



Cisco Nexus 3500 Series NX-OS Programmability Guide, Release 10.5(x)

First Published: 2024-07-26

Last Modified: 2024-11-27

Americas Headquarters

Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
<http://www.cisco.com>
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS REFERENCED IN THIS DOCUMENTATION ARE SUBJECT TO CHANGE WITHOUT NOTICE. EXCEPT AS MAY OTHERWISE BE AGREED BY CISCO IN WRITING, ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS DOCUMENTATION ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED.

The Cisco End User License Agreement and any supplemental license terms govern your use of any Cisco software, including this product documentation, and are located at: <https://www.cisco.com/c/en/us/about/legal/cloud-and-software/software-terms.html>. Cisco product warranty information is available at <https://www.cisco.com/c/en/us/products/warranty-listing.html>. US Federal Communications Commission Notices are found here <https://www.cisco.com/c/en/us/products/us-fcc-notice.html>.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Any products and features described herein as in development or available at a future date remain in varying stages of development and will be offered on a when-and if-available basis. Any such product or feature roadmaps are subject to change at the sole discretion of Cisco and Cisco will have no liability for delay in the delivery or failure to deliver any products or feature roadmap items that may be set forth in this document.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

The documentation set for this product strives to use bias-free language. For the purposes of this documentation set, bias-free is defined as language that does not imply discrimination based on age, disability, gender, racial identity, ethnic identity, sexual orientation, socioeconomic status, and intersectionality. Exceptions may be present in the documentation due to language that is hardcoded in the user interfaces of the product software, language used based on RFP documentation, or language that is used by a referenced third-party product.

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: <https://www.cisco.com/c/en/us/about/legal/trademarks.html>. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1721R)

© 2024 Cisco Systems, Inc. All rights reserved.



CONTENTS

PREFACE

はじめに xvii

対象読者 xvii

表記法 xvii

Cisco Nexus 3000 シリーズ スイッチの関連資料 xviii

マニュアルに関するフィードバック xix

通信、サービス、およびその他の情報 xix

CHAPTER 1

New and Changed Information 1

New and Changed Information 1

CHAPTER 2

Overview 3

ライセンス要件 3

サポートされるプラットフォーム 3

CHAPTER 3

bash 5

Bash について 5

Bash へのアクセス 5

権限をルートにエスカレーションする 6

Bash コマンドの例 7

システム統計情報の表示 7

CLI からの Bash の実行 8

Bash からの Python の実行 8

Kstack を介してコピー 9

CHAPTER 4

ゲスト シェル 11

Guest Shell について 11

注意事項と制約事項 12

Guestshell に関する注意事項と制限事項 17

Guestshell に関する注意事項と制限事項 24

Guest Shell へのアクセス 31

ゲスト シェルに使用されるリソース 32

ゲストシェルの機能 32

Guest Shell の NX-OS CLI 33

Guest Shell でのネットワーク アクセス 33

ゲスト シェルでのブートフラッシュへのアクセス 36

Guest Shell の Python 36

Guest Shell バージョン 2.10 までの Python 3 (CentOS 7) 37

Guest Shell バージョン 2.10 までの Python 3 (CentOS 7) 39

ゲスト シェルでの RPM のインストール 42

仮想サービス ゲスト シェルのセキュリティ ポスチャ 44

[デジタル署名されたアプリケーションパッケージ (Digitally Signed Application Packages)]
44

[カーネル脆弱性パッチ (Kernel Vulnerability Patches)] 45

[ASLR および X-Space のサポート (ASLR and X-Space Support)] 45

名前空間の分離 45

名前空間の分離 46

ルートユーザーの制限 47

リソース管理 48

ゲスト ファイル システムのアクセス制限 48

ゲスト シェルの管理 49

Guest Shell の無効化 53

ゲスト シェルの破棄 54

Guest Shell の有効化 54

ゲスト シェルの複製 56

ゲスト シェル rootfs のエクスポート	56
Guest Shell rootfs のインポート	57
YAML ファイルのインポート	59
show guestshell コマンド	62
仮想サービスと Guest Shell 情報の検証	62
ゲスト シェルからのアプリケーションの永続的な起動	64
Guest Shell からアプリケーションを永続的に起動する手順	65
ゲスト シェルでのサンプル アプリケーション	65
Guest Shell に関する問題のトラブルシューティング	67

