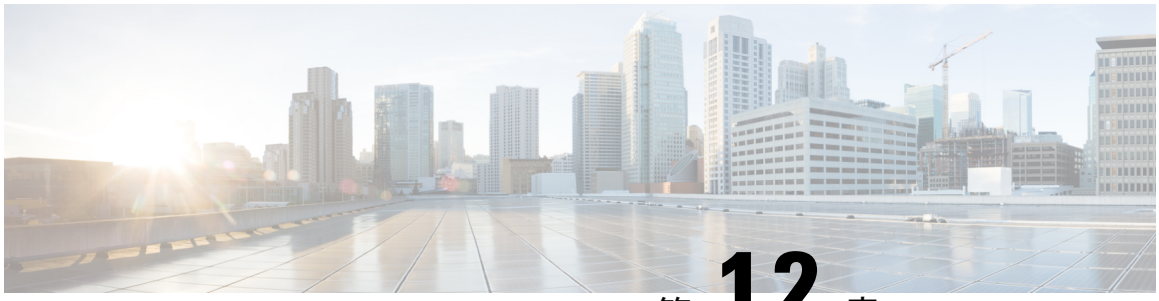
モデルベースの AAA に関する機能情報

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 *25 :* プログラマビリティの機能情報：データ モデル

機能名	リリース	機能情報
モデルベースの AAA	Cisco IOS XE Fuji 16.8.1	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 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 ASR 1000 シリーズ アグリゲーション サービス ルータ • Cisco CSR 1000v スイッチ • Cisco ISR 1100 シリーズ サービス統合型ルータ • Cisco ISR 4000 シリーズ サービス統合型ルータ • Cisco NCS 4200 シリーズ
	Cisco IOS XE Fuji 16.8.1a	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 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 シリーズ スイッチ



第 12 章

モデル駆動型テレメトリ

モデル駆動型テレメトリ

モデル駆動型テレメトリは、YANG モデル化されたデータをデータ コレクタにストリーミングするためのメカニズムを提供します。このモジュールでは、モデル駆動型テレメトリについて説明し、テレメトリ RPC の例を示します。

モデル駆動型テレメトリの前提条件

- テレメトリを使用する際に必要なデータを理解して定義するには、YANGに関する知識が必要です。
- XML、XML 名前空間、および XML XPath の知識。
- IETF テレメトリ仕様で定義されている標準および原則の知識。
- NETCONF-YANG がデバイス上で設定済みであり稼働している必要があります。

（注） NETCONF を使用しない場合でも、テレメトリが機能するように NETCONF-YANG を設定する必要があります。NETCONF-YANG の設定の詳細については、「NETCONF プロトコル」モジュールを参照してください。

show platform software yang-management process コマンドを使用して、次のプロセスが実行中であることを確認します。

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

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

```
dmiauthd         : Running
vtyserverutild   : Running
opdatamgrd       : Running
nginx            : Running
ndbmand          : Running
pubd             : Running
```

（注） プロセス pubd はモデル駆動型テレメトリ プロセスであり、これが実行していない場合にはモデル駆動型テレメトリは機能しません。

表 *26:* フィールドの説明

DMI プロセス名	主要な役割
confd	コンフィギュレーション デーモン
nesd	ネットワーク要素シンクロナイザ デーモン
syncfd	同期デーモン（実行状態と対応するモデル間の同期を維持）
ncsshd	NETCONF セキュア シェル（SSH）デーモン
dmiauthd	デバイス管理インターフェイス（DMI）認証デーモン
nginx	NGINX Web サーバ。RESTCONF の Web サーバとして機能します。
ndbmand	NETCONF データベース マネージャ
pubd	モデル駆動型テレメトリに使用されるパブリケーションマネージャおよびパブリッシャ

- urn:ietf:params:netconf:capability:notification:1.1 機能は、hello メッセージでリストする必要があります。この機能は、IETF テレメトリをサポートするデバイスでのみアドバタイズされます。

NETCONF 固有の前提条件

- NETCONF とその使用方法に関する次の知識。
  - NETCONF セッションの確立。
  - hello メッセージと機能メッセージの送信および受信。

- 確立された NETCONF セッションによる YANG XML リモート プロシージャ コール（RPC）の送受信詳細については、『NETCONF-YANG の設定例』を参照してください。

NETCONF の有効化と検証

NETCONF の機能を確認するには、有効なユーザ名とパスワードを使用してデバイスへの SSH 接続を作成し、デバイスの機能を含む hello メッセージを受信します。

```
Device:~ USER1$ ssh -s cisco1@172.16.167.175 -p 830 netconf
cisco1@172.16.167.175's password: cisco1

<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<capabilities>
<capability>urn:ietf:params:netconf:base:1.0</capability>
<capability>urn:ietf:params:netconf:base:1.1</capability>
<capability>urn:ietf:params:netconf:capability:writable-running:1.0</capability>
<capability>urn:ietf:params:netconf:capability:xpath:1.0</capability>
<capability>urn:ietf:params:netconf:capability:validate:1.0</capability>
<capability>urn:ietf:params:netconf:capability:validate:1.1</capability>
<capability>urn:ietf:params:netconf:capability:rollback-on-error:1.0</capability . . .
></capabilities>
<session-id>2870</session-id></hello>]]>]]>

Use < ^C > to exit
```

hello メッセージに対して正常な応答を受信すれば、NETCONF を使用する準備が整います。

RESTCONF 固有の前提条件

- RESTCONF とその使用方法に関する次の知識（RESTCONF を使用してサブスクリプションを作成する場合）。
- RESTCONF がデバイスで設定されている必要があります。
- RESTCONF RFC 8040に準拠した、正しい形式の Uniform Resource Identifier（URI）を送信する必要があります。

RESTCONF の有効化と検証

適切なクレデンシャルと次の URI を使用して、RESTCONF を検証します。

```
Operation: GET
Headers:
" Accept: application/yang-data.collection+json, application/yang-data+json,
application/yang-data.errors+json
" Content-Type: application/yang-data+json
Returned Output (omitted for breverity):
{
    "ietf-restconf:data": {
        "ietf-yang-library:modules-state": {
            "module": [
                {
                    "name": "ATM-FORUM-TC-MIB",
                    "revision": "",
                    "schema":
```

```
"https://10.85.116.28:443/restconf/tailf/modules/ATM-FORUM-TC-MIB",
                    "namespace": "urn:ietf:params:xml:ns:yang:smiv2:ATM-FORUM-TC-MIB"
                },
                {
                    "name": "ATM-MIB",
                    "revision": "1998-10-19",
                    "schema":
"https://10.85.116.28:443/restconf/tailf/modules/ATM-MIB/1998-10-19",
                    "namespace": "urn:ietf:params:xml:ns:yang:smiv2:ATM-MIB"
                },
                {
                    "name": "ATM-TC-MIB",
                    "revision": "1998-10-19",
                    "schema": "https://10.85.116.28:443/restconf/tailf/
..
<snip>
..
}
```

すべてのデバイス機能で前述の応答を受信すると、RESTCONF が正常に検証されます。

gRPC

- キー値 Google Protocol Buffers（GPB）エンコーディングを理解する gRPC コレクタを設定します。

モデル駆動型テレメトリの制約事項

- yang-push ストリームを使用している場合、選択における自動階層は、変更時サブスクリプション向けにサポートされません。つまり、リストを選択するときに、リストの子リストが自動的には含まれません。たとえば、サブスクライバでは、子リストごとにサブスクリプションを手動で作成する必要があります。
- データアクセスの許可チェックはサポートされていません。サブスクライバによって要求されたすべてのデータが送信されます。
- サブツリーフィルタはサポートされていません。サブツリーフィルタが指定された場合、サブスクリプションは無効としてマークされます。
- サブスクリプションパラメータの中で複数の受信者を定義することはサポートされていません。最初の受信者の宛先だけが試行されます。他の定義済みの受信者は無視されます。

gRPC 固有の制限事項

- デバイスと受信者間の Transport Layer Security（TLS）ベースの認証はサポートされていません。

yang-push 固有の制限

- サブスクリプションの Quality of Service（QoS）はサポートされていません。

モデル駆動型テレメトリについて

モデル駆動型テレメトリの概要

テレメトリは、自動の通信プロセスです。これにより、測定およびその他のデータがリモートポイントまたはアクセス不能なポイントで収集され、モニタ用の受信装置に送信されます。モデル駆動型テレメトリは、YANG モデル化されたデータをデータ コレクタにストリーミングするためのメカニズムを提供します。

アプリケーションでは、NETCONF、RESTCONF、または gRPC ネットワーク管理インターフェイス（gNMI）の各プロトコルを介した標準ベースの YANG データ モデルを使用して、必要とする特定のデータ項目をサブスクライブできます。サブスクリプションは CLI を使用して作成することもできます（設定済みサブスクリプションの場合）。

構造化データは、サブスクリプション基準とデータ タイプに基づき、定義されたパターンでまたは変更時にパブリッシュされます。

テレメトリ ロール

テレメトリを使用するシステムでは、さまざまなロールが関与します。このドキュメントでは、次のテレメトリ ロールを使用します。

- パブリッシャ：テレメトリ データを送信するネットワーク要素。
- 受信者：テレメトリ データを受信するネットワーク要素。コレクタとも呼ばれます。
- コントローラ：サブスクリプションを作成するがテレメトリ データを受信しないネットワーク要素。作成したサブスクリプションに関連付けられたテレメトリ データが受信者に送信されます。管理エージェントまたは管理エンティティとも呼ばれます。
- サブスクライバ：サブスクリプションを作成するネットワーク要素。技術的には、サブスクライバは受信者である必要はありませんが、このドキュメントの目的から見た場合はどちらも同じです。

サブスクリプションの概要

サブスクリプションは、テレメトリ ロール間の関連付けを作成する項目であり、ロール間で送信されるデータを定義します。

具体的には、サブスクリプションは、テレメトリ データの一部として要求される一連のデータを定義するために使用されます。たとえば、データがいつ必要か、データの書式設定の方法、また暗黙的でない場合は誰（どの受信者）がデータを受信するかを定義します。

サポートされているサブスクリプションの最大数はプラットフォームによって異なりますが、現在は 100 個のサブスクリプションがサポートされています。サブスクリプションは、設定済みか動的のいずれかにすることができ、トランスポート プロトコルの任意の組み合わせを使用できます。有効なすべての設定済みサブスクリプションをアクティブにするために同時に多数のサブスクリプションが動作している場合、サブスクリプションの数が多すぎると、アクティブなサブスクリプションを削除したときに、非アクティブであるが有効な設定済みサブスクリ

プションの1つが試行されます。定期的にトリガーされるサブスクリプション（デフォルトの最小値は100センチ秒）と、変更時にトリガーされるサブスクリプションがサポートされています。

サブスクリプションの設定では、NETCONF やその他のノースバウンド プログラマブル インターフェイス（RESTCONF、gNMI など） がサポートされています。

Cisco IOS XE システムのテレメトリで使用されるサブスクリプションには、動的サブスクリプションと設定済みサブスクリプションの 2 種類があります。

動的サブスクリプションは、パブリッシャに接続するクライアント（サブスクライバ）によって作成されるため、ダイヤルインと見なされます。設定済みサブスクリプションでは、パブリッシャは受信者への接続を開始し、その結果ダイヤルアウトと見なされます。

ダイヤルインおよびダイヤルアウトのモデル駆動型テレメトリ

モデル駆動型テレメトリには、ダイヤルインとダイヤルアウトの 2 種類があります。

表 *27:* ダイヤルインおよびダイヤルアウトのモデル駆動型テレメトリ

ダイヤルイン（動的）	ダイヤルアウト（静的または設定済み）
テレメトリの更新は、イニシエータ/サブスクライバに送信されます。	テレメトリの更新は、指定された受信者/コレクタに送信されます。
サブスクリプションの存続期間は、そのサブスクリプションを作成した接続（セッション）に結び付けられ、その存続期間中テレメトリの更新が送信されます。実行コンフィギュレーションでは変更は観察されません。	サブスクリプションは実行コンフィギュレーションの一部として作成されます。これは、コンフィギュレーションが削除されるまでデバイス設定として残ります。
ダイヤルイン サブスクリプションはリロード後に再起動する必要があります。これは、確立された接続またはセッションがステートフル スイッチオーバー時に kill されるためです。	ダイヤルアウト サブスクリプションはデバイス設定の一部として作成され、ステートフル スイッチオーバー後に自動的に受信者に再接続します。
サブスクリプション ID は、サブスクリプションの確立が成功したときに動的に生成されます。	サブスクリプション ID は固定であり、設定の一部としてデバイス上で設定されます。

データ ソースの仕様

サブスクリプション内のテレメトリ データのソースは、ストリームとフィルタを使用して指定されます。ここでのストリームとは、関連する一連のイベントを指します。RFC 5277 ではイベント ストリームを、いくつかの転送基準に一致する一連のイベント通知としてを定義しています。

通常は、ストリームからの一連のイベントはフィルタ処理されます。異なるストリーム タイプごとに異なるフィルタ タイプが使用されます。

Cisco IOS XE は、yang-push と yang-notif-native の 2 つのストリームをサポートしています。

更新の通知

サブスクリプションの一部として、データが必要になるタイミングを指定できます。ただし、これはストリームによって異なります。ストリームの中で変更またはイベントが発生した場合にのみデータを使用できるようにするストリームもあれば、変更発生時に、または定義済みの時間間隔でデータを使用できるようにするストリームもあります。

この「タイミング」指定の結果は、対象のテレメトリ データを送る一連の更新通知となります。データの送信方法は、パブリッシャと受信者間の接続に使用されるプロトコルによって異なります。

サブスクリプション識別子

サブスクリプションは 32 ビットの正の整数値で識別されます。設定済みサブスクリプションのサブスクリプション ID はコントローラによって設定され、動的サブスクリプションの場合はパブリッシャによって設定されます。

コントローラは、パブリッシャで作成された動的サブスクリプションとの競合を避けるために、設定済みサブスクリプションに使用する値を 0 ～ 2147483647 の範囲に制限する必要があります。動的サブスクリプションの ID 空間はグローバルです。つまり、独立して作成された動的サブスクリプションのサブスクリプション ID は重複しません。

サブスクリプション管理

管理操作の任意の形式を使用して、設定済みサブスクリプションの作成、削除、および変更を行うことができます。これには、CLI とネットワーク プロトコルの両方の管理操作が含まれます。

すべてのサブスクリプション（設定済みと動的）は、**show** コマンド、およびネットワーク プロトコル管理操作を使用して表示できます。

以下では、サポートされているストリームとエンコーディングについて説明します。入力としてのストリームは出力としてのプロトコルから独立していることを意図していますが、すべての組み合わせがサポートされているわけではありません。次の表で、サポートされている組み合わせについて説明します。

表 *28* : サポートされるプロトコルの組み合わせ

Transport Protocol	**NETCONF**		**gRPC**	
	ダイヤルイン	発信	ダイヤルイン	発信
Stream				
yang-push	対応	非対応	N/A	対応
yang-notif-native	対応	なし	N/A	非対応
Encodings	XML	N/A	N/A	キー値 Google Protocol Buffers（kvGPB）

テレメトリの *RPC* サポート

確立された NETCONF セッションで、YANG XML リモート プロシージャ コール（RPC）の送受信が行えます。

テレメトリには <establish-subscription> RPC と <delete-subscription> RPC がサポートされています。

<establish-subscription> RPC が送信されると、パブリッシャからの RPC 応答には <rpc-reply> メッセージと、結果ストリングを含む <subscription-result> 要素が含まれます。

次の表は、<rpc-reply> メッセージでの応答と、応答の理由を示しています。

結果文字列	RPC	原因
ok	<establish-subscription> <delete-subscription>	成功
error-no-such-subscription	<delete-subscription>	指定されたテンプレートは存在しません。
error-no-such-option	<establish-subscription>	要求されたサブスクリプションはサポートされていません。
error-insufficient-resources	<establish-subscription>	サブスクリプションは次の理由により作成できません。 • サブスクリプションが多すぎる。 • 要求されたデータの量が大きすぎる。 • 定期的なサブスクリプションの間隔が短すぎる。
error-other	<establish-subscription>	その他の何らかのエラーです。

動的サブスクリプションの制御

ここでは、動的サブスクリプションを作成および削除する方法について説明します。

動的サブスクリプションの作成

動的サブスクリプションは、パブリッシャに接続し、その接続内部のメカニズム（通常はリモート プロシージャ コール（RPC））を使用してサブスクリプション作成のための呼び出しを行うサブスクライバによって作成されます。サブスクリプションの存続期間は、サブスクライバとパブリッシャ間の接続の存続期間に制限され、テレメトリ データはそのサブスクライバ

にのみ送信されます。これらのサブスクリプションは、パブリッシャまたはサブスクライバのいずれかが再起動された場合は存続しません。動的サブスクリプションの作成にはインバンドの <establish-subscription> RPC を使用できます。<establish-subscription> RPC は、IETF テレメトリのサブスクライバからネットワークデバイスに送信されます。RPC では、stream、xpath-filter、および period の各フィールドが必須です。

定期的な動的サブスクリプション

次に、定期的なサブスクリプションの例を示します。

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
      <establish-subscription
          xmlns="urn:ietf:params:xml:ns:yang:ietf-event-notifications"
          xmlns:yp="urn:ietf:params:xml:ns:yang:ietf-yang-push">
        <stream>yp:yang-push</stream>
        <yp:xpath-filter>/mdt-oper:mdt-oper-data/mdt-subscriptions</yp:xpath-filter>
        <yp:period>1000</yp:period>
      </establish-subscription>
    </rpc>
```

変更時動的サブスクリプション

次に、NETCONF を介した変更時動的サブスクリプションの例を示します。

```
<establish-subscription xmlns="urn:ietf:params:xml:ns:yang:ietf-event-notifications"
xmlns:yp="urn:ietf:params:xml:ns:yang:ietf-yang-push">
    <stream>yp:yang-push</stream>

<yp:xpath-filter>/cdp-ios-xe-oper:cdp-neighbor-details/cdp-neighbor-detail</yp:xpath-filter>

    <yp:dampening-period>0</yp:dampening-period>
</establish-subscription>
```

動的サブスクリプションの削除

動的サブスクリプションを削除するには、インバンドの <delete subscription> RPC を使用し、トランスポート セッションを切断します。

<delete-subscription> RPC は、サブスクライバのみが発行でき、そのサブスクライバが所有するサブスクリプションのみを削除します。

親の NETCONF セッションが切断されると、サブスクリプションも削除されます。ネットワーク接続が中断された場合は、SSH/NETCONF セッションがタイムアウトしてその後にサブスクリプションが削除されるまで、多少の時間がかかることがあります。

NETCONF による動的サブスクリプションの作成および削除に使用する RPC は、イベント通知のカスタム サブスクリプション draft-ietf-netconf-subscribed-notifications-03 および YANG データストア プッシュ更新のサブスクライブ draft-ietf-netconf-yang-push-07 で定義されています。

NETCONF <delete-Subscription> RPC を使用したサブスクリプションの削除

次に、NETCONF を使用してサブスクリプションを削除する例を示します。

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <delete-subscription xmlns="urn:ietf:params:xml:ns:yang:ietf-event-notifications"
    xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
        <subscription-id>2147483650</subscription-id>
    </delete-subscription>
</rpc>
```

設定済みサブスクリプションの管理

ここでは、設定済みサブスクリプションを作成、変更、および削除する方法について説明します。

設定済みサブスクリプションの作成

設定済みサブスクリプションは、コントローラによるパブリッシャでの管理操作によって作成され、サブスクリプションによって定義されたテレメトリデータの受信者の指定が明示的に含まれています。これらのサブスクリプションは、パブリッシャの再起動後も持続します。

設定済みサブスクリプションは複数の受信者を使用して設定できますが、最初の有効な受信者のみが使用されます。受信者がすでに接続されている場合、または接続中の場合は、他の受信者への接続は試行されません。その受信者が削除されると、別の受信者が接続されます。

ここでは、設定済みサブスクリプションを作成するための RPC の例を示します。

定期的なサブスクリプション

次の RPC の例は、NETCONF を使用して定期的なサブスクリプションを作成し、60 秒ごとにテレメトリの更新を受信者に送信します。

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"><edit-config>
 <target>
  <running/>
 </target>
 <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <mdt-config-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-mdt-cfg">
   <mdt-subscription>
    <subscription-id>200</subscription-id>
    <base>
     <stream>yang-push</stream>
     <encoding>encode-kvgpb</encoding>
     <period>6000</period>
     <xpath>/memory-ios-xe-oper:memory-statistics/memory-statistic</xpath>
    </base>
    <mdt-receivers>
     <address>10.22.23.48</address>
     <port>57555</port>
     <protocol>grpc-tcp</protocol>
    </mdt-receivers>
   </mdt-subscription>
  </mdt-config-data>
 </config>
</edit-config>
</rpc>
```

次に、RESTCONF を使用して定期的なサブスクリプションを作成する RPC の例を示します。

```
URI:https://10.85.116.28:443/restconf/data/Cisco-IOS-XE-mdt-cfg:mdt-config-data
Headers:
application/yang-data.collection+json, application/yang-data+json,
application/yang-data.errors+json
Content-Type:
application/yang-data+json
BODY:
{
"mdt-config-data": {
 "mdt-subscription":[
 {
  "subscription-id": "102",
  "base": {
   "stream": "yang-push",
   "encoding": "encode-kvgpb",
                    "period": "6000",
   "xpath": "/memory-ios-xe-oper:memory-statistics/memory-statistic"
  }
        "mdt-receivers": {
            "address": "10.22.23.48"
            "port": "57555"
        }
 }
 ]
}
}
```

変更時サブスクリプション

次の RPC の例は、NETCONFを使用して変更時サブスクリプションを作成し、ターゲットデータベースに変更が生じた場合にのみ更新を送信します。

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"><edit-config>
 <target>
  <running/>
 </target>
 <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <mdt-config-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-mdt-cfg">
   <mdt-subscription>
    <subscription-id>200</subscription-id>
    <base>
     <stream>yang-push</stream>
     <encoding>encode-kvgpb</encoding>
     <no-synch-on-start>false</no-synch-on-start>
     <xpath>/cdp-ios-xe-oper:cdp-neighbor-details/cdp-neighbor-detail</xpath>
    </base>
    <mdt-receivers>
     <address>10.22.23.48</address>
     <port>57555</port>
     <protocol>grpc-tcp</protocol>
    </mdt-receivers>
   </mdt-subscription>
  </mdt-config-data>
 </config>
</edit-config>
</rpc>
```

次に、RESTCONF を使用して変更時サブスクリプションを作成する RPC の例を示します。

```
URI:
https://10.85.116.28:443/restconf/data/Cisco-IOS-XE-mdt-cfg:mdt-config-data
Headers:
```

```
application/yang-data.collection+json, application/yang-data+json,
application/yang-data.errors+json
Content-Type:
application/yang-data+json
BODY:
{
"mdt-config-data": {
 "mdt-subscription":[
 {
  "subscription-id": "102",
  "base": {
   "stream": "yang-push",
   "encoding": "encode-kvgpb",
                     "dampening period": "0",
   "xpath": "/cdp-ios-xe-oper:cdp-neighbor-details/cdp
                           -neighbor-detail "
  }
        "mdt-receivers": {
            "address": "10.22.23.48"
            "port": "57555"
        }
 }
 ]
}
}
```

設定済みサブスクリプションの変更

設定済みサブスクリプションを変更するには、次の 2 つの方法があります。

- NETCONF <edit-config> RPC などの管理プロトコル設定操作
- CLI（サブスクリプションの作成と同じ手順）

サブスクリプションの受信者はアドレスとポート番号によって識別されます。受信者を変更することはできません。受信者の特性（プロトコル、プロファイルなど）を変更するには、先に受診者を削除してから新しい受信者を作成する必要があります。

有効なサブスクリプションの有効な受信者設定が切断状態にあり、管理側で受信者への接続のセットアップ時に新しい試行を強制する場合は、同一の特性を持つ受信者を書き換える必要があります。

設定済みサブスクリプションの削除

CLI または管理操作を使用して、設定済みサブスクリプションを削除できます。**no telemetry ietf subscription** コマンドは、設定済みサブスクリプションを削除します。RPC を使用して設定済みサブスクリプションを削除することはできません。これらのサブスクリプションは、設定インターフェイスを介して削除されます。

CLI を使用したサブスクリプションの削除

```
Device# configure terminal
Device(config)# no telemetry ietf subscription 101
Device(config)# end
```

NETCONF を使用したサブスクリプションの削除

次の RPC の例は、設定済みサブスクリプションを削除する方法を示しています。

```
<edit-config>
   <target>
     <running/>
   </target>
   <config>
     <mdt-config-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-mdt-cfg">
       <mdt-subscription  operation="delete">
       <subscription-id>102</subscription-id>
      </mdt-subscription>
     </mdt-config-data>
   </config>
 </edit-config>
```

サブスクリプションのモニタリング

CLIおよび管理プロトコル操作を使用して、すべてのタイプのサブスクリプションを監視できます。

CLI

テレメトリのサブスクリプションに関する情報を表示するには、**show telemetry ietf subscription** コマンドを使用します。コマンドからの出力例を、次に示します。

```
Device#  show telemetry ietf subscription 2147483667 detail

Telemetry subscription detail:

  Subscription ID: 2147483667
  State: Valid
  Stream: yang-push
  Encoding: encode-xml
  Filter:
    Filter type: xpath
    XPath: /mdt-oper:mdt-oper-data/mdt-subscriptions
  Update policy:
    Update Trigger: periodic
    Period: 1000
  Notes:
```

NETCONF

次に、テレメトリのサブスクリプションに関する情報を表示するNETCONFメッセージを示します。

```
<get>
<filter>
<mdt-oper-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-mdt-oper">
<mdt-subscriptions/>
</mdt-oper-data>
</filter>
</get>
```

```
* Enter a NETCONF operation, end with an empty line
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="2">
  <data>
    <mdt-oper-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-mdt-oper">
      <mdt-subscriptions>
        <subscription-id>101</subscription-id>
        <base>
          <stream>yang-push</stream>
          <encoding>encode-kvgpb</encoding>
          <source-vrf>RED</source-vrf>
          <period>10000</period>
          <xpath>/ios:native/interface/Loopback[name="1"]</xpath>
        </base>
        <type>sub-type-static</type>
        <state>sub-state-valid</state>
        <comments/>
        <mdt-receivers>
          <address>5.22.22.45</address>
          <port>57500</port>
          <protocol>grpc-tcp</protocol>
          <state>rcvr-state-connecting</state>
          <comments/>
          <profile/>
          <last-state-change-time>1970-01-01T00:00:00+00:00</last-state-change-time>
        </mdt-receivers>
        <last-state-change-time>1970-01-01T00:00:00+00:00</last-state-change-time>
      </mdt-subscriptions>
      <mdt-subscriptions>
        <subscription-id>2147483648</subscription-id>
        <base>
          <stream>yang-push</stream>
          <encoding>encode-xml</encoding>
          <source-vrf/>
          <period>1000</period>

<xpath>/if:interfaces-state/interface[name="GigabitEthernet0/0"]/oper-status</xpath>
        </base>
        <type>sub-type-dynamic</type>
        <state>sub-state-valid</state>
        <comments/>
        <mdt-receivers>
          <address>5.22.22.45</address>
          <port>51259</port>
          <protocol>netconf</protocol>
          <state>rcvr-state-connected</state>
          <comments/>
          <profile/>
          <last-state-change-time>1970-01-01T00:00:00+00:00</last-state-change-time>
        </mdt-receivers>
        <last-state-change-time>1970-01-01T00:00:00+00:00</last-state-change-time>
      </mdt-subscriptions>
    </mdt-oper-data>
  </data>
</rpc-reply>
```

ストリーム

ストリームは、サブスクライブ可能な一連のイベントを定義します。ほぼすべてのイベントがこの一連のイベントとして有効です。ただし、各ストリームの定義に従い、すべてのイベント

の候補は何らかの形で関連しています。ここでは、サポートされているストリームについて説明します。

サポートされているストリームのセットを表示するには、管理プロトコル操作を使用して、mdt-streams コンテナにある Cisco-IOS-XE-mdt-oper モジュール（YANG モデル Cisco-IOS-XE-mdt-oper.yang からのもの）から streams テーブルを取得します。

次に、NETCONF を使用して、サポートされているストリームを取得する例を示します。

```
<get>
<filter>
<mdt-oper-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-mdt-oper">
<mdt-streams/>
</mdt-oper-data>
</filter>
</get>

* Enter a NETCONF operation, end with an empty line

<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="2">
  <data>
    <mdt-oper-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-mdt-oper">
      <mdt-streams>
        <stream>native</stream>
          <stream>yang-notif-native</stream>
          <stream>yang-push</stream>
      </mdt-streams>
    </mdt-oper-data>
  </data>
</rpc-reply>
```

この例は、native、yang-notif-native、yang-push の 3 つのストリームがサポートされていることを示しています。ストリーム native は汎用としては使用できず、無視できます。

（注） 現在のところ、サポートされているストリームのリストを返す CLI はありません。

YANG-push ストリーム

yang-push ストリームは、サポートされている YANG モデルにより記述される、構成データベース内と運用データベース内のデータです。このストリームは、ストリームの中で対象とするデータを指定するための XPath フィルタをサポートしており、XPath 式は対象のデータを定義する YANG モデルに基づきます。

このストリームの更新通知は、対象のサブスクリプションについて、データの変更時または固定間隔で送信される場合がありますが、両方に対応して送信されることはありません。現在存在しないデータのサブスクリプションは許可され、通常のサブスクリプションとして実行されます。

サポートされている唯一のターゲット データベースは「実行中」です。

変更時機能の決定

現在のところ、変更時サブスクリプションを使用し、サブスクライブ可能なデータのタイプについて YANG モデルの中で指定する手段はありません。変更時サブスクリプションを使用して、サブスクライブができないデータにサブスクライブしようとすると、失敗（動的）となるか、無効なサブスクリプション（設定済み）となります。

IETF ドラフトへの準拠

yang-push ストリームを使用するテレメトリは、テレメトリの IETF NETCONF ワーキング グループの初期ドラフトに基づいています。それらは次のとおりです。

- *イベント通知のカスタム サブスクリプション、バージョン 03*
- *YANG データストア プッシュ更新のサブスクライブ、バージョン 07*

ドラフトに記載されている次の機能はサポートされていません。

- サブツリー フィルタ
- アウトオブバンドの通知
- サポート対象として明示的に記載されていないすべてのサブスクリプション パラメータ

YANG-push の *XPath* フィルタ

サブスクライブ先の yang-push ストリーム内のデータセットは、XPath フィルタを使用して指定します。XPath 式には次の制限が適用されます。

- 単一のオブジェクトを指定する必要があります。このオブジェクトには、コンテナ、リーフ、リーフリスト、またはリストを使用できます。
- リストまたはコンテナでは、1 つのエントリを指定するためのキーを持たせることができます。サポートされているキー指定の構文は次のとおりです。

  ```
  [{key name}={key value}]
  ```

  複合キーを使用するには、複数のキー指定を使用します。キーの名前と値は正確である必要があります。範囲やワイルドカードによる値はサポートされていません。
- 単一のサブスクリプションで複数のオブジェクトをサポートできるように、結合演算子（|）を使用できます。

YANG-push の定期パブリケーション

定期的なサブスクリプションでは、サブスクライブ対象情報による最初のプッシュ更新は即時に送信されます。ただしデバイスがビジー状態であったりネットワークが混雑していたりすると遅延することがあります。次に更新は、設定された定期タイマーの満了時に送信されます。たとえば、期間を 10 分と設定すると、サブスクリプションの作成直後に最初の更新が送信され、その後は 10 分おきに送信されます。

期間は、定期的なプッシュ更新間のセンチ秒（1/100 秒）単位の時間です。期間が 1000 であれば、サブスクライブ対象情報の更新は 10 秒ごとになります。設定できる最小の期間間隔は 100

（つまり 1 秒）です。デフォルト値はありません。この値は、動的サブスクリプションの場合は <establish-subscription> RPC で明示的に設定する必要があり、設定済みサブスクリプションの場合は設定で明示的に設定する必要があります。

定期的な更新には、サポートされているすべてのトランスポートプロトコルに関連するサブスクライブ対象のデータ要素またはテーブルのフルコピーが含まれています。

定期的なサブスクリプションを使用して空のデータをサブスクライブすると、要求された期間で空の更新通知が送信されます。データが存在するようになると、次の期間の値が通常の更新通知として送信されます。

YANG-push の変更時パブリケーション

変更時サブスクリプションを作成する場合は、ダンプニング期間がないことを示すためにダンプニング期間を 0 に設定する必要があります。その他の値はサポートされていません。

変更時サブスクリプションでは、最初のプッシュ更新は、サブスクライブされたデータのセット全体です（IETF の文書で定義されている初期同期）。これは制御できません。以降の更新は、データが変更され、変更後のデータのみで構成されている場合に送信されます。ただし、変更とみなされる最小のデータ分解能は行です。したがって、変更時サブスクリプションが行内のリーフに対するものである場合、その行のいずれかの項目が変更されると、更新通知が送信されます。更新通知の正確な内容はトランスポート プロトコルによって異なります。

また、変更時サブスクリプションは階層状ではありません。つまり、子コンテナを持つコンテナにサブスクライブしても、子コンテナ内の変更はサブスクリプションには認識されません。