CHAPTER 5

Python API 69

Python API の概要	69
Python の使用	70
Cisco Python パッケージ	70
CLI コマンド API の使用	71
CLI からの Python インタープリタの呼び出し	73
表示フォーマット	74
非インタラクティブ Python	75
Embedded Event Manager でのスクリプトの実行	77
Cisco NX-OS ネットワーク インターフェイスとの Python 統合	78
Python による Cisco NX-OS セキュリティ	78
セキュリティとユーザー権限の例	78
Scheduler でのスクリプトの実行例	79

CHAPTER 6

Python API 81

[Python API について (About the Python API)]	81
Python の使用	82
Cisco Python パッケージ	82
CLI コマンド API の使用	83
CLI からの Python インタープリタの呼び出し	85
表示フォーマット	85

非インタラクティブ Python	87
Embedded Event Manager でのスクリプトの実行	88
Cisco NX-OS ネットワーク インターフェイスとの Python 統合	89
Python による Cisco NX-OS セキュリティ	90
セキュリティとユーザー権限の例	90
スケジューラでスクリプトを実行する例	91

CHAPTER 7**tcl によるスクリプティング** 95

Tcl について	95
telsh コマンドのヘルプ	95
telsh コマンドの履歴	96
telsh のタブ補完	96
telsh の CLI コマンド	96
telsh コマンドの区切り	97
tcl 変数	97
tclquit	97
Telsh セキュリティ	97
Telsh コマンドの実行	98
Telsh コマンドからの Cisco NX-OS モード間の移動	99
tcl の参照	100

CHAPTER 8**Ansible** 103

前提条件	103
アンシブルについて	103
Cisco Ansible モジュール	104

CHAPTER 9**Puppet Agent** 105

Puppet について	105
前提条件	106
Puppet エージェント NX-OS 環境	106
ciscopuppet モジュール	106

CHAPTER 10**SaltStack 109**

- SaltStack について 109
 - NX-OS と SaltStack について 110
- 注意事項と制約事項 110
- SaltStack の Cisco NX-OS 環境 111
- SaltStack の NX-API の有効化 111
- NX-OS 用の SaltStack のインストール 112

CHAPTER 11**Cisco NX-OS でのシェフ クライアントの使用 113**

- シェフについて 113
- 前提条件 114
- Chef クライアント NX-OS 環境 114
- cisco-cookbook 115

CHAPTER 12**Cisco NX-OS での Docker の使用 117**

- Cisco NX-OS での Docker について 117
- 注意事項と制約事項 118
- Cisco NX-OS 内で Docker コンテナを設定するための前提条件 118
- Docker デーモンの開始 119
- 自動的に起動するように Docker を構成する 119
- Docker コンテナの開始: ホスト ネットワーク モデル 120
- Docker コンテナの開始: ブリッジ型ネットワーク モデル 121
- Docker コンテナでのブートフラッシュおよび揮発性パーティションのマウント 122
- 拡張 ISSU スイッチオーバーでの Docker デーモンの永続性の有効化 123
- Cisco Nexus Platform Switches Switchover 時に Docker デーモンの永続性を有効にする 124
- Docker ストレージ バックエンドのサイズ変更 125
- Docker デーモンの停止 127
- Docker コンテナ セキュリティ 128
 - ユーザー[名前空間 (namespace)]の分離による Docker コンテナの保護 128
- cgroup パーティションの移動 129

Kubernetes クラスタにノードを追加	130
Docker のトラブルシューティング	132
Docker の起動が機能不全になる	132
ストレージが不足しているため、Docker が起動に失敗する	133
Docker Hub からのイメージのプルの失敗（509 証明書失効 エラー メッセージ）	133
Docker Hub からのイメージのプルの失敗（クライアント タイムアウト エラー メッセージ）	134
スイッチのリロードまたはスイッチオーバーで Docker デーモンまたはコンテナが実行されない	135
Docker ストレージバックエンドのサイズ変更が失敗する	135
Docker コンテナがポートで着信トラフィックを受信しない	135
Docker コンテナでデータ ポートと / または管理インターフェイスを表示できません	136
一般的なトラブルシューティングのヒント	136