現在存在しないデータのサブスクリプションは許可され、通常のサブスクリプションとして実行されます。初期同期更新通知は空であり、データが存在するまでそれ以上更新されません。

yang-notif-native ストリーム

yang-notif-native ストリームは、パブリッシャ内の任意の YANG 通知であり、通知の元のイベントソースで Cisco XE のネイティブのテクノロジーが使用されています。このストリームは、対象となる通知を指定する XPath フィルタもサポートしています。このストリームの更新通知は、通知の目的であるイベントが発生した場合にのみ送信されます。

このストリームは変更時サブスクリプションのみをサポートしているため、ダンプニング間隔として値 0 を指定する必要があります。

（注） 現在のところ、このストリームは Google リモート プロシージャ コール（gRPC）経由ではサポートされていません。

yang-notif-native の *XPath* フィルタ

サブスクライブ先の yang-notif-native ストリーム内のデータセットは、XPath フィルタを使用して指定します。XPath 式には次の制限が適用されます。

- 単一のオブジェクトを指定する必要があります。このオブジェクトには、コンテナ、リーフ、リーフリスト、またはリストを使用できます。

- YANG通知全体を指定する必要があります。属性のフィルタ処理はサポートされていません。
- 単一のサブスクリプションで複数のオブジェクトをサポートできるように、結合演算子（|）を使用できます。

トランスポート プロトコル

データの送信方法は、パブリッシャと受信者間の接続に使用されるプロトコルによって決まります。このプロトコルはトランスポートプロトコルと呼ばれ、設定済みサブスクリプションの管理プロトコルからは独立しています。トランスポートプロトコルは、データのエンコーディング（XML、Google Protocol Buffers（GPB）など）と更新通知自体の形式に影響を与えます。

（注） また、選択したストリームも更新通知の形式に影響を与える場合があります。

サポートされているトランスポート プロトコルは、NETCONF と gRPC です。

NETCONF プロトコル

NETCONFプロトコルは、動的サブスクリプションのトランスポートにのみ使用でき、yang-pushストリームと yang-notif-native ストリームで使用できます。

NETCONF をトランスポート プロトコルとして使用する場合は、次の3つの更新通知形式が使用されます。

- サブスクリプションで yang-push ストリームが使用されていて、定期的な場合、または、初期同期更新通知が変更時サブスクリプションで送信される場合。
- サブスクリプションで yang-push ストリームが使用されていて、初期同期更新通知以外の変更時サブスクリプションの場合。
- サブスクリプションで yang-notif-native ストリームが使用されている場合。

yang-push 形式

この形式は更新通知の2つの形式を定義します。その際、draft-ietf-netconf-yang-push-07 で定義されている XML エンコーディングを使用して yang-push ストリームが NETCONF を介してトランスポートとして送信されます。詳細については、IETF ドラフトの3.7項を参照してください。

yang-notif-native 形式

ソース ストリームが yang-notif-native の場合、NETCONF を介して XML でエンコードされるときの更新通知の形式はRFC 7950によって定義されています。詳細については、RFCの7.16.2項を参照してください。

yang-push ストリームの形式とは異なり、サブスクリプション ID は更新通知にはありません。

gRPC プロトコル

gRPC プロトコルは、設定済みサブスクリプションのトランスポートに対してのみ使用でき、yang-push ストリームでのみ使用できます。gRPC トランスポート プロトコルでは kvGPB エンコーディングのみがサポートされています

gRPCプロトコルに基づく受信者の接続の再試行（指数バックオフ）がサポートされています。

proto ファイルで定義されたテレメトリメッセージについては、 mdt_grpc_dialout.proto および telemetry.proto を参照してください。

テレメトリにおけるハイ アベイラビリティ

テレメトリの動的な接続は、アクティブなスイッチかスイッチ スタック内のメンバーへの SSH、またはハイ アベイラビリティ対応デバイスでのアクティブなルートプロセッサへの SSH を介して NETCONF セッションで確立されます。切り替え後は、テレメトリのサブスクリプションを伝送する NETCONF セッションを含め、暗号を使用するすべてのセッションを破棄し、再確立する必要があります。また、スイッチオーバー後にすべての動的サブスクリプションを再作成する必要もあります。

gRPC ダイヤルアウト サブスクリプションは、アクティブなスイッチまたはスタック メンバの実行コンフィギュレーションの一部としてデバイスに設定されます。スイッチオーバーが発生すると、テレメトリ受信者への既存の接続が切断され、再接続されます（受信者へのルートが残っている限り）。サブスクリプションを再設定する必要はありません。

（注） デバイスのリロード時には、サブスクリプションの設定をデバイスのスタートアップ コンフィギュレーションに同期させる必要があります。これにより、デバイスの再起動後もサブスクリプション設定がデバイス上にそのまま残ります。必要なプロセスが起動して実行されると、デバイスはテレメトリ受信者への接続を試行し、通常の動作を再開します。

サンプルのモデル駆動型テレメトリ RPC

設定済みサブスクリプションの管理

変更時サブスクリプションをサポートしている YANG モデルのリストを表示するには、**show platform software ndbman switch** {*switch-number* | **active** | **standby**} **models** コマンドを使用します。

（注） 現在のところ、設定済みサブスクリプションの管理に使用できるのは gRPC プロトコルのみです。

手順の概要

1. **enable**

2. **configure terminal**
3. **telemetry ietf subscription** *id*
4. **stream yang-push**
5. **filter xpath** *path*
6. **update-policy** {**on-change** | **periodic**} *period*
7. **encoding encode-kvgpb**
8. **source-vrf** *vrf-id*
9. **source-address** *source-address*
10. **receiver ip address** *ip-address receiver-port* **protocol** *protocol* **profile** *name*
11. **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）。
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	**telemetry ietf subscription** *id* 例： `Device(config)# telemetry ietf subscription 101`	テレメトリのサブスクリプションを作成し、テレメトリサブスクリプション モードを開始します。
ステップ 4	**stream yang-push** 例： `Device(config-mdt-subs)# stream yang-push`	サブスクリプションのストリームを設定します。
ステップ 5	**filter xpath** *path* 例： `Device(config-mdt-subs)# filter xpath /memory-ios-xe-oper:memory-statistics/memory-statistic`	サブスクリプションの XPath フィルタを指定します。
ステップ 6	**update-policy** {**on-change**	**periodic**} *period* 例： `Device(config-mdt-subs)# update-policy periodic 6000`	サブスクリプションの定期的な更新ポリシーを設定します。
ステップ 7	**encoding encode-kvgpb** 例： `Device(config-mdt-subs)# encoding encode-kvgpb`	kvGPB エンコードを指定します。

	コマンドまたはアクション	目的
ステップ 8	**source-vrf** *vrf-id* 例： `Device(config-mdt-subs)# source-address Mgmt-intf`	ソースの VRF インスタンスを設定します。
ステップ 9	**source-address** *source-address* 例： `Device(config-mdt-subs)# source-vrf 192.0.2.1`	送信元アドレスを設定します。
ステップ 10	**receiver ip address** *ip-address receiver-port* **protocol** *protocol* **profile** *name* 例： `Device(config-mdt-subs)# receiver ip address 10.28.35.45 57555 protocol grpc-tcp`	通知の受信者の IP アドレス、プロトコル、およびプロファイルを設定します。
ステップ 11	**end** 例： `Device(config-mdt-subs)# end`	テレメトリサブスクリプションの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。

gRPC の変更時サブスクリプションの設定

手順の概要

1. **enable**
2. **configure terminal**
3. **telemetry ietf subscription** *id*
4. **stream yang-push**
5. **filter xpath** *path*
6. **update-policy** {**on-change** | **periodic** *period*}
7. **encoding encode-kvgpb**
8. **receiver ip address** *ip-address receiver-port* **protocol** *protocol* **profile** *name*
9. **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）。
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。

	コマンドまたはアクション	目的
ステップ 3	**telemetry ietf subscription** *id* 例： `Device(config)# telemetry ietf subscription 8`	テレメトリのサブスクリプションを作成し、テレメトリサブスクリプション モードを開始します。
ステップ 4	**stream yang-push** 例： `Device(config-mdt-subs)# stream yang-push`	サブスクリプションのストリームを設定します。
ステップ 5	**filter xpath** *path* 例： `Device(config-mdt-subs)# filter xpath /iosxe-oper:ios-oper-db/hwidb-table`	サブスクリプションの XPath フィルタを指定します。
ステップ 6	**update-policy** {**on-change** | **periodic** *period*} 例： `Device(config-mdt-subs)# update-policy on-change`	サブスクリプションの変更時更新ポリシーを設定します。
ステップ 7	**encoding encode-kvgpb** 例： `Device(config-mdt-subs)# encoding encode-kvgpb`	kvGPB エンコードを指定します。
ステップ 8	**receiver ip address** *ip-address receiver-port* **protocol** *protocol* **profile** *name* 例： `Device(config-mdt-subs)# receiver ip address 10.22.22.45 45000 protocol grpc_tls profile secure_profile`	通知の受信者の IP アドレス、プロトコル、およびプロファイルを設定します。
ステップ 9	**end** 例： `Device(config-mdt-subs)# end`	テレメトリサブスクリプションの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。

応答コードの受信

サブスクリプションが正常に作成されると、デバイスはサブスクリプション結果 notif-bis:ok およびサブスクリプション ID で応答します。次に、動的サブスクリプションの応答 RPC メッセージの例を示します。

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
<subscription-result xmlns="urn:ietf:params:xml:ns:yang:ietf-event-notifications"
xmlns:notif-bis="urn:ietf:params:xml:ns:yang:ietf-event-notifications">notif-bis:
ok</subscription-result>
<subscription-id
xmlns="urn:ietf:params:xml:ns:yang:ietf-event-notifications">2147484201</subscription-id>
</rpc-reply>
```

サブスクリプションのプッシュ更新の受信

デバイスからプッシュされるサブスクリプション更新はXML RPC形式であり、それらが作成された同じNETCONFセッションにより送信されます。サブスクライブ対象情報の要素またはツリーはdatastore-contents-xmlタグ内で返されます。次に示すのは、サブスクライブ対象情報を提供するサンプルRPCメッセージです。

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
    <eventTime>2017-05-09T21:34:51.74Z</eventTime>
    <push-update xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">
        <subscription-id>2147483650</subscription-id>
        <datastore-contents-xml>
            <cpu-usage
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-process-cpu-oper"><cpu-utilization>
            <five-minutes>5</five-minutes></cpu-utilization></cpu-usage>
        </datastore-contents-xml>
    </push-update>
</notification>
```

サブスクリプションが行われる情報要素が空である場合、またはそれが動的（名前付きアクセス リストなど）であり存在しない場合、定期更新は空になり、自己終結datastore-contents-xmlタグを持つことになります。次に示すのは、定期更新が空であるサンプルRPCメッセージです。

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
    <eventTime>2017-05-09T21:34:09.74Z</eventTime>
    <push-update xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">
        <subscription-id>2147483649</subscription-id>
        <datastore-contents-xml />
    </push-update>
</notification>
```

サブスクリプションの詳細の取得

現在のサブスクリプションの一覧を取得するには、<get> RPCをCisco-IOS-XE-mdt-operモデルに送信します。現在のサブスクリプションの一覧を表示するには、**show telemetry ietf subscription**コマンドも使用できます。

次に、<get> RPCメッセージの例を示します。

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter>
      <mdt-oper-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-mdt-oper">
        <mdt-subscriptions/>
      </mdt-oper-data>
    </filter>
  </get>
</rpc>
```

次に、RPC応答の例を示します。

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <data>
    <mdt-oper-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-mdt-oper">
      <mdt-subscriptions>
        <subscription-id>2147485164</subscription-id>
        <base>
          <stream>yang-push</stream>
          <encoding>encode-xml</encoding>
          <period>100</period>
          <xpath>/ios:native/router/ios-rip:rip/ios-rip:version</xpath>
        </base>
        <type>sub-type-dynamic</type>
        <state>sub-state-valid</state>
        <comments/>
        <updates-in>0</updates-in>
        <updates-dampened>0</updates-dampened>
        <updates-dropped>0</updates-dropped>
      </mdt-subscriptions>
    </mdt-oper-data>
  </data>
</rpc-reply>
```

次に、**show telemetry ietf subscription dynamic brief** コマンドの出力例を示します。

```
Device# show telemetry ietf subscription dynamic brief

Telemetry subscription brief

  ID               Type        State       Filter type
  ------------------------------------------------------
  2147483667       Dynamic     Valid       xpath
  2147483668       Dynamic     Valid       xpath
  2147483669       Dynamic     Valid       xpath
```

次に、**show telemetry ietf subscription** *subscription-ID***detail** コマンドの出力例を示します。

```
Device# show telemetry ietf subscription 2147483667 detail

Telemetry subscription detail:

  Subscription ID: 2147483667
  State: Valid
  Stream: yang-push
  Encoding: encode-xml
  Filter:
    Filter type: xpath
    XPath: /mdt-oper:mdt-oper-data/mdt-subscriptions
  Update policy:
    Update Trigger: periodic
    Period: 1000
  Notes:
```

次に、**show telemetry ietf subscription all detail** コマンドの出力例を示します。

```
Device# show telemetry ietf subscription all detail

Telemetry subscription detail:
```

```
  Subscription ID: 101
  Type: Configured
  State: Valid
  Stream: yang-push
  Encoding: encode-kvgpb
  Filter:
    Filter type: xpath
    XPath: /iosxe-oper:ios-oper-db/hwidb-table
  Update policy:
    Update Trigger: on-change
    Synch on start: Yes
    Dampening period: 0
  Notes:
```

RESTCONF を使用したサブスクリプションの詳細の取得

```
Subscription details can also be retrieved through a RESTCONF GET request to the
Cisco-IOS-XE-mdt-oper database:
URI:
https://10.85.116.28:443/restconf/data/Cisco-IOS-XE-mdt-oper:
mdt-oper-data/mdt-subscriptions
Headers:
application/yang-data.collection+json, application/yang-data+json,
application/yang-data.errors+json
Content-Type:
application/yang-data+json
Returned output:
{
  "Cisco-IOS-XE-mdt-oper:mdt-subscriptions": [
    {
      "subscription-id": 101,
      "base": {
        "stream": "yang-push",
        "encoding": "encode-kvgpb",
        "source-vrf": "",
        "no-synch-on-start": false,
        "xpath": "/iosxe-oper:ios-oper-db/hwidb-table"
      },
      "type": "sub-type-static",
      "state": "sub-state-valid",
      "comments": "",
      "updates-in": "0",
      "updates-dampened": "0",
      "updates-dropped": "0",
      "mdt-receivers": [
        {
          "address": "5.28.35.35",
          "port": 57555,
          "protocol": "grpc-tcp",
          "state": "rcvr-state-connecting",
          "comments": "Connection retries in progress",
          "profile": ""
        }
      ]
    }
  ]
}
```

モデル駆動型テレメトリに関するその他の参考資料

関連資料

関連項目	マニュアル タイトル
NETCONF-YANG パッチ	https://tools.ietf.org/wg/netconf/draft-ietf-netconf-yang-patch/
YANG エクスプローラ	https://github.com/CiscoDevNet/yang-explorer

標準および RFC

標準/RFC	タイトル
イベント通知のカスタム サブスクリプション *draft-ietf-netconf-subscribed-notifications-03*	https://tools.ietf.org/id/draft-ietf-netconf-subscribed-notifications-03.txt
イベント通知の *NETCONF* サポート	draft-ietf-netconf-netconf-event-notifications-01
RFC 5277	NETCONF イベント通知
RFC 6241	ネットワーク設定プロトコル（NETCONF）
RFC 7950	YANG 1.1 データ モデリング言語
RFC 8040	RESTCONF プロトコル
イベント通知への登録	draft-ietf-netconf-rfc5277bis-01
YANG データストア プッシュのサブスクライブ	draft-ietf-netconf-yang-push-04
YANG データストア プッシュ更新のサブスクライブ *draft-ietf-netconf-yang-push-07*	https://tools.ietf.org/id/draft-ietf-netconf-yang-push-07.txt

シスコのテクニカル サポート

説明	Link
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

モデル駆動型テレメトリの機能情報

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 *29 :* モデル駆動型テレメトリの機能情報

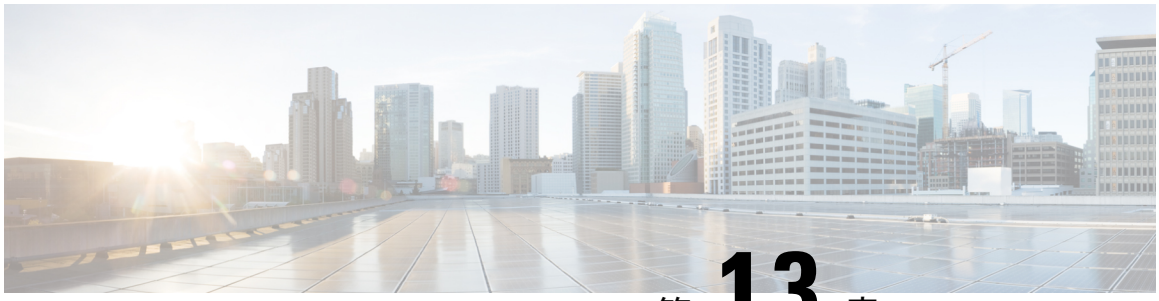
機能名	リリース	機能情報
モデル駆動型テレメトリ NETCONF ダイヤルイン	Cisco IOS XE Everest 16.6.1	モデル駆動型テレメトリでは、ネットワーク デバイスからサブスクライバに、リアルタイムの設定や運用状態の情報を継続的にストリームすることができます。 • Cisco Catalyst 3650 シリーズ スイッチ • Cisco Catalyst 3850 シリーズ スイッチ • Cisco Catalyst 9300 シリーズ スイッチ • Cisco Catalyst 9500 シリーズ スイッチ
	Cisco IOS XE Everest 16.6.2	• Cisco Catalyst 9400 シリーズ スイッチ
	Cisco IOS XE Fuji 16.7.1	• Cisco 4000 シリーズ サービス統合型ルータ • Cisco ASR 1001-HX シリーズ アグリゲーション サービス ルータ • Cisco ASR 1001-X シリーズ アグリゲーション サービス ルータ • Cisco ASR 1002-HX シリーズ アグリゲーション サービス ルータ • Cisco ASR 1002-X シリーズ アグリゲーション サービス ルータ
	Cisco IOS XE Fuji 16.8.1	• Cisco ASR 1000 RP2 および RP3 シリーズ アグリゲーション サービス ルータ

機能名	リリース	機能情報
	Cisco IOS XE Fuji 16.8.1a	• Cisco Catalyst 9500 ハイ パフォーマンス シリーズ スイッチ
	Cisco IOS XE Fuji 16.9.1	• 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 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 ジブラルタル 16.9.2	Cisco Catalyst 9200 シリーズ スイッチ
	Cisco IOS XE ジブラルタル 16.10.1	• Cisco IR1101 耐環境性能サービス統合型ルータ • Cisco クラウド サービス ルータ 1000v • Cisco Network Convergence System 520 シリーズ

機能名	リリース	機能情報
モデル駆動型テレメトリ gRPC ダイヤルアウト	Cisco IOS XE Fuji 16.10.1	

機能名	リリース	機能情報
		設置済みサブスクリプションでは、パブリッシャが受信者への接続を開始し、それらの接続はダイヤルアウトと見なされます。 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 IR1101 耐環境性能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 1000 シリーズ アグリゲーション サービス 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 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 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 Cloud Services Router 1000V シリーズ

機能名	リリース	機能情報
		• Cisco Network Convergence System 4200 シリーズ • Cisco Network Convergence System 520 シリーズ



第 13 章

In-Service Model Update

このモジュールでは、In-Service Model Update によりデバイス上の YANG データ モデルを更新する方法を説明します。

In-Service Model Update の制約事項

- ハイ アベイラビリティまたは In-Service Software Upgrade（ISSU）はサポートされていません。スイッチオーバーの後、ユーザはスタンバイ デバイスにソフトウェア メンテナンス アップデート（SMU）をインストールする必要があります。

In-Service Model Update について

In-Service Model Update の概要

サービス中モデル更新プログラムは、既存のデータモデルに新しいデータ モデルまたは拡張機能を追加します。サービス中モデル更新プログラムは、リリース サイクル外の YANG モデルの拡張機能を提供します。更新プログラム パッケージはすべての既存のモデルの上位セットです。これには、更新された YANG モデルを始めとするすべての既存モデルが含まれています。

データ モデル インフラストラクチャは、Cisco IOS XE デバイス用の YANG モデル定義管理インターフェイスを実装します。データ モデル インフラストラクチャは、Cisco IOS XE デバイスからノースバウンドに NETCONF インターフェイスを公開します。サポートされているデータ モデルには、IETF などの業界標準モデルと、Cisco IOS XE デバイス固有のモデルが含まれます。

In-Service Model Update によって提供される機能は、その後の Cisco IOS XE ソフトウェア メンテナンス リリースに統合されます。データ モデル更新プログラム パッケージは、シスコ ソフトウェア ダウンロード センター からダウンロードできます。

In-Service Model Update パッケージの互換性

更新パッケージは、リリース単位で作成され、プラットフォームに固有になります。たとえば、Cisco ASR 1000 シリーズ アグリゲーション サービス ルータの更新パッケージを Cisco CSR 1000V シリーズ クラウド サービス ルータにインストールすることはできません。同様に、Cisco IOS XE Fuji 16.7.1 用に作成された更新パッケージを、Cisco IOS XE Everest 16.5.2 バージョンを実行しているデバイスに適用することはできません。

更新プログラム パッケージのすべてのコンテンツは、将来のメインライン リリースまたはメンテナンス リリースのイメージの一部になります。イメージとプラットフォームのバージョンは、パッケージの追加およびアクティブ化の際に、In-Service Model Update コマンドによってチェックされます。イメージまたはプラットフォームの不一致が発生すると、パッケージのインストールが失敗します。

更新プログラム パッケージの命名規則

In-Service Model Update は、.bin ファイルとしてパッケージ化されています。このファイルには、特定のリリースおよびプラットフォームのすべての更新プログラムと、Readme ファイルが含まれています。これらのファイルにはリリース日があり、追加モデルの更新をともなって定期的に更新されます。

データ モデルの更新プログラム パッケージの命名規則は、次の形式に従っています。プラットフォームの種類-ライセンス レベル.リリース バージョン.DDTS ID-ファイル。次に、データ モデル更新ファイルの例を示します。

- isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin
- asr1000-universalk9.2017-08-23_17.48.0.CSCxxxxxxx.SSA.dmp.bin

Readme ファイルは、次の情報を提供します。

- データ モデルのアクティブ化または非アクティブ化中に表示されるコンソール メッセージおよびエラー メッセージ
- データ モデルのインストールによる影響
- 副作用と考えられる回避策
- In-Service Model Update によって影響を受けるパッケージ
- リスタートのタイプ

更新プログラム パッケージのインストール

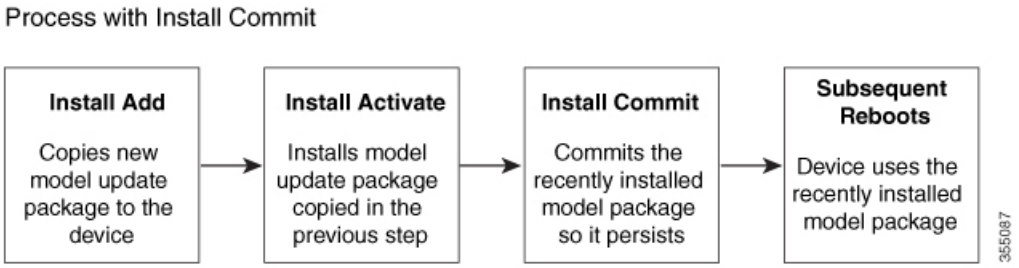
デバイスに In-Service Model Update パッケージをインストールするには、**install add**、 **install activate**、および **install commit** コマンドを特権 EXEC モードで使用します。

install add コマンドは、更新パッケージをリモートの場所からデバイスにコピーします。パッケージをコピーするには他の方法も使用できますが、その場合も、インストールしたプログラムを動作させるために **install add** コマンドを有効化する必要があります。 **install activate** コマンドを動作させるには、パッケージをデバイスのブートフラッシュで使用可能にする必要があります。**install commit** コマンドを有効化して、更新プログラムをリロード全体にわたって確定します。

更新プログラムをインストールすると、以前にインストールされたデータ モデルがある場合、それは置き換えられます。デバイスには常に、1 つの更新プログラムのみがインストールされます。データ モデル パッケージには、すべての更新された YANG モデルと、以前にデバイスにインストールされたすべての既存 YANG モデルが含まれています。

次のフロー チャートでは、モデル更新プログラム パッケージの動作を説明します。

図 *5 :* モデル更新プログラム パッケージのコミット



パッケージをアクティブ化する際にNETCONF-YANGが有効化されていると、NETCONFプロセスがリスタートされます。すべてのアクティブなNETCONFセッションは、パッケージのアクティブ化中に破棄されます。パッケージの検証中にエラーが発生すると、アクティブ化プロセスは中止されます。

更新プログラム パッケージの非アクティブ化

更新パッケージを非アクティブ化するには、**install deactivate** コマンドを使用します。変更を確定するには、**install commit** コマンドを有効化します。

表 *30 :* モデル更新プログラム パッケージの非アクティブ化

操作	使用コマンド
パッケージの削除	**install remove** コマンドを使用します。 （注） パッケージを削除する前に非アクティブ化します。

操作	使用コマンド
パッケージの非アクティブ化	**install deactivate** コマンドを使用し、その後に **install commit** コマンドを使用します。 （注） **install commit** コマンドの使用が必要なのは、モデルパッケージの非アクティブ化をリロード全体にわたって確定するためです。非アクティブ化がコミットされていないと、その後にパッケージを削除しようとしても失敗します。

更新プログラムを非アクティブ化する際に、2つ以上のモデル更新プログラム パッケージがインストールされている場合、最も最近コミットされたモデル更新プログラム パッケージがデバイスによって使用されるモデルパッケージになります。以前にコミットされたその他のモデルパッケージがない場合、標準的なイメージとともに含まれているベース バージョンのデータモデルが使用されるようになります。

更新プログラム パッケージのロールバック

ロールバックは、デバイスを更新前の動作状態に戻すメカニズムを提供します。ロールバック後は、変更が表示されるようになる前に NETCONF-YANG プロセスが再始動します。

更新は、**install rollback** コマンドを使用して、基本バージョン、最終コミット バージョン、または既知のコミット ID までロールバックできます。

In-Service Model Update の管理方法

更新プログラム パッケージの管理

手順の概要

1. **enable**
2. **install add file tftp:** *filename*
3. **install activate file bootflash:** *filename*
4. **install commit**
5. **install deactivate file bootflash:** *filename*
6. **install commit**
7. **install rollback to** {**base** | **committed** | **id** *commit-ID*}
8. **install remove** {**file bootflash:** *filename* | **inactive**}
9. **show install summary**

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）。

ステップ 2	**install add file tftp:** *filename*
	例：
	`Device# install add file tftp://172.16.0.1//tftpboot/folder1/ isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin`
	`Device# install add file tftp://172.16.0.1//tftpboot/folder1/ asr1000-universalk9.2017-08-23_17.48.0.CSCxxxxxxx.SSA.dmp.bin`
		リモート ロケーションから（FTP、TFTP 経由で）デバイスにモデル更新プログラム パッケージをコピーし、プラットフォームとイメージのバージョンの互換性チェックを実行します。
		• 他の方法を使用してリモートの場所からデバイスに更新パッケージをコピーすることもできます。ただし、その場合もパッケージをアクティブにする前に **install add** コマンドを実行する必要があります。

ステップ 3	**install activate file bootflash:** *filename*
	例：
	`Device# install activate file bootflash: isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin`
	`Device# install activate file bootflash: asr1000-universalk9.2017-08-23_17.48.0.CSCxxxxxxx.SSA.dmp.bin`
		更新パッケージが **install add** コマンドにより追加されていることを確認し、NETCONF プロセスを再開します。
		• 更新パッケージをアクティブにする前に **install add** 操作を実行します。

ステップ 4	**install commit** 例： `Device# install commit`	リロードが繰り返されても持続する変更を行います。 • NETCONF プロセスは再開されません。

ステップ 5	**install deactivate file bootflash:** *filename*
	例：
	`Device# install deactivate file bootflash: isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin`
	`Device# install deactivate file bootflash: asr1000-universalk9.2017-08-23_17.48.0.CSCxxxxxxx.SSA.dmp.bin`
		指定された更新プログラムパッケージを非アクティブにして、NETCONF プロセスを再開します。

ステップ 6	**install commit** 例： `Device# install commit`	リロードが繰り返されても持続する変更を行います。 • NETCONF プロセスは再開されません。
ステップ 7	**install rollback to** {**base** | **committed** | **id** *commit-ID*} 例： `Device# install rollback to base`	更新を基本バージョン、最後にコミットしたバージョン、または既知のコミット ID にロールバックし、NETCONF プロセスを再起動します。 • *commit-id* 引数の有効な値は 1 ～ 4294967295 です。

	コマンドまたはアクション	目的
		• データ モデル更新の古いバージョンが使用可能です。

ステップ 8	**install remove** {**file bootflash:** *filename* | **inactive**}
	例：
	`Device# install remove file bootflash: isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin`
	`Device# install remove file bootflash: asr1000-universalk9.2017-08-23_17.48.0.CSCxxxxxxx.SSA.dmp.bin`
		指定された更新プログラム パッケージをブートフラッシュから削除します。
		• パッケージは削除する前に非アクティブにする必要があります。

ステップ 9	**show install summary** 例： `Device# show install summary`	アクティブパッケージに関する情報を表示します。 • このコマンドの出力は、設定されている **install** コマンドに応じて変化します。

In-Service Model Update の設定例

例：更新プログラム パッケージの管理

次の例で使用しているのは、Cisco 4000 シリーズ サービス統合型ルータのサンプル イメージです。

次の例では、モデル更新プログラム パッケージ ファイルの追加方法を示しています。

```
Device# install add file tftp://172.16.0.1//tftpboot/folder1/
isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin

install_add: START Sun Feb 26 05:57:04 UTC 2017
Downloading file
tftp://172.16.0.1//tftpboot/folder1/isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin
Finished downloading file
tftp://172.16.0.1//tftpboot/folder1/isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin
to bootflash:isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin
SUCCESS: install_add /bootflash/isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin
Sun Feb 26 05:57:22 UTC 2017
Device#
```

次の例で使用しているのは、Cisco ASR1000 シリーズ アグリゲーション サービス ルータのサンプル イメージです。

次の例では、モデル更新プログラム パッケージ ファイルの追加方法を示しています。

```
Device# install add file tftp://172.16.0.1//tftpboot/folder1/
asr1000-universalk9.2017-08-23_17.48.0.CSCxxxxxxx.SSA.dmp.bin

install_add: START Sun Feb 26 05:57:04 UTC 2017
Downloading file
tftp://172.16.0.1//tftpboot/folder1/asr1000-universalk9.2017-08-23_17.48.0.CSCxxxxxxx.SSA.dmp.bin
Finished downloading file
```

```
tftp://172.16.0.1//tftpboot/folder1/asr1000-universalk9.2017-08-23_17.48.0.CSCxxxxxxx.SSA.dmp.bin
to bootflash: asr1000-universalk9.2017-08-23_17.48.0.CSCxxxxxxx.SSA.dmp.bin
SUCCESS: install_add
/bootflash/asr1000-universalk9.2017-08-23_17.48.0.CSCxxxxxxx.SSA.dmp.bin
Sun Feb 26 05:57:22 UTC 2017
Device#
```

次に、更新パッケージ ファイルをデバイスに追加した後の **show install summary** コマンドの出力例を示します。

```
Device# show install summary

Active Packages:
No packages
Inactive Packages:
bootflash: isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin
Committed Packages:
No packages
Uncommitted Packages:
No packages
Device#
```

次の例では、追加された更新プログラム パッケージ ファイルをアクティブにする方法を示しています。

```
Device# install activate file bootflash:
isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin

install_activate: START Sun Feb 26 05:58:41 UTC 2017
DMP package.
Netconf processes stopped
SUCCESS: install_activate /bootflash/isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin
Sun Feb 26 05:58:58 UTC 2017*Feb 26 05:58:47.655: %DMI-4-CONTROL_SOCKET_CLOSED:
SIP0: nesd: Confd control socket closed Lost connection to ConfD (45): EOF on socket to
 ConfD.
*Feb 26 05:58:47.661: %DMI-4-SUB_READ_FAIL: SIP0: vtyserverutild:
Confd subscription socket read failed Lost connection to ConfD (45):
EOF on socket to ConfD.
*Feb 26 05:58:47.667: %DMI-4-CONTROL_SOCKET_CLOSED: SIP0: syncfd:
Confd control socket closed Lost connection to ConfD (45): EOF on socket to ConfD.
*Feb 26 05:59:43.269: %DMI-5-SYNC_START: SIP0: syncfd:
External change to running configuration detected.
The running configuration will be synchronized to the NETCONF running data store.
*Feb 26 05:59:44.624: %DMI-5-SYNC_COMPLETE: SIP0: syncfd:
The running configuration has been synchronized to the NETCONF running data store.
Device#
```

次に示すのは、**show install summary** コマンドがモデル パッケージのステータスをアクティブでありコミット未完了と表示する場合の出力例です。

```
Device# show install summary

Active Packages:
bootflash:isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin
Inactive Packages:
No packages
Committed Packages:
No packages
Uncommitted Packages:
bootflash:isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin
Device#
```

次の例では、**install commit** コマンドの実行方法を示しています。