CHAPTER 13
NX-SDK 137

NX-SDK について	137
Go バインディングに関する考慮事項	138
オンボックス（ローカル）アプリケーションについて	139
デフォルト Docker イメージ	139
NX-SDK に関する注意事項と制限事項	139
オフボックス（リモート）アプリケーション	140
NX-SDK2.0 について	140
リモート アプリケーションについて	141
NX-SDK セキュリティ	142
NX SDK 2.0 のセキュリティ プロファイル	142

CHAPTER 14
NX-API 145

NX-APIについて	145
機能 NX-API	146
転送	146
メッセージ形式	146
セキュリティ	147

NX-API の使用	148
NX-API 管理コマンド	151
NX-API を使用したインタラクティブ コマンドの操作	153
NX-API リクエスト要素	153
NX-API 応答要素	163
JSON の概要（JavaScript オブジェクト表記）	164
JSON に関する注意事項	165
CLI の実行	165
XML および JSON でサポートされたコマンド	165
XML および JSON 出力の例	166
JSON-RPC、JSON、および XML でサポートされるコマンド	174
XML および JSON 出力の例	175

CHAPTER 15

NX-API 応答コード	179
NX-API 応答コードの表	179

CHAPTER 16

NX-API 開発者サンドボックス	183
NX-API 開発者サンドボックス: 9.2（2）より前の NX-OS リリース	183
About the NX-API デベロッパー サンドボックス	183
注意事項と制約事項	184
メッセージフォーマットとコマンドタイプの構成	184
デベロッパー サンドボックスを使用	187
デベロッパー サンドボックスを使用して CLI コマンドをペイロードに変換する	187
NX-API 開発者サンドボックス: NX-OS リリース 9.2（2）以降	190
About the NX-API デベロッパー サンドボックス	190
注意事項と制約事項	192
メッセージフォーマットと入力タイプの構成	194
デベロッパー サンドボックスを使用	197
デベロッパー サンドボックスを使用して CLI コマンドを REST ペイロードに変換する	198
デベロッパー サンドボックスを使用した REST ペイロードから CLI コマンドへの変換	201

デベロッパー サンドボックスを使用して RESTCONF から json または XML に変換する
207

CHAPTER 17	N3500 での ABM および LM の XML サポート	211
	N3500 での ABM および LM の XML サポート	211

CHAPTER 18	CLI コマンドのネットワーク構成フォーマットへの変換	219
	XMLIN に関する情報	219
	XMLIN のライセンス要件	219
	XMLIN ツールのインストールおよび使用	220
	show コマンド出力の XML への変換	221
	XMLIN の構成例	221

CHAPTER 19	OpenConfig YANG	225
	About OpenConfig YANG	225
	Guidelines and Limitations for OpenConfig YANG	225
	Understanding Deletion of BGP Routing Instance	233
	Verifying YANG	235
	Enabling OpenConfig Support	235

CHAPTER 20	XML 管理インターフェイス	237
	XML 管理インターフェイスについて	237
	XML 管理インターフェイスについて	237
	NETCONF レイヤ	238
	SSH xmlagent	238
	XML 管理インターフェイスのライセンス要件	239
	XML 管理インターフェイスを使用するための前提条件	239
	XML 管理インターフェイスを使用	239
	SSH および XML サーバ オプションの構成	239
	SSH セッションを開始	240
	Hello メッセージを送信	240
	XSD ファイルの取得	241