```
Device# install commit

install_commit: START Sun Feb 26 06:46:48 UTC 2017
SUCCESS: install_commit Sun Feb 26 06:46:52 UTC 2017
Device#
```

次に示すのは、**show install summary** コマンドが、更新パッケージがコミットされてリロードが繰り返されても持続することを表示する場合の出力例です。

```
Device# show install summary

Active Packages:
bootflash:isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin
Inactive Packages:
No packages
Committed Packages:
bootflash:isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin
Uncommitted Packages:
No packages
Device#
```

次の例は、更新プログラム パッケージを基本パッケージにロールバックする方法を示しています。

```
Device# install rollback to base

install_rollback: START Sun Feb 26 06:50:29 UTC 2017
7 install_rollback: Restarting impacted processes to take effect
7 install_rollback: restarting confd
*Feb 26 06:50:34.957: %DMI-4-CONTROL_SOCKET_CLOSED: SIP0: syncfd:
Confd control socket closed Lost connection to ConfD (45): EOF on socket to ConfD.
*Feb 26 06:50:34.962: %DMI-4-CONTROL_SOCKET_CLOSED: SIP0: nesd:
Confd control socket closed Lost connection to ConfD (45): EOF on socket to ConfD.
*Feb 26 06:50:34.963: %DMI-4-SUB_READ_FAIL: SIP0: vtyserverutild:
Confd subscription socket read failed Lost connection to ConfD (45):
EOF on socket to ConfD.Netconf processes stopped
7 install_rollback: DMP activate complete
SUCCESS: install_rollback Sun Feb 26 06:50:41 UTC 2017
*Feb 26 06:51:28.901: %DMI-5-SYNC_START: SIP0: syncfd:
External change to running configuration detected.
The running configuration will be synchronized to the NETCONF running data store.
*Feb 26 06:51:30.339: %DMI-5-SYNC_COMPLETE: SIP0: syncfd:
The running configuration has been synchronized to the NETCONF running data store.
Device#
```

次に、**show install package** コマンドの出力例を示します。

```
Device# show install package bootflash:
isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin

Name: isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin
Version: 16.5.1.0.199.1484082952..Everest
Platform: ISR4300
Package Type: dmp
Defect ID: CSCxxxxxxx
Package State: Added
Supersedes List: {}
Smu ID: 1
```

```
Device#
```

次の NETCONF hello メッセージの例では、新規データ モデル パッケージのバージョンを確認します。

```
Getting Capabilities: (admin @ 172.16.0.1:830)
PROTOCOL netconf
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<capabilities>
<capability>urn:ietf:params:netconf:base:1.0</capability>
<capability>urn:ietf:params:netconf:base:1.1</capability>
<capability>urn:ietf:params:netconf:capability:writable-running:1.0</capability>
<capability>urn:ietf:params:netconf:capability:xpath:1.0</capability>
<capability>urn:ietf:params:netconf:capability:validate:1.0</capability>
<capability>urn:ietf:params:netconf:capability:validate:1.1</capability>
<capability>urn:ietf:params:netconf:capability:rollback-on-error:1.0</capability>
<capability>urn:ietf:params:netconf:capability:notification:1.0</capability>
<capability>urn:ietf:params:netconf:capability:interleave:1.0</capability>
<capability>http://tail-f.com/ns/netconf/actions/1.0</capability>
<capability>http://tail-f.com/ns/netconf/extensions</capability>
<capability>urn:ietf:params:netconf:capability:with-defaults:1.0?basic-mode=
explicit&also-supported=report-all-tagged</capability>
<capability>urn:ietf:params:xml:ns:yang:ietf-netconf-with-defaults?
revision=2011-06-01&module=ietf-netconf-with-defaults</capability>
<capability>http://cisco.com/ns/yang/Cisco-IOS-XE-aaa?module=
Cisco-IOS-XE-aaa&revision=2017-02-07</capability>
<<capability>http://cisco.com/ns/yang/Cisco-IOS-XE-native?module=
Cisco-IOS-XE-native&revision=2017-01-07&features=virtual-
template,punt-num,multilink,eth-evc,esmc,efp,dot1x</capability>
Device#
```

次に、**show install log** コマンドの出力例を示します。

```
Device# show install log

[0|install_op_boot]: START Fri Feb 24 19:20:19 Universal 2017
[0|install_op_boot]: END SUCCESS Fri Feb 24 19:20:23 Universal 2017
[3|install_add]: START Sun Feb 26 05:55:31 UTC 2017
[3|install_add( FATAL)]: File path (scp) is not yet supported for this command
[4|install_add]: START Sun Feb 26 05:57:04 UTC 2017
[4|install_add]: END SUCCESS /bootflash/isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin

Sun Feb 26 05:57:22 UTC 2017
[5|install_activate]: START Sun Feb 26 05:58:41 UTC 2017
Device#
```

次の例で使用しているのは、Cisco Catalyst 3000 シリーズ スイッチのサンプル イメージです。

次の例では、モデル更新プログラム パッケージ ファイルの追加方法を示しています。

```
Device# install add file tftp://172.16.0.1//tftpboot/folder1/
cat3k_caa-universalk9.16.06.01.CSCxxxxxxx.dmp.bin

install_add: START Sat Jul 29 05:57:04 UTC 2017
Downloading file tftp://172.16.0.1//tftpboot/folder1/
cat3k_caa-universalk9.16.06.01.CSCxxxxxxx.dmp.bin
Finished downloading file tftp://172.16.0.1//tftpboot/folder1/
cat3k_caa-universalk9.16.06.01.CSCxxxxxxx.SPA.smu.bin
```

```
to bootflash:cat3k_caa-universalk9.16.06.01.CSCxxxxxxx.dmp.bin
SUCCESS: install_add /bootflash/cat3k_caa-universalk9.16.06.01.CSCxxxxxxx.dmp.bin
Sat Jul 29 05:57:22 UTC 2017
Device#
```

次に示すのは、**show install summary** コマンドが、更新パッケージがコミットされてリロードが繰り返されても持続することを表示する場合の出力例です。

```
Device# show install summary

Active Packages:
bootflash:cat3k_caa-universalk9.16.06.01.CSCxxxxxxx.dmp.bin
Inactive Packages:
No packages
Committed Packages:
bootflash:cat3k_caa-universalk9.16.06.01.CSCxxxxxxx.dmp.bin
Uncommitted Packages:
No packages
Device#
```

In-Service Model Update 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。

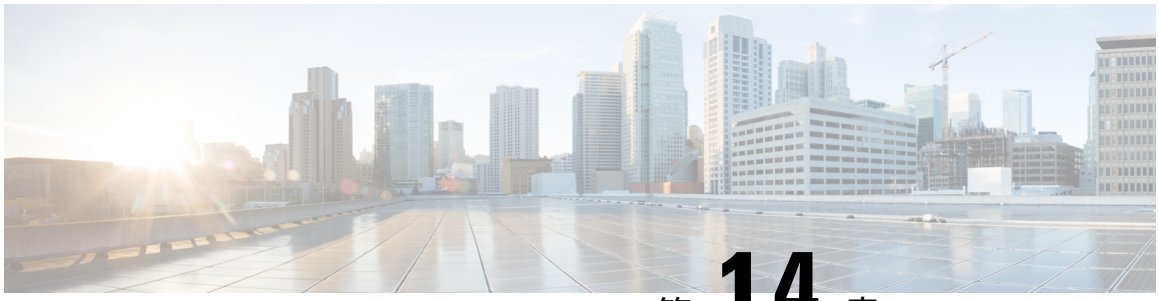
表 *31 : In-Service Model Update* の機能情報

機能名	リリース	機能情報
In-Service Model Update	Cisco IOS XE Everest 16.5.1a Cisco IOS XE Everest 16.5.1b	このモジュールでは、In-Service Model Update で YANG データ モデルを更新する方法を説明します。 Cisco IOS XE Everest 16.5.1a 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 Catalyst 9300 シリーズ スイッチ • Cisco Catalyst 9500 シリーズ スイッチ Cisco IOS XE Everest 16.5.1b 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 4000 シリーズ サービス統合型ルータ • Cisco クラウド サービス ルータ 1000v • Cisco サービス統合型仮想ルータ（ISRv） コマンド **install (Programmability)**、**show install (Programmability)** が導入または更新されました。
	Cisco IOS XE Everest 16.6.1	Cisco IOS XE Everest 16.5.1b 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 3650 シリーズ スイッチ • Cisco Catalyst 3850 シリーズ スイッチ
	Cisco IOS XE Fuji 16.7.x	Cisco IOS XE Fuji 16.7.x 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 アグリゲーション サービス ルータ

機能名	リリース	機能情報
	Cisco IOS XE Fuji 16.8.1a	Cisco IOS XE Fuji 16.8.1a では、この機能は Cisco Catalyst 9500 ハイ パフォーマンス シリーズ スイッチに実装されていました。

第 IV 部

アプリケーション ホスティング



第 14 章

アプリケーション ホスティング

ホステッドアプリケーションは Software as a Service（SaaS）ソリューションであり、コマンドを使用してリモート実行できます。アプリケーションのホスティングによって、管理者には独自のツールやユーティリティを利用するためのプラットフォームが与えられます。

このモジュールでは、アプリケーションホスティング機能とその有効化の方法について説明します。

アプリケーション ホスティングの制約事項

Catalyst 9300 シリーズ スイッチ、Catalyst 9400 シリーズ スイッチ、および Catalyst 9500 シリーズ スイッチの制約事項

- ネットワーク アドレス変換（NAT）はサポートされていません。
- アプリケーション ホスティングは、Virtual Routing and Forwarding（VRF）に対応していません。
- アプリケーションホスティングでは、専用ストレージの割り当てが必要であり、ブートフラッシュでは無効になっています。
- Cisco Catalyst 9300 シリーズ スイッチおよび Cisco Catalyst 9500 シリーズ スイッチでは、背面パネルのシスコ認定 USB フラッシュが使用可能な場合に、アプリケーションホスティングが実行されます。
- Cisco IOS XE Fuji 16.9.1 では、前面パネルの USB スティックは Cisco Catalyst 9400 シリーズスイッチではサポートされていません。アプリケーションホスティングにはハードディスク ドライブ（HDD）のみがサポートされています。

アプリケーション ホスティングに関する情報

アプリケーション ホスティングの必要性

仮想環境への移行により、再利用ができポータブルでスケーラブルなアプリケーションを開発する必要性が高まってきました。アプリケーションのホスティングによって、管理者には独自のツールやユーティリティを利用するためのプラットフォームが与えられます。ネットワークデバイスでホスティングされているアプリケーションは、さまざまな用途に利用できます。これは、既存のツールのチェーンによる自動化から、設定管理のモニタリング、統合に及びます。

Cisco のデバイスは、Linux ツール チェーンを使用して開発されたサードパーティ製の市販アプリケーションをサポートしています。ユーザは、シスコが提供するソフトウェア開発キットと相互にコンパイルされたカスタム アプリケーションを実行できます。アプリケーション ホスティングは、「Kernel Virtual Machine（KVM）」と「コンテナ」という 2 つの形態で提供されます。

IOx の概要

IOx は Cisco が開発したエンド ツー エンド アプリケーション フレームワークであり、Cisco ネットワーク プラットフォーム上のさまざまなタイプのアプリケーションに対し、アプリケーション ホスティング機能を提供します。Cisco ゲスト シェルは特殊なコンテナ展開であり、システムの開発および使用に役立つアプリケーションの 1 つです。

IOxは、構築済みアプリケーションをパッケージ化し、それらをターゲットデバイス上にホストする開発者の作業を支援する一連のサービスを提供することにより、アプリケーションのライフサイクル管理とデータ交換を容易化します。IOxのライフサイクル管理には、アプリケーションおよびデータの配布、展開、ホスティング、開始、停止（管理）、およびモニタが含まれます。IOx サービスにはアプリケーションの配布および管理ツールも含まれており、ユーザがアプリケーションを発見して IOx フレームワークに展開するのに役立ちます。

アプリケーション ホスティングは、次の機能を提供します。

- ネットワークの不均質性の遮蔽。
- デバイス上にホストされているアプリケーションのライフサイクルをリモートで管理する IOx アプリケーション プログラミング インターフェイス（API）。
- 一元的なアプリケーション ライフ サイクル管理。
- クラウド ベースの開発。

シスコ アプリケーション ホスティングの概要

シスコのアプリケーション ホスティング フレームワークは、デバイス上で実行される仮想化アプリケーションやコンテナ アプリケーションを管理する、IOx の Python プロセスです。

アプリケーション ホスティングは、次のサービスを提供します。

- コンテナ内の指定されたアプリケーションを起動する。
- 使用可能なリソース（メモリ、CPU、およびストレージ）を確認し、それらを割り当て、管理する。
- コンソール ロギングのサポートを提供する。
- REST API を介してサービスへのアクセスを提供する。
- CLI エンドポイントを提供する。
- Cisco Application Framework（CAF）と呼ばれるアプリケーション ホスティング インフラストラクチャを提供する。
- VirtualPortGroup および管理インターフェイスを介したプラットフォーム固有のネットワーキング（パケットパス）のセットアップを支援する。

コンテナは、ホスト オペレーティング システムでゲスト アプリケーションを実行するために提供される「仮想化環境」と呼ばれます。Cisco IOS-XE 仮想化サービスは、ゲスト アプリケーションを実行するための管理性とネットワーキング モデルを提供します。仮想化インフラストラクチャにより、管理者はホストとゲスト間の接続を指定する論理インターフェイスを定義できます。IOx は、論理インターフェイスをゲスト アプリケーションが使用する仮想ネットワーク インターフェイス カードにマッピングします。

コンテナに展開されるアプリケーションは、TAR ファイルとしてパッケージ化されます。これらのアプリケーションに固有の設定は、TAR ファイルの一部としてもパッケージ化されています。

デバイス上の管理インターフェイスにより、アプリケーション ホスティング ネットワークが IOS 管理インターフェイスに接続します。アプリケーションのレイヤ 3 インターフェイスは、IOS 管理インターフェイスからレイヤ 2 ブリッジ トラフィックを受信します。管理インターフェイスは、管理ブリッジを使用してコンテナ/VM インターフェイスに接続します。IP アドレスは、管理インターフェイス IP アドレスと同じサブネット上にある必要があります。

IOXMAN

IOXMAN は、シリアル デバイスをエミュレートする Libvirt を除く、ゲスト アプリケーションのロギングまたはトレース サービスを提供するトレース インフラストラクチャを確立するプロセスです。IOXMAN は、ゲスト アプリケーションのライフサイクルに基づいて、トレース サービスを有効または無効にし、ロギング データを IOS syslog に送信し、トレース データを IOx トレース ログに保存し、各ゲスト アプリケーションの IOx トレースロ グを維持します。

Catalyst 9000 シリーズ スイッチでのアプリケーション ホスティング

ここでは、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 Catalyst 9500 シリーズ ハイ パフォーマンス スイッチに固有の、アプリケーション ホスティングの特性について説明します。

これらのスイッチは、Linux コンテナでホストされるサードパーティ製アプリケーションに前面パネルからアクセスするための VirtualPortGroup インターフェイスをサポートしています。

USB 3.0 SSD は Cisco Catalyst 9300 シリーズ スイッチで有効になっています。USB 3.0 SSD は、アプリケーションをホストするための追加の 120 GB ストレージを提供します。詳細については、『インターフェイスおよびハードウェア コンフィギュレーション ガイド』の「USB 3.0 SSD の設定」の章を参照してください。

Cisco Catalyst 9400 シリーズ スイッチは、リムーバブルのスーパーバイザに装着できる M2 Serial Advanced Technology Attachment（SATA）ドライブをサポートしています。前面パネルの USB もサポートされています。ただし、Cisco Catalyst 9400 シリーズ スイッチの前面パネルの USB ではアプリケーション ホスティングはサポートされていません。Cisco IOS XE Gibraltar 16.10.1 以降のリリースでは、Cisco Catalyst 9400 シリーズ スイッチは USB 3.0 を使用したアプリケーション ホスティングをサポートしています。

Cisco Catalyst 9500 シリーズ ハイ パフォーマンス スイッチは、Cisco Catalyst 9300 シリーズ スイッチおよび Cisco Catalyst 9500 シリーズ スイッチと同様に、背面パネルの USB 3.0 をサポートしています。

VirtualPortGroup

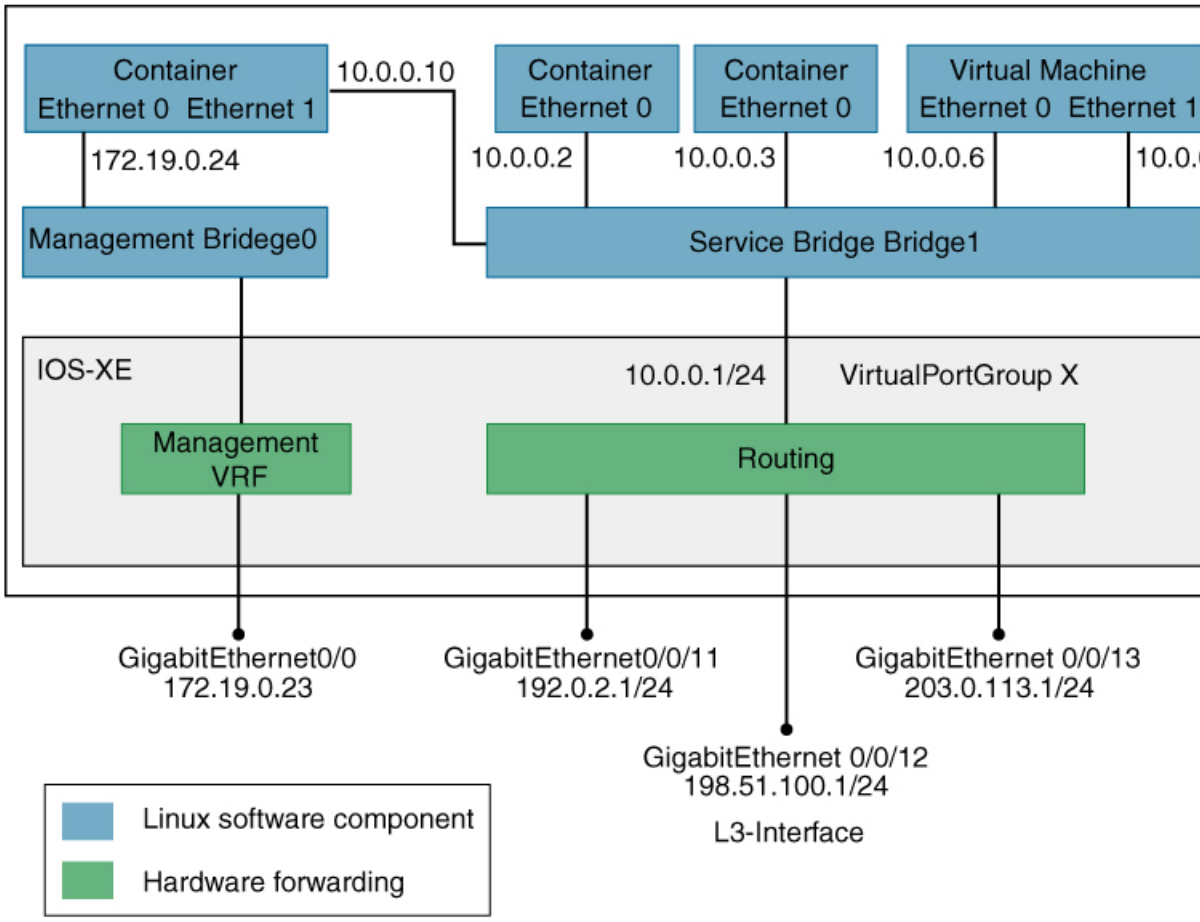
VirtualPortGroup は、Linux ブリッジ IP アドレスにマッピングする Cisco IOS 上のソフトウェア構成要素です。そのため、VirtualPortGroup は、Linux コンテナのスイッチ仮想インターフェイス（SVI）を表します。各ブリッジには、複数のインターフェイスを含めることができ、それぞれ異なるコンテナへマッピングされます。各コンテナには、複数のインターフェイスを含めることもできます。

VirtualPortGroup インターフェイスは、**interface virtualportgroup** コマンドを使用して設定します。これらのインターフェイスが作成されると、IP アドレスとその他のリソースが割り当てられます。

最大 32 個の VirtualPortGroup インターフェイスを設定できます。これらの VirtualPortGroup インターフェイスはそれぞれ 2 つの転送エントリを持ちます。

VirtualPortGroup インターフェイスは、アプリケーション ホスティング ネットワークを IOS ルーティング ドメインに接続します。アプリケーションのレイヤ 3 インターフェイスは、IOS からルーティングされたトラフィックを受信します。VirtualPortGroup インターフェイスは、SVC ブリッジを介してコンテナ/VM インターフェイスに接続します。

図 *6*: アプリケーション ホスティングのネットワーキング



vNIC

コンテナのライフサイクル管理には、内部論理インターフェイスごとに 1 つのコンテナをサポートするレイヤ 3 ルーティング モデルが使用されます。これは、各アプリケーションに対して仮想イーサネット ペアが作成されることを意味します。このペアのうち vNIC と呼ばれるインターフェイスは、アプリケーション コンテナの一部です。vpgX と呼ばれるもう 1 つのインターフェイスは、ホスト システムの一部です。

NIC は、コンテナ内の標準イーサネット インターフェイスで、プラットフォーム データプレーンに接続してパケットを送受信します。IOx は、コンテナ内の vNIC ごとに、ゲートウェイ（VirtualPortGroup インターフェイス）、IP アドレス、および一意の MAC アドレス割り当てを行います。

コンテナ/VM 内の vNIC は、標準のイーサネット インターフェイスと見なされます。

アプリケーション ホスティングの設定方法

IOx の有効化

IOx Local Manager へのアクセスを有効にするには、次の作業を実行します。Local Manager を使用することで、ホストシステム上のアプリケーションの管理、制御、モニタ、トラブルシューティング、および関連するさまざまなアクティビティを実行できます。

手順の概要

1. **enable**
2. **configure terminal**
3. **iox**
4. **ip http server**
5. **ip http secure-server**
6. **username** *name* **privilege** *level* **password** {**0** | **7** | *user-password*}*encrypted-password*
7. **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）。
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	**iox** 例： `Device(config)# iox`	IOx をイネーブルにします。
ステップ 4	**ip http server** 例： `Device(config)# ip http server`	IP または IPv6 システム上の HTTP サーバを有効化します。
ステップ 5	**ip http secure-server** 例： `Device(config)# ip http secure-server`	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
ステップ 6	**username** *name* **privilege** *level* **password** {**0** | **7** | *user-password*}*encrypted-password* 例： `Device(config)# username cisco privilege 15 password 0 ciscoI`	ユーザ名ベースの認証システムとユーザの権限レベルを確立します。 • ユーザ名の特権レベルは 15 に設定する必要があります。
ステップ 7	**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。

レイヤ 3 データ ポートへの VirtualPortGroup の設定

複数のレイヤ 3 データポートを 1 つ以上の VirtualPortGroup またはコンテナにルーティングできます。VirutalPortGroups とレイヤ 3 のデータポートは、異なるサブネット上にある必要があります。

レイヤ 3 データポートで外部ルーティングを許可するには、**ip routing** コマンドを有効にします。

手順の概要

1. **enable**
2. **configure terminal**
3. **ip routing**
4. **interface** *type number*
5. **no switchport**
6. **ip address** *ip-address mask*
7. **exit**
8. **interface** *type number*
9. **ip address** *ip-address mask*
10. **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）。
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。

	コマンドまたはアクション	目的
ステップ 3	**ip routing** 例： `Device(config)# ip routing`	IP ルーティングをイネーブルにします。 • レイヤ3データポートで外部ルーティングを許可するには、**ip routing** コマンドを有効にする必要があります。
ステップ 4	**interface** *type number* 例： `Device(config)# interface gigabitethernet 1/0/1`	インターフェイスを設定し、インターフェイス コンフィギュレーション モードを開始します。
ステップ 5	**no switchport** 例： `Device(config-if)# no switchport`	インターフェイスをレイヤ3モードにし、スイッチポートではなくルータ インターフェイスのように動作させます。
ステップ 6	**ip address** *ip-address mask* 例： `Device(config-if)# ip address 10.1.1.1 255.255.255.254`	インターフェイスに IP アドレスを設定します。
ステップ 7	**exit** 例： `Device(config-if)# exit`	インターフェイス 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	**interface** *type number* 例： `Device(config)# interface virtualportgroup 0`	インターフェイスを設定し、インターフェイス コンフィギュレーション モードを開始します。
ステップ 9	**ip address** *ip-address mask* 例： `Device(config-if)# ip address 192.0.2.1 255.255.255.1`	インターフェイスに IP アドレスを設定します。
ステップ 10	**end** 例： `Device(config-if)# end`	インターフェイス 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。

アプリケーションのインストールとアンインストール

手順の概要

1. **enable**
2. **app-hosting install appid** *application-name* **package** *package-path*
3. **app-hosting activate appid** *application-name*

4. **app-hosting start appid** *application-name*
5. **app-hosting stop appid** *application-name*
6. **app-hosting deactivate appid** *application-name*
7. **app-hosting uninstall appid** *application-name*

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）。
ステップ 2	**app-hosting install appid** *application-name* **package** *package-path* 例： `Device# app-hosting install appid lxc_app package flash:my_iox_app.tar`	指定された場所からアプリケーションをインストールします。 • アプリケーションは、flash、bootflash、usbflash0、usbflash1、harddisk などの任意のローカル ストレージの場所からインストールできます。
ステップ 3	**app-hosting activate appid** *application-name* 例： `Device# app-hosting activate appid lxc_app`	アプリケーションをアクティブ化します。 • このコマンドは、すべてのアプリケーション リソース要求を検証し、すべてのリソースが使用可能な場合はアプリケーションがアクティブになります。それ以外の場合は、アクティベーションが失敗します。
ステップ 4	**app-hosting start appid** *application-name* 例： `Device# app-hosting start appid lxc_app`	アプリケーションを起動します。 • アプリケーションの起動スクリプトがアクティブ化されます。
ステップ 5	**app-hosting stop appid** *application-name* 例： `Device# app-hosting stop appid lxc_app`	アプリケーションを停止します。
ステップ 6	**app-hosting deactivate appid** *application-name* 例： `Device# app-hosting deactivate appid lxc_app`	アプリケーションに割り当てられているすべてのリソースを無効にします。
ステップ 7	**app-hosting uninstall appid** *application-name* 例： `Device# app-hosting uninstall appid lxc_app`	アプリケーションをアンインストールします。 • 保存されているすべてのパッケージとイメージをアンインストールします。アプリケーションに対するすべての変更と更新も削除されます。

アプリケーションの IP アドレスの手動設定

次の方法で、Kernel-based Virtual Machine（KVM）または Linux コンテナ（LXC）の IP アドレスを設定できます。

- KVM または LXC に直接ログインし、 **ifconfig** Linux コマンドを設定します。
- KVM または LXC で Dynamic Host Configuration Protocol（DHCP）を有効にし、IOS 設定で DHCP サーバ/リ レーを設定します。

手順の概要

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **vrf forwarding** *vrf-name*
5. **ip address** *ip-address mask*
6. **exit**
7. **interface** *type number*
8. **ip address** *ip-address mask*
9. **exit**
10. **app-hosting appid** *name*
11. **app-vnic gateway virtualportgroup** *number* **guest-interface** *network-interface*
12. **exit**
13. **app-vnic management guest-interface** *network-interface*
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）。
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	**interface** *type number* 例： `Device(config)# interface gigabitethernet 0/0`	インターフェイスを設定し、インターフェイス コンフィギュレーション モードを開始します。
ステップ 4	**vrf forwarding** *vrf-name* 例： `Device(config-if)# vrf forwarding Mgmt-vrf`	インターフェイスまたはサブインターフェイスに Virtual Routing and Forwarding（VRF）インスタンスまたは仮想ネットワークを関連付けます。

	コマンドまたはアクション	目的
ステップ 5	**ip address** *ip-address mask* 例： `Device(config-if)# ip address 198.51.100.1 255.255.255.254`	インターフェイスに IP アドレスを設定します。
ステップ 6	**exit** 例： `Device(config-if)# exit`	インターフェイス 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 7	**interface** *type number* 例： `Device(config)# interface VirtualPortGroup 0`	インターフェイスを設定し、インターフェイス コンフィギュレーション モードを開始します。
ステップ 8	**ip address** *ip-address mask* 例： `Device(config-if)# ip address 192.0.2.1 255.255.255.1`	インターフェイスに IP アドレスを設定します。
ステップ 9	**exit** 例： `Device(config-if)# exit`	インターフェイス 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 10	**app-hosting appid** *name* 例： `Device(config)# app-hosting appid lxc_app`	アプリケーションを設定し、アプリケーション ホスティング コンフィギュレーション モードを開始します。
ステップ 11	**app-vnic gateway virtualportgroup** *number* **guest-interface** *network-interface* 例： `Device(config-app-hosting)# app-vnic gateway1 virtualportgroup 0 guest-interface 1`	ゲストインターフェイスを VirtualPortGroup に接続し、アプリケーションホスティング ゲートウェイ コンフィギュレーション モードを開始します。 • **gateway** キーワードは、定義されたポートマッピングを表す一意の識別子です。任意のゲートウェイ番号を指定できます。この例では gateway1 を使用しています。 • number**virtualportgroup** のキーワード引数ペアは、コンテナへの接続に使用される IOS VirtualPortGroup インターフェイス番号を指定します。この例では virtualportgroup 0 を使用しています。 • **guest-interface** *network-interface* のキーワード引数ペアは、指定された VirtualPortGroup に接続されているコンテナの内部イーサネット インターフェイス番号を指定します。この例では、

	コマンドまたはアクション	目的
		コンテナのイーサネット1インターフェイスに対して guest-interface 1 を使用しています。
ステップ 12	**exit** 例： `Device(config-app-hosting-gateway0)# exit`	アプリケーションホスティング ゲートウェイ コンフィギュレーション モードを終了し、アプリケーション ホスティング コンフィギュレーション モードに戻ります。
ステップ 13	**app-vnic management guest-interface** *network-interface* 例： `Device(config-app-hosting)# vnic management guest-interface 1`	ゲスト インターフェイスを管理ポートに接続し、アプリケーションホスティング管理ゲートウェイ コンフィギュレーション モードを開始します。 • **management** キーワードは、コンテナに接続されている IOS 管理 GigabitEthernet0/0 インターフェイスを指定します。 • network-interface**guest-interface** のキーワード引数ペアは、IOS管理インターフェイスに接続されているコンテナの内部イーサネットインターフェイス番号を指定します。この例では、コンテナのイーサネット1インターフェイスに対して guest-interface 1 を使用しています。
ステップ 14	**end** 例： `Device(config-app-hosting-mgmt-gateway)# end`	アプリケーションホスティング管理ゲートウェイ 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。

LXC での静的 IP アドレスの設定

- Kernel-based Virtual Machine（KVM）は、アプリケーション ホスティングの静的 IP 設定をサポートしていません。
- 最後に設定されたデフォルト ゲートウェイ設定のみが使用されます。
- 最後に設定されたネーム サーバ設定のみが使用されます。

LXC の IP アドレスは、IOS コマンドを使用して設定できます。

手順の概要

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **vrf forwarding** *vrf-name*
5. **ip address** *ip-address mask*

6. **exit**
7. **interface** *type number*
8. **ip address** *ip-address mask*
9. **exit**
10. **app-hosting appid** *name*
11. **app-vnic gateway virtualportgroup** *number* **guest-interface** *network-interface*
12. **guest-ipaddress** *ip-address* **netmask** *netmask*
13. **exit**
14. **name-server** *ip-address*
15. **app-vnic management guest-interface** *interface-number*
16. **guest-ipaddress** *ip-address* **netmask** *netmask*
17. **exit**
18. **app-default-gateway** *ip-address* **guest-interface** *network-interface*
19. **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）。
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	**interface** *type number* 例： `Device(config)# interface gigabitethernet 0/0`	インターフェイスを設定し、インターフェイス コンフィギュレーション モードを開始します。
ステップ 4	**vrf forwarding** *vrf-name* 例： `Device(config-if)# vrf forwarding Mgmt-vrf`	インターフェイスまたはサブインターフェイスに Virtual Routing and Forwarding（VRF）インスタンスまたは仮想ネットワークを関連付けます。
ステップ 5	**ip address** *ip-address mask* 例： `Device(config-if)# ip address 198.51.100.1 255.255.255.254`	インターフェイスに IP アドレスを設定します。
ステップ 6	**exit** 例： `Device(config-if)# exit`	インターフェイス 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 7	**interface** *type number* 例：	インターフェイスを設定し、インターフェイス コンフィギュレーション モードを開始します。

	コマンドまたはアクション	目的
	`Device(config)# interface VirtualPortGroup 0`	
ステップ 8	**ip address** *ip-address mask* 例： `Device(config-if)# ip address 192.0.2.1 255.255.255.1`	インターフェイスに IP アドレスを設定します。
ステップ 9	**exit** 例： `Device(config-if)# exit`	インターフェイス 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 10	**app-hosting appid** *name* 例： `Device(config)# app-hosting appid lxc_app`	アプリケーションを設定し、アプリケーション ホスティング コンフィギュレーション モードを開始します。
ステップ 11	**app-vnic gateway virtualportgroup** *number* **guest-interface** *network-interface* 例： `Device(config-app-hosting)# app-vnic gateway1 virtualportgroup 0 guest-interface 1`	仮想ネットワーク インターフェイス ゲートウェイおよびゲスト インターフェイスの詳細を設定し、アプリケーションホスティング ゲートウェイ コンフィギュレーション モードを開始します。
ステップ 12	**guest-ipaddress** *ip-address* **netmask** *netmask* 例： `Device(config-app-hosting-gateway1)# guest-ipaddress 10.0.0.3 netmask 255.255.255.0`	ゲストの IP アドレスとマスクを設定します。
ステップ 13	**exit** 例： `Device(config-app-hosting-gateway1)# exit`	アプリケーションホスティング ゲートウェイ コンフィギュレーション モードを終了し、アプリケーション ホスティング コンフィギュレーション モードに戻ります。
ステップ 14	**name-server** *ip-address* 例： `Device(config-app-hosting)# name-server0 10.2.2.2`	DNS サーバを設定します。
ステップ 15	**app-vnic management guest-interface** *interface-number* 例： `Device(config-app-hosting)# app-vnic management guest-interface 0`	仮想ネットワーク インターフェイスおよびゲスト インターフェイスの管理ゲートウェイを設定し、アプリケーションホスティング ゲートウェイ コンフィギュレーション モードを開始します。
ステップ 16	**guest-ipaddress** *ip-address* **netmask** *netmask* 例： `Device(config-app-hosting-mgmt-gateway)# guest-ipaddress 172.19.0.24 netmask 255.255.255.0`	管理ゲストインターフェイスの詳細を設定します。

	コマンドまたはアクション	目的
ステップ 17	**exit** 例： `Device(config-app-hosting-mgmt-gateway)# exit`	アプリケーションホスティング管理ゲートウェイ コンフィギュレーション モードを終了し、アプリケーション ホスティング コンフィギュレーション モードに戻ります。
ステップ 18	**app-default-gateway** *ip-address* **guest-interface** *network-interface* 例： `Device(config-app-hosting-mgmt-gateway)# app-default-gateway 172.19.0.23 guest-interface 0`	デフォルトの管理ゲートウェイを設定します。
ステップ 19	**end** 例： `Device(config-app-hosting)# end`	アプリケーション ホスティング 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。

アプリケーションのリソース設定の上書き

リソースの変更は、**app-hosting activate** コマンドが設定された後にのみ有効になります。

手順の概要

1. **enable**
2. **configure terminal**
3. **app-hosting appid** *name*
4. **app-resource profile** *name*
5. **cpu** *unit*
6. **memory** *memory*
7. **vcpu** *number*
8. **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）。
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。

	コマンドまたはアクション	目的
ステップ 3	**app-hosting appid** *name* 例： `Device(config)# app-hosting appid lxc_app`	アプリケーション ホスティングをイネーブルにし、アプリケーション ホスティング コンフィギュレーション モードを開始します。
ステップ 4	**app-resource profile** *name* 例： `Device(config-app-hosting)# app-resource profile custom`	カスタム アプリケーション リソース プロファイルを設定し、カスタム アプリケーション リソース プロファイル コンフィギュレーション モードを開始します。 • カスタムプロファイル名のみがサポートされています。
ステップ 5	**cpu** *unit* 例： `Device(config-app-resource-profile-custom)# cpu 7400`	アプリケーションのデフォルトのCPU割り当てを変更します。 • リソース値はアプリケーション固有のため、これらの値を変更した場合、アプリケーションが変更後も確実に稼働できることを確認する必要があります。
ステップ 6	**memory** *memory* 例： `Device(config-app-resource-profile-custom)# memory 2048`	デフォルトのメモリ割り当てを変更します。
ステップ 7	**vcpu** *number* 例： `Device(config-app-resource-profile-custom)# vcpu 2`	アプリケーションの仮想 CPU（vCPU）割り当てを変更します。
ステップ 8	**end** 例： `Device(config-app-resource-profile-custom)# end`	カスタム アプリケーション リソース プロファイル 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。

アプリケーション ホスティングを有効にするための LVM の作成

（注） このタスクは Cisco Catalyst 9300 シリーズスイッチにのみ適用されます。

ドライブでアプリケーションホスティングを有効にするには、ローカルボリュームマネージャ（LVM）を作成する必要があります。

手順の概要

1. **enable**
2. 次のいずれかのコマンドを使用します。
   - **request platform hardware filesystem ssdflash-***slot*: **initialize**
   - **format ssdflash:LVM**
3. **request platform hardware filesystem ssdflash-** *slot*: **sanitize**
4. **show** *usbflash1:* **filesys**
5. **show file systems**

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）。
ステップ 2	次のいずれかのコマンドを使用します。 • **request platform hardware filesystem ssdflash-***slot*: **initialize** • **format ssdflash:LVM**	ドライブが LVM モードに入り、フォーマットされます。
ステップ 3	**request platform hardware filesystem ssdflash-** *slot*: **sanitize**	ドライブが raw デバイスの状態に戻ります。すべてのデータがドライブ上で消去されます。
ステップ 4	**show** *usbflash1:* **filesys**	LVM ファイル システムの情報を表示します。
ステップ 5	**show file systems**	ファイル システムの全般情報を表示します。

アプリケーション ホスティング コンフィギュレーションの確認

手順の概要

1. **enable**
2. **show iox-service**
3. **show app-hosting detail**
4. **show app-hosting list**

手順の詳細

ステップ 1 **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）。

例：