XML ドキュメントを XML サーバに送信する	241
NETCONF XML インスタンスの作成	242
RPC リクエスト タグ <code>rpc</code>	242
NETCONF 動作タグ	243
デバイスタグ	245
拡張された NETCONF の操作	247
NETCONF 応答	251
RPC 応答タグ	251
データタグにカプセル化されたタグの解釈	251
サンプル XML インスタンスに関する情報	252
XML インスタンスの例	252
NETCONF クローズ セッション インスタンス	253
NETCONF キルセッション インスタンス	253
NETCONF <code>copy-config</code> インスタンス	253
NETCONF <code>edit-config</code> インスタンス	254
NETCONF <code>get-config</code> インスタンス	255
NETCONF ロック インスタンス	256
NETCONF ロック解除インスタンス	256
NETCONF コミット インスタンス - 候補構成機能	257
NETCONF <code>Confirmed-commit</code> インスタンス	257
NETCONF <code>rollback-on-error</code> インスタンス	258
NETCONF 検証機能インスタンス	258
その他の参考資料	259

PART I
Model-Driven Programmability 261

CHAPTER 21
Managing Components 263

コンポーネント RPM パッケージについて	263
インストールの準備	266
Cisco Artifactory からのコンポーネントのダウンロード	267
Installing RPM Packages	267

プログラマブルインターフェイスおよび共通モデルコンポーネント RPM パッケージのインストール	267

CHAPTER 22 **Converting CLI Commands to Network Configuration Format** 269

XMLIN に関する情報	269
XMLIN のライセンス要件	269
XMLIN ツールのインストールおよび使用	270
show コマンド出力の XML への変換	271
XMLIN の構成例	271

CHAPTER 23 **gNMI - gRPC Network Management Interface** 275

About gNMI	275
gNMI RPC and SUBSCRIBE	276
gNMI に関する注意事項と制限事項	277
Configuring gNMI	280
gNMI - gRPC Network Management Interface	282
サーバー証明書の構成	282
Generating Key/Certificate Examples	283
キー/証明書の生成の例	284
Cisco NX-OS リリース 9.3(3) 以降のキー/証明書の生成と構成の例	285
Enabling gNMI	287
Verifying gNMI	288
gRPC クライアント証明書認証	293
新しいクライアントルート CA 証明書の生成	293
NX-OS デバイスでの生成されたルート CA 証明書の構成	294
gRPC へのトラストポイントの関連付け	294
証明書の詳細の検証	295
任意の gNMI クライアントのクライアント証明書認証を使用した接続の確認	296
Clients	297
Sample DME Subscription - JSON Encoding	297
Sample DME Subscription - PROTO Encoding	298
Capabilities	299

About Capabilities	299
Guidelines and Limitations for Capabilities	299
Example Client Output for Capabilities	300
Get	303
About Get	303
Guidelines and Limitations for Get	303
Set	304
About Set	304
Guidelines and Limitations for Set	304
Subscribe	305
Guidelines and Limitations for Subscribe	305
gNMI Payload	306
Streaming Syslog	309
About Streaming Syslog for gNMI	309
Guidelines and Limitations for Streaming Syslog - gNMI	309
Syslog Native YANG Model	310
Subscribe Request Example	310
Sample PROTO Output	311
Sample JSON Output	314
Troubleshooting	315
Gathering TM-Trace Logs	315
Gathering MTX-Internal Logs	315

CHAPTER 24

gNOI-gRPC ネットワーク操作インターフェイス	319
gNOI について	319
サポートされる gNOI RPC	320
システム proto	320
OS プロトコル	322
証明書 Proto	323
ファイル Proto	323
gNOI 工場リセット	324
注意事項と制約事項	325
gNOI の確認	326

CHAPTER 25**Model Driven Telemetry 327****About Telemetry 327****Telemetry Components and Process 327****High Availability of the Telemetry Process 329****Licensing Requirements for Telemetry 329****Installing and Upgrading Telemetry 329****Guidelines and Limitations for Model Driven Telemetry 330****Configuring Telemetry Using the CLI 337****Configuring Telemetry Using the NX-OS CLI 337****Configuring Cadence for YANG Paths 342****Configuration Examples for Telemetry Using the CLI 344****Displaying Telemetry Configuration and Statistics 348****Displaying Telemetry Log and Trace Information 356****Configuring Telemetry Using the NX-API 358****Configuring Telemetry Using the NX-API 358****Configuration Example for Telemetry Using the NX-API 368****Telemetry Model in the DME 371****Telemetry Path Labels 372****About Telemetry Path Labels 372****Polling for Data or Receiving Events 373****Guidelines and Limitations for Path Labels 373****Configuring the Interface Path to Poll for Data or Events 373****Configuring the Interface Path for Non-Zero Counters 375****Configuring the Interface Path for Operational Speeds 377****Configuring the Interface Path with Multiple Queries 379****Configuring the Environment Path to Poll for Data or Events 380****Configuring the Resources Path for Poll for Events or Data 382****Configuring the VXLAN Path to Poll for Events or Data 384****Verifying the Path Label Configuration 385****Displaying Path Label Information 386****Native Data Source Paths 389****About Native Data Source Paths 389****Telemetry Data Streamed for Native Data Source Paths 389**