```
Device> enable
```

ステップ 2　show iox-service

すべての IOx サービスのステータスを表示します。

例：

```
Device# show iox-service

IOx Infrastructure   Summary:
-------------------------------------
IOx Service (CAF)    : Running
IOx Service (HA)     : Running
IOx Service (IOxman) : Running
Libvirtd             : Running
```

ステップ 3　show app-hosting detail

アプリケーションに関する詳細情報を表示します。

例：

```
Device# show app-hosting detail

State                     : Running
Author                    : Cisco Systems, Inc
Application
 Type                     : vm
 App id                   : Wireshark
 Name                     : Wireshark
 Version                  : 3.4
 Activated Profile Name   : custom
 Description              : Lubuntu based Wireshark
Resource Reservation
 Memory                   : 1900 MB
 Disk                     : 10 MB
 CPU                      : 4000 units
 VCPU                     : 2
Attached devices
Type          Name         Alias
---------------------------------------------------------------------
Serial/shell
Serial/aux
Serial/Syslog              serial2
Serial/Trace               serial3
Network Interfaces
---------------------------------------------------------------
eth0:
 MAC address              : 52:54:dd:80:bd:59
 IPv4 address
eth1:
 MAC address              : 52:54:dd:c7:7c:aa
 IPv4 address
```

ステップ 4　show app-hosting list

アプリケーションとそれらのステータスの一覧を表示します。

例：

```
Device# show app-hosting list

App id              State
------------------------------------------
Wireshark           Running
```

アプリケーション ホスティングの設定例

例：IOx の有効化

```
Device> enable
Device# configure terminal
Device(config)# iox
Device(config)# ip http server
Device(config)# ip http secure-server
Device(config)# username cisco privilege 15 password 0 ciscoI
Device(config)# end
```

例：レイヤ 3 データ ポートへの VirtualPortGroup の設定

```
Device> enable
Device# configure terminal
Device(config)# ip routing
Device(config)# interface gigabitethernet 1/0/1
Device(config-if)# no switchport
Device(config-if)# ip address 10.1.1.1 255.255.255.254
Device(config-if)# exit
Device(config)# interface virtualportgroup 0
Device(config-if)# ip address 192.0.2.1 255.255.255.1
Device(config-if)# end
```

例: アプリケーションのインストールとアンインストール

```
Device> enable
Device# app-hosting install appid lxc_app package flash:my_iox_app.tar.tar
Device# app-hosting activate appid lxc_app
Device# app-hosting start appid lxc_app
Device# app-hosting stop appid lxc_app
Device# app-hosting deactivate appid lxc_app
Device# app-hosting uninstall appid lxc_app
```

例：アプリケーションの IP アドレスの手動設定

```
Device# configure terminal
Device(config)# interface gigabitethernet 0/0
Device(config-if)# vrf forwarding Mgmt-vrf
Device(config-if)# ip address 198.51.100.1 255.255.255.254
Device(config-if)# exit
Device(config)# interface virtualportgroup 0
Device(config-if)# ip address 192.0.2.1 255.255.255.1
Device(config-if)# exit
Device(config)# app-hosting appid lxc_app
Device(config-app-hosting)# app-vnic gateway1 virtualportgroup 0 guest-interface 1
Device(config-app-hosting-gateway0)# exit
Device(config-app-hosting)# vnic management guest-interface 1
Device(config-app-hosting-mgmt-gateway)# end
```

例：LXC での静的 IP アドレスの設定

```
Device# configure terminal
Device(config)# interface gigabitethernet 0/0
Device(config-if)# vrf forwarding Mgmt-vrf
Device(config-if)# ip address 198.51.100.1 255.255.255.254
Device(config-if)# exit
Device(config)# interface virtualportgroup 0
Device(config-if)# ip address 192.0.2.1 255.255.255.1
Device(config-if)# exit
Device(config)# app-hosting appid lxc_app
Device(config-app-hosting)# app-vnic gateway1 virtualportgroup 0 guest-interface 1
Device(config-app-hosting-gateway1)# guest-ipaddress 10.0.0.3 netmask 255.255.255.0
Device(config-app-hosting-gateway1)# exit
Device(config-app-hosting)# name-server0 10.2.2.2
Device(config-app-hosting)# app-vnic management guest-interface 0
Device(config-app-hosting-mgmt-gateway)# guest-ipaddress 172.19.0.24 netmask 255.255.255.0
Device(config-app-hosting-mgmt-gateway)# exit
Device(config-app-hosting-mgmt-gateway)# app-default-gateway 172.19.0.23 guest-interface
 0
Device(config-app-hosting)# end
```

例：アプリケーションのリソース設定の上書き

```
Device# configure terminal
Device(config)# app-hosting appid lxc_app
Device(config-app-hosting)# app-resource profile custom
Device(config-app-resource-profile-custom)# cpu 7400
Device(config-app-resource-profile-custom)# memory 2048
Device(config-app-resource-profile-custom)# vcpu 2
Device(config-app-resource-profile-custom)# end
```

アプリケーション ホスティングに関する機能情報

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。

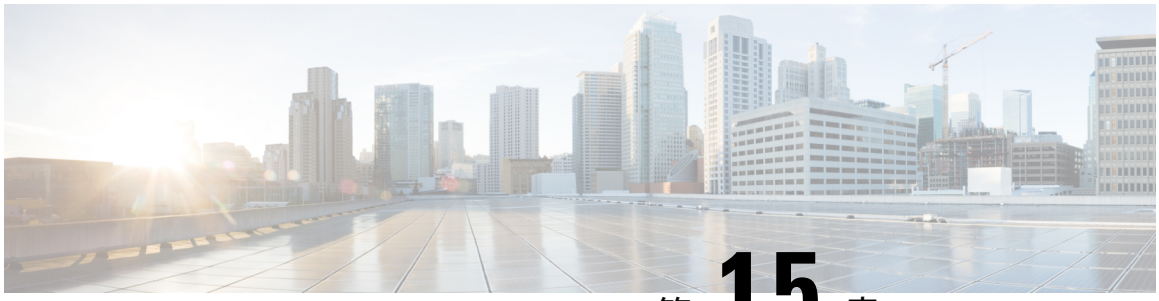
プラットフォームのサポートおよびシスコ ソフトウェア イメージのサポートに関する情報を検索するには、Cisco Feature Navigator を使用します。Cisco Feature Navigator にアクセスするには、www.cisco.com/go/cfn に移動します。Cisco.com のアカウントは必要ありません。

表 *32 :* アプリケーション ホスティングに関する機能情報

機能名	リリース	機能情報
アプリケーション ホスティング	Cisco IOS XE Fuji 16.9.1	ホステッドアプリケーションはSoftware as a Service（SaaS）ソリューションであり、ユーザはこのソリューションの実行と運用を完全にクラウドから行うことができます。このモジュールでは、アプリケーション ホスティング機能とその有効化の方法について説明します。 Cisco IOS XE Fuji 16.9.1 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 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

第 V 部

OpenFlow



第 15 章

OpenFlow

このモジュールでは、デバイスで OpenFlow を有効化して設定する方法について説明します。

OpenFlow の前提条件

デバイスを OpenFlow モードで起動する必要があります。

OpenFlow モードは、スイッチで **boot mode openflow** コマンドを設定すると有効になります。すべてのポートがこのモードになり、スイッチは通常の Cisco IOS XE 機能をサポートしなくなります。

OpenFlow について

OpenFlow の概要

OpenFlow は Open Networking Foundation（ONF）の仕様で、フローベースの転送インフラストラクチャと、標準化されたアプリケーション プログラム インターフェイスを定義します。OpenFlow では、セキュアなチャネルを介して、デバイスのフォワーディング機能を方向付けすることができます。

OpenFlow は、コントローラ（コントロール プレーン）とイーサネット スイッチ（データ）プレーン）の間のプロトコルです。スイッチ/デバイスには、パイプラインに配置されたフローテーブルがあります。フローとは、これらのテーブルに到達するパケットを調べるためのルールです。

スイッチ上の OpenFlow エージェントは、OpenFlow プロトコルを使用してコントローラと通信します。エージェントは、OpenFlow 1.0（有線プロトコル 0x1）と OpenFlow 1.3（有線プロトコル 0x4）の両方をサポートしています。エージェントは最大 8 つのコントローラ接続を持つことができます。これらの接続はスイッチオーバー後は維持されず、コントローラはエージェントに再接続する必要があります。

Cisco Catalyst 9400 シリーズ スイッチでの OpenFlow の実装はステートレスです。ノンストップ フォワーディング（NSF）はサポートされていません。スタンバイのスーパーバイザは、フロー データベースと同期しません。

Openflow コントローラ

Openflow コントローラは、Openflow プロトコルを使用して Openflow スイッチとやり取りするエンティティです。ほとんどの場合、コントローラは多数の Openflow 論理スイッチを管理するソフトウェアです。コントローラではネットワークを一元的に表示でき、管理者はこれを使用して、ネットワーク トラフィックの処理方法について基盤となるシステム（スイッチおよびルータ）に指示を出すことができます。通常、コントローラは Linux サーバで実行され、OpenFlow 対応スイッチに IP 接続できる必要があります。

コントローラはスイッチを管理し、スイッチ上でフローを挿入および削除します。これらのフローは、OpenFlow 1.3 および 1.0 の「照合」と「アクション」の基準のサブセットをサポートしています。

スイッチは、管理ポートを使用してコントローラに接続します。管理ポートは管理用の Virtual Routing and Forwarding（VRF）インスタンスの中にあり、そのためコントローラへのセキュアな接続を提供します。コントローラをスイッチに接続するには、コントローラへの到達が可能な IP アドレスとポート番号を設定します。

OpenFlow テーブル パイプライン

OpenFlow テーブル機能要求メッセージを使用すると、OpenFlow コントローラから、OpenFlow が管理するデバイスについて既存のフローテーブルの機能を照会したり、指定した設定と一致するようにこれらのテーブルを設定したりできます。

テーブルはすべて、照合およびアクション機能のサブセットを使用して設定できます。テーブルのサイズを実行時に変更することもできます。新しいフローテーブル設定が正常に適用されると、古いフロー テーブルのフロー エントリが通知なく削除されます。動的に設定されたフローテーブルは、再起動後は維持されません。デバイスが起動するとデフォルトのパイプラインが起動します。

OpenFlow コントローラからの要求に基づいて新しいフロー テーブルを設定している間、既存のフローを流れる進行中のトラフィックはすべてドロップされます。

フローの管理

フロー エントリは、パケットの照合と処理に使用されるフロー テーブル内の要素です。これには、照合設定の優先順位、パケットを照合するための一連の照合フィールド、適用する一連

の命令、パケット カウンタ、およびバイト カウンタが含まれています。また、フローごとにタイムアウト（ハード タイムアウトまたは非アクティブ タイムアウト）も関連付けられており、フローの自動削除に使用されます。

Cisco Catalyst 9000 シリーズ スイッチは最大 9 個のフロー テーブルをサポートしています。

各フローは次の情報を提供します。

- 優先順位：優先順位の高いフローが先に照合されます。フローの更新では、設定された優先順位に基づいて、すべてのフローに優先順位を付ける必要があります。
- 照合フィールド：パケットを照合する際のフローエントリの一部。照合フィールドは、さまざまなパケット ヘッダー フィールドと照合できます。フィールドに照合情報が指定されていない場合は、ワイルドカードが使用されます。
- アクション：パケットに対して作用する操作。

フローの操作

ここでは、フローが OpenFlow デバイスでプログラムされるようにコントローラから送信されるときに実行される操作について説明します。

デバイスには、パイプラインに配置されたフローテーブルがあります。パイプライン機能情報は、パイプラインの構造を指定します。たとえば、テーブル/ステージの数、各ステージが実行できる機能（照合/アクション）、各テーブルのサイズなどがあります。

コントローラがフロー要求を送信すると、OpenFlow エージェントは、ハードウェアがフローを処理できるかどうかを確認します。エージェントは、スイッチの起動時に定義されるハードウェアの機能とフローとを比較します。フローが有効であれば、該当するフローテーブルにプログラムされます。

新しいパイプラインが検証された場合（ハードウェアがパイプラインをサポートできるかどうか）、そのパイプラインは、フローをインストールできるかどうかのチェックに使用される新しい機能セットになります。

パイプラインがインスタンス化され、フローがインストールされると、パケットがスイッチから転送されます。優先順位の最も高い、一致するフローエントリが見つかるまで、入力パケットが各フローテーブル内のフローと照合されます。パケットの照合は、完全一致の場合もあれば（テーブルのすべてのフィールドが正確に一致する）、部分一致の場合もあります（一部または全部のフィールドに一致し、ビットマスクを持つフィールドが部分的に一致する場合がある）。設定されたアクションに基づいて、パケットが変更されるか転送される場合があります。アクションは、パイプライン内でいつでも適用できます。アクションによって、次の照合対象のフロー テーブル、パケットの出力ポートのセット、およびパケットをコントローラにルーティングするかどうかが決まる場合があります。

サポートされている OpenFlow 操作

OpenFlow 1.3 の次のテーブル、照合、および操作がサポートされています。

表 *33 :* サポートされている *OpenFlow* 操作

テーブル	テーブル名	照合フィールド	オペレーション
0	PORT_ACL	該当なし	ユーザが指定したアクセスコントロールリスト（ACL）をポートに適用し、次のテーブルに送信します。
1	VLAN	in_port vlan_vid eth_src eth_dst eth_type	• スパニング ツリー プロトコル（STP）のブリッジ プロトコル データ ユニット（BPDU）をドロップします。 • Link Layer Discovery Protocol（LLDP）パケットをドロップします。 • ブロードキャスト ソース トラフィックをドロップします。 • コントローラのMACアドレスをスプーフィングしている送信元からのトラフィックをドロップします。 • タグ付きポートの場合、VLAN_VID と照合し、次のテーブルに送信します。 • タグなしポートの場合、は、ポートのネイティブ VLAN を表す VLAN_VID を持つパケットに VLAN フレームをプッシュし、次のテーブルに送信します。 • 不明なトラフィックをドロップします。
2	ACL	該当なし	ユーザが指定した ACL をパケットに適用し、次のテーブルに送信します。

テーブル	テーブル名	照合フィールド	オペレーション

3	ETH_SRC	in_port
		vlan_vid
		eth_src
		eth_dst
		eth_type
		ip_proto
		icmpv6_type
		ipv6_nd_target
		arp_tpa
		ipv4_src
			• レイヤ 3 トラフィックを IPv4 またはIPv6転送情報ベース（FIB）テーブルに送信して処理します。
			• メッセージ内のパケットを介して、コントローラ宛てのトラフィックを送信します。
			• 学習した送信元MACアドレスを ETH_DST に送信します。
			  • パケットを介して、不明なトラフィックをコントローラに送信します（学習のため）。
			  • ETH_DSTテーブルに送信します。

4	IPv4_FIB	vlan_vid eth_type ip_proto ipv4_src ipv4_dst	• 各ルートのネクストホップに IP トラフィックをルーティングします。 • コントローラのMACアドレスに ETH_SRC を設定します。 • ETH_DSTをネクストホップの解決済みMACアドレスに設定します。 • 存続可能時間（TTL）を減らします。 • ETH_DST テーブルに送信します。 • 不明なトラフィックをドロップします。

テーブル	テーブル名	照合フィールド	オペレーション
5	IPv6_FIB	vlan_vid eth_type ip_proto icmpv6_type ipv6_dst	• 学習した各ルートのネクストホップに IP トラフィックをルーティングします。 • コントローラの MAC アドレスに ETH_SRC を設定します。 • ETH_DST をネクストホップの解決済み MAC アドレスに設定します。 • TTL を減らします。 • ETH_DST テーブルに送信します。 • 不明なトラフィックをドロップします。
6	ETH_DST	該当なし	コントローラにトラフィックを送信します。
7	ETH_DST	vlan_vid eth_dst	• 学習した宛先 MAC アドレスのホストに出力パケットを送信します。 • 不明なトラフィックを FLOOD テーブルに送信します。
8	FLOOD	in_port vlan_vid eth_dst	• VLAN 内のブロードキャストをフラッディングします。 • VLAN 内のマルチキャストをフラッディングします。 • VLAN 内の不明なトラフィックをフラッディングします。

OpenFlow の設定方法

デバイスでの OpenFlow モードの有効化

スイッチが通常モードで動作している場合は、以前の設定を削除するように **write erase** コマンドを設定することをお勧めします。

手順の概要

1. **enable**
2. **configure terminal**
3. **boot mode openflow**
4. **exit**
5. **write erase**
6. **reload**
7. **enable**
8. **show boot mode**

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）。
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	**boot mode openflow** 例： `Device(config)# boot mode openflow`	OpenFlow 転送モードをイネーブルにします。
ステップ 4	**exit** 例： `Device(config)# exit`	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。
ステップ 5	**write erase** 例： `Device# write erase`	NVRAM 内のすべてのファイルを消去します。 • この操作は、デバイスが以前に通常モードで動作していた場合にお勧めします。
ステップ 6	**reload** 例： `Device# reload`	スイッチをリロードし、スイッチの OpenFlow フォワーディング モードを有効にします。
ステップ 7	**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）。
ステップ 8	**show boot mode** 例： `Device# show boot mode`	デバイスのフォワーディング モードに関する情報を表示します。

例

次の **show boot mode** コマンドの出力例は、デバイスが OpenFlow モードであることを示しています。

```
Device# show boot mode

System initialized in openflow forwarding mode
System configured to boot in openflow forwarding mode
```

次のタスク

通常モードに戻るには、**no boot mode openflow**コマンドを設定して、デバイスをリロードします。

OpenFlow の設定

手順の概要

1. **enable**
2. **configure terminal**
3. **feature openflow**
4. **openflow**
5. **switch 1 pipeline 1**
6. **controller ipv4** *ip-address* **port** *port-number* **vrf** *vrf-name* **security** {**none** | **tls**}
7. **datapath-id** *ID*
8. **tls trustpoint local** *name* **remote** *name*
9. **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。 • プロンプトが表示されたら、パスワードを入力します。
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	**feature openflow** 例： `Device(config)# feature openflow`	OpenFlow 機能をイネーブルにします。

	コマンドまたはアクション	目的
ステップ 4	**openflow** 例： `Device(config)# openflow`	OpenFlow設定をイネーブルにし、OpenFlow コンフィギュレーション モードを開始します。
ステップ 5	**switch 1 pipeline 1** 例： `Device(config-openflow)# switch 1 pipeline 1`	論理スイッチとパイプラインを設定し、OpenFlow のスイッチ コンフィギュレーション モードを開始します。
ステップ 6	**controller ipv4** *ip-address* **port** *port-number* **vrf** *vrf-name* **security** {**none** | **tls**} 例： `Device(config-openflow-switch)# controller ipv4 10.2.2.2 port 6633 vrf Mgmt-vrf security tls`	コントローラに接続します。 • OpenFlow コントローラの接続セキュリティ オプションとしてTLSを設定している場合は、**tls trustpoint** コマンドを設定する必要があります。 • OpenFlowコントローラのセキュリティ オプションを設定していない場合は、**tls trustpoint** コマンドを設定する必要はありません。
ステップ 7	**datapath-id** *ID* 例： `Device(config-openflow-switch)# datapath-id 0x12345678`	（任意）OpenFlow の論理スイッチ ID を設定します。 • ID 引数にはスイッチ ID を指定します。これは 16 進値です。
ステップ 8	**tls trustpoint local** *name* **remote** *name* 例： `Device(config-openflow-switch)# tls trustpoint local trustpoint1 remote trustpoint1`	（任意）OpenFlow スイッチの Transport Layer Security（TLS）トラストポイントを設定します。
ステップ 9	**end** 例： `Device(config-openflow-switch)# end`	OpenFlow スイッチの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。

OpenFlow の確認

手順の概要

1. **enable**
2. **show openflow hardware capabilities**
3. **show openflow switch 1 controller**
4. **show openflow switch 1 ports**
5. **show openflow switch 1 flows list**

手順の詳細

ステップ 1 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）。

例：

```
Device> enable
```

ステップ 2 show openflow hardware capabilities

OpenFlow デバイスのハードウェア機能を表示します。

例：

```
Device# show openflow hardware capabilities

  Max Interfaces: 1000
  Aggregated Statistics: YES


  Pipeline ID: 1
    Pipeline Max Flows: 2322
    Max Flow Batch Size: 100
    Statistics Max Polling Rate (flows/sec): 10000
    Pipeline Default Statistics Collect Interval: 5

Flow table ID: 0

    Max Flow Batch Size: 100
    Max Flows: 1022
    Bind Subintfs: FALSE
    Primary Table: TRUE
    Table Programmable: TRUE
    Miss Programmable: TRUE
    Number of goto tables: 1
    Goto table id: 1
    Number of miss goto tables: 1
    Miss Goto table id: 1
    Stats collection time for full table (sec): 1

!
!
!
```

ステップ 3 show openflow switch 1 controller

スイッチに接続されているコントローラに関する情報を表示します。

例：

```
Device# show openflow switch 1 controller

Logical Switch Id: 1
Total Controllers: 1
Controller: 1
10.10.23.200:6633
Protocol: tcp
VRF: Mgmt-vrf
```

```
Connected: Yes
Role: Equal
Negotiated Protocol Version: OpenFlow 1.3
Last Alive Ping: 2018-06-04 17:59:20 PDT
state: ACTIVE
sec_since_connect: 50
```

ステップ 4 show openflow switch 1 ports

OpenFlow スイッチのポートに関する情報を表示します。

例：

```
Device# show openflow switch 1 ports
Logical Switch Id: 1
Port    Interface Name   Config-State     Link-State  Features
   1           Gi1/0/1        PORT_UP        LINK_UP  1GB-FD
   2           Gi1/0/2        PORT_UP        LINK_UP  1GB-FD
   3           Gi1/0/3        PORT_UP        LINK_UP  1GB-FD
   4           Gi1/0/4        PORT_UP        LINK_UP  1GB-FD
   5           Gi1/0/5        PORT_UP      LINK_DOWN  1GB-HD
   6           Gi1/0/6        PORT_UP      LINK_DOWN  1GB-HD
   7           Gi1/0/7        PORT_UP      LINK_DOWN  1GB-HD
   8           Gi1/0/8        PORT_UP      LINK_DOWN  1GB-HD
   9           Gi1/0/9        PORT_UP        LINK_UP  1GB-FD
  10          Gi1/0/10        PORT_UP        LINK_UP  1GB-FD
  11          Gi1/0/11        PORT_UP        LINK_UP  1GB-FD
  12          Gi1/0/12        PORT_UP        LINK_UP  1GB-FD
  13          Gi1/0/13        PORT_UP      LINK_DOWN  1GB-HD
  14          Gi1/0/14        PORT_UP      LINK_DOWN  1GB-HD
  15          Gi1/0/15        PORT_UP      LINK_DOWN  1GB-HD
  16          Gi1/0/16        PORT_UP      LINK_DOWN  1GB-HD
  17          Gi1/0/17        PORT_UP      LINK_DOWN  1GB-HD
  18          Gi1/0/18        PORT_UP      LINK_DOWN  1GB-HD
  19          Gi1/0/19        PORT_UP        LINK_UP  1GB-FD
  20          Gi1/0/20        PORT_UP        LINK_UP  1GB-FD
  21          Gi1/0/21        PORT_UP        LINK_UP  1GB-FD
  22          Gi1/0/22        PORT_UP        LINK_UP  1GB-FD
  23          Gi1/0/23        PORT_UP      LINK_DOWN  1GB-HD
  24          Gi1/0/24        PORT_UP      LINK_DOWN  1GB-HD
  25           Gi1/1/1        PORT_UP      LINK_DOWN  1GB-HD
  26           Gi1/1/2        PORT_UP      LINK_DOWN  1GB-HD
  27           Gi1/1/3        PORT_UP      LINK_DOWN  1GB-HD
  28           Gi1/1/4        PORT_UP      LINK_DOWN  1GB-HD
  29           Te1/1/1        PORT_UP      LINK_DOWN  10GB-FD
  30           Te1/1/2        PORT_UP      LINK_DOWN  10GB-FD
  31           Te1/1/3        PORT_UP      LINK_DOWN  10GB-FD
  32           Te1/1/4        PORT_UP      LINK_DOWN  10GB-FD
  33           Te1/1/5        PORT_UP      LINK_DOWN  10GB-FD
  34           Te1/1/6        PORT_UP      LINK_DOWN  10GB-FD
  35           Te1/1/7        PORT_UP      LINK_DOWN  10GB-FD
  36           Te1/1/8        PORT_UP      LINK_DOWN  10GB-FD
  37           Fo1/1/1        PORT_UP      LINK_DOWN  40GB-FD
  38           Fo1/1/2        PORT_UP      LINK_DOWN  40GB-FD
  39          Twe1/1/1        PORT_UP      LINK_DOWN  10GB-FD
  40          Twe1/1/2        PORT_UP      LINK_DOWN  10GB-FD
```

ステップ 5 show openflow switch 1 flows list

OpenFlow のエントリを表示します。

次の出力例は、テーブル 0 にインストールされているフローを示しています。match any はテーブル 1 に移動します（「match any」とは、すべてのパケットがテーブル 1 に移動するという意味です）。テーブル 1 では、宛先 MAC アドレス 00:00:01:00:00:01 が照合され、出力ポートが 36 に設定されます。

例：

```
Device# show openflow switch 1 flows list

Logical Switch Id: 1
Total flows: 8

Flow: 1 Match: any Actions: goto_table:1, Priority: 9000, Table: 0, Cookie: 0x1,
Duration: 2382.117s, Packets: 34443, Bytes: 3359315

Flow: 2 Match: any Actions: drop, Priority: 0, Table: 0, Cookie: 0x0,
Duration: 2382.118s, Packets: 294137, Bytes: 28806211

Flow: 3 Match: any Actions: drop, Priority: 0, Table: 1, Cookie: 0x0,
Duration: 2382.118s, Packets: 34443, Bytes: 3359315

Flow: 4 Match: dl_dst=00:00:01:00:00:01 Actions: output:36, Priority: 9000,
Table: 1, Cookie: 0x1, Duration: 2382.116s, Packets: 0, Bytes: 0
```

OpenFlow の設定例

例：デバイスでの OpenFlow の有効化

```
Device# configure terminal
Device(config)# boot mode openflow
Device(config)# exit
Device# write erase
Device# reload
Device> enable
Device# show boot mode
```

例：OpenFlow の設定

```
Device# configure terminal
Device(config)# feature openflow
Device(config)# openflow
Device(config-openflow)# switch 1 pipeline 1
Device(config-openflow-switch)# controller ipv4 10.2.2.2 port 6633 vrf Mgmt-vrf security
 tls
Device(config-openflow-switch)# datapath-id 0x12345678
Device(config-openflow-switch)# tls trustpoint local trustpoint1 remote trustpoint1
Device(config-openflow-switch)# end
```

OpenFlow に関するその他の参考資料

関連資料

関連項目	マニュアル タイトル
OpenFlow のコマンド	プログラマビリティ コマンド リファレンス
Open Network Foundation	https://www.opennetworking.org/
Faucet OpenFlow コントローラ	• https://faucet.nz/ • https://docs.faucet.nz/en/latest/

MIB

MIB	MIB のリンク
	選択したプラットフォーム、Cisco IOS リリース、およびフィーチャ セットに関する MIB を探してダウンロードするには、次の URL にある Cisco MIB Locator を使用します。 http://www.cisco.com/go/mibs

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

OpenFlow 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 *34 : OpenFlow* の機能情報

機能名	リリース	機能情報
OpenFlow	Cisco IOS XE Fuji 16.9.1	OpenFlow は Software Defined Networking（SDN）の標準規格であり、SDN 環境での通信プロトコルを定義します。これにより、SDN コントローラは、スイッチやルータなどのネットワーク デバイスのフォワーディング プレーンと直接やり取りできるようになります。 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。 • Catalyst 9300 シリーズ スイッチ • Catalyst 9400 シリーズ スイッチ • Catalyst 9500 シリーズ スイッチ • Catalyst 9500 シリーズ ハイ パフォーマンス スイッチ
	Cisco IOS XE ジブラルタル 16.10.1	Catalyst 9500 シリーズ ハイ パフォーマンス スイッチでのテーブル機能メッセージのサポートが導入されました。