Guidelines and Limitations for Native Data Source Path	391
Configuring the Native Data Source Path for Routing Information	392
Configuring the Native Data Source Path for MAC Information	394
Configuring the Native Data Source Path for All MAC Information	396
Configuring the Native Data Path for IP Adjacencies	398
Displaying Native Data Source Path Information	400
Streaming Syslog	401
About Streaming Syslog for Telemetry	401
Configuring the Native Data Source Path for Routing Information	401
Telemetry Data Streamed for Syslog Path	404
Sample JSON Output	405
Sample KVGPB Output	405
Additional References	408
Related Documents	408

APPENDIX A

ストリーミング テレメトリの送信元	409
ストリーミング テレメトリについて	409
注意事項と制約事項	409
テレメトリで利用可能なデータ	409



はじめに

この前書きは、次の項で構成されています。

- [対象読者 \(xvii ページ\)](#)
- [表記法 \(xvii ページ\)](#)
- [Cisco Nexus 3000 シリーズ スイッチの関連資料 \(xviii ページ\)](#)
- [マニュアルに関するフィードバック \(xix ページ\)](#)
- [通信、サービス、およびその他の情報 \(xix ページ\)](#)

対象読者

このマニュアルは、Cisco Nexus スイッチの設置、設定、および維持に携わるネットワーク管理者を対象としています。

表記法

コマンドの説明には、次のような表記法が使用されます。

表記法	説明
bold	太字の文字は、表示どおりにユーザが入力するコマンドおよびキーワードです。
<i>italic</i>	イタリック体の文字は、ユーザが値を入力する引数です。
[x]	省略可能な要素（キーワードまたは引数）は、角かっこで囲んで示しています。
[x y]	いずれか 1 つを選択できる省略可能なキーワードや引数は、角カッコで囲み、縦棒で区切って示しています。
{x y}	必ずいずれか 1 つを選択しなければならない必須キーワードや引数は、波かっこで囲み、縦棒で区切って示しています。

表記法	説明
[x {y z}]	角かっこまたは波かっこが入れ子になっている箇所は、任意または必須の要素内の任意または必須の選択肢であることを表します。角かっこ内の波かっこと縦棒は、省略可能な要素内で選択すべき必須の要素を示しています。
variable	ユーザが値を入力する変数であることを表します。イタリック体不能使用できない場合に使用されます。
string	引用符を付けない一組の文字。string の前後には引用符を使用しません。引用符を使用すると、その引用符も含めて string とみなされます。

例では、次の表記法を使用しています。

表記法	説明
screen フォント	スイッチが表示する端末セッションおよび情報は、スクリーンフォントで示しています。
太字の screen フォント	ユーザが入力しなければならない情報は、太字のスクリーンフォントで示しています。
イタリック体の screen フォント	ユーザが値を指定する引数は、イタリック体の screen フォントで示しています。
<>	パスワードのように出力されない文字は、山カッコ (<>) で囲んで示しています。
[]	システム プロンプトに対するデフォルトの応答は、角カッコで囲んで示しています。
!、#	コードの先頭に感嘆符 (!) またはポンド記号 (#) がある場合には、コメント行であることを示します。

Cisco Nexus 3000 シリーズ スイッチの関連資料

Cisco Nexus 3000 シリーズ スイッチ全体のマニュアルセットは、次の URL にあります。

<https://www.cisco.com/c/en/us/support/switches/nexus-3000-series-switches/tsd-products-support-series-home.html>

マニュアルに関するフィードバック

このマニュアルに関する技術的なフィードバック、または誤りや記載もれなどお気づきの点がございましたら、HTML ドキュメント内のフィードバック フォームよりご連絡ください。ご協力をよろしくお願いいたします。

通信、サービス、およびその他の情報

- シスコからタイムリーな関連情報を受け取るには、[Cisco Profile Manager](#) でサインアップしてください。
- 重要な技術によって求めるビジネス成果を得るには、[Cisco Services](#) [英語] にアクセスしてください。
- サービスリクエストを送信するには、[Cisco Support](#) [英語] にアクセスしてください。
- 安全で検証済みのエンタープライズクラスのアプリケーション、製品、ソリューション、およびサービスを探して参照するには、[Cisco DevNet](#) [英語] にアクセスしてください。
- 一般的なネットワーク、トレーニング、認定関連の出版物を入手するには、[Cisco Press](#) [英語] にアクセスしてください。
- 特定の製品または製品ファミリの保証情報を探すには、[Cisco Warranty Finder](#) にアクセスしてください。

シスコバグ検索ツール

[シスコバグ検索ツール](#) (BST) は、シスコ製品とソフトウェアの障害と脆弱性の包括的なリストを管理するシスコバグ追跡システムへのゲートウェイとして機能する、Web ベースのツールです。BST は、製品とソフトウェアに関する詳細な障害情報を提供します。



CHAPTER 1

New and Changed Information

- [New and Changed Information](#), on page 1

New and Changed Information

This table summarizes the new and changed features for the *Cisco Nexus 3500 Series NX-OS Programmability Guide, Release 10.5(x)* and where they are documented.

Table 1: New and Changed Features

Feature	Description	Changed in Release	Where Documented
NA	No new features added for this release.	10.5(1)F	NA



CHAPTER 2

Overview

- [ライセンス要件](#) (3 ページ)
- [サポートされるプラットフォーム](#) (3 ページ)

ライセンス要件

Cisco NX-OS ライセンス方式の推奨の詳細と、ライセンスの取得および適用の方法については、『[Cisco NX-OS ライセンス ガイド](#)』および『[Cisco NX-OS ライセンス オプション ガイド](#)』を参照してください。

サポートされるプラットフォーム

Cisco NX-OS リリース 7.0(3)I7(1) 以降では、[Nexus スイッチ プラットフォーム サポート マトリクス](#)に基づいて、選択した機能をさまざまな Cisco Nexus 9000 および 3000 スイッチで使用するために、どの Cisco NX-OS リリースが必要かを確認してください。



第 3 章

bash

- [Bash について](#) (5 ページ)
- [Bash へのアクセス](#) (5 ページ)
- [権限をルートにエスカレーションする](#) (6 ページ)
- [Bash コマンドの例](#) (7 ページ)
- [Kstack を介してコピー](#) (9 ページ)

Bash について

Cisco NX-OS CLI に加えて、Cisco Nexus 3500 プラットフォーム スイッチは Bourne-Again SHell (Bash) へのアクセスをサポートします。Bash は、ユーザーが入力したコマンドまたはシェルスクリプトから読み取られたコマンドを解釈します。Bash を使用すると、デバイス上の基盤となる Linux システムにアクセスしてシステムを管理できます。

Bash へのアクセス

Cisco NX-OS では、Cisco NX-OS dev-ops ロールまたは Cisco NX-OS network-admin ロールに関連付けられたユーザ アカウントから Bash にアクセスできます。

次の例は、dev-ops ロールと network-admin ロールの権限を示しています。

```
switch# show role name dev-ops
```

```
Role: dev-ops
```

```
Description: Predefined system role for devops access. This role cannot be modified.
```

```
Vlan policy: permit (default)
```

```
Interface policy: permit (default)
```

```
Vrf policy: permit (default)
```

Rule	Perm	Type	Scope	Entity
4	permit	command		conf t ; username *
3	permit	command		bcm module *
2	permit	command		run bash *
1	permit	command		python *

```
switch# show role name network-admin

Role: network-admin
Description: Predefined network admin role has access to all commands
on the switch
-----
Rule      Perm      Type      Scope      Entity
-----
1         permit   read-write
switch#
```

feature bash-shell コマンドを実行すると、Bash が有効になります。

この **run bash** コマンドは Bash を読み込み、ユーザーのホーム ディレクトリから開始します。

次の例は、Bash シェル機能を有効にする方法と、Bash を実行する方法を示しています。

```
switch# configure terminal
switch(config)# feature bash-shell

switch# run bash
Linux# whoami
admin
Linux# pwd
/bootflash/home/admin
Linux#
```



(注) **run bash** <[コマンド (command)]> コマンドで Bash コマンドを実行することもできます。

以下は **run bash** <[コマンド (command)]> コマンドの例です。

```
run bash whoami
```

権限をルートにエスカレーションする

管理者ユーザーの特権は、ルート アクセスの特権をエスカレーションできます。

以下は、権限をエスカレーションするためのガイドラインです：

- 特権を root にエスカレーションできるのは管理者ユーザーのみです。
- 権限をエスカレーションする前に、Bash を有効にする必要があります。
- root へのエスカレーションはパスワードで保護されています。
- 非管理インターフェイスを介した root ユーザー名を使用したスイッチへの SSH では、root ユーザーの Linux Bash シェル タイプ アクセスがデフォルトになります。NX-OS シェル アクセスに戻るために **vsh** を入力します。

次の例は、特権を root にエスカレーションする方法と、エスカレーションを確認する方法を表示しています。

```
switch# run bash
Linux# sudo su root
```

```
We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:
```

- #1) Respect the privacy of others.
- #2) Think before you type.
- #3) With great power comes great responsibility.

```
Password:
```

```
Linux# whoami
root
Linux# exit
exit
```

Bash コマンドの例

このセクションには、Bash コマンドと出力の例が含まれています。

システム統計情報の表示

次に、システム統計情報の表示方法の例を示します：

```
switch# run bash
Linux# cat /proc/meminfo
MemTotal:      3795100 kB
MemFree:       1472680 kB
Buffers:       136 kB
Cached:        1100116 kB
ShmFS:         1100116 kB
Allowed:       948775 Pages
Free:          368170 Pages
Available:     371677 Pages
SwapCached:    0 kB
Active:        1198872 kB
Inactive:      789764 kB
SwapTotal:     0 kB
SwapFree:      0 kB
Dirty:         0 kB
Writeback:     0 kB
AnonPages:     888272 kB
Mapped:        144044 kB
Slab:          148836 kB
SReclaimable:  13892 kB
SUnreclaim:    134944 kB
PageTables:    28724 kB
NFS_Unstable:  0 kB
Bounce:        0 kB
WritebackTmp:  0 kB
CommitLimit:   1897548 kB
Committed_AS:  19984932 kB
VmallocTotal:  34359738367 kB
VmallocUsed:    215620 kB
VmallocChunk:  34359522555 kB
HugePages_Total: 0
HugePages_Free: 0
HugePages_Rsvd: 0
```

```
HugePages_Surp:      0
Hugepagesize:       2048 kB
DirectMap4k:        40960 kB
DirectMap2M:        4190208 kB
Linux#
```

CLI からの Bash の実行

次の例は、CLI から **run bash <command>** コマンドを使用して **bash** コマンドを実行する方法を示しています。

```
switch# run bash ps -el
F S  UID  PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S   0    1    0  0  80   0 -   497 select ?        00:00:08 init
5 S   0    2    0  0  75  -5 -    0 kthrea ?        00:00:00 kthreadd
1 S   0    3    2  0 -40   - -    0 migrat ?        00:00:00 migration/0
1 S   0    4    2  0  75  -5 -    0 ksofti ?        00:00:01 ksoftirqd/0
5 S   0    5    2  0  58   - -    0 watchd ?        00:00:00 watchdog/0
1 S   0    6    2  0 -40   - -    0 migrat ?        00:00:00 migration/1
1 S   0    7    2  0  75  -5 -    0 ksofti ?        00:00:00 ksoftirqd/1
5 S   0    8    2  0  58   - -    0 watchd ?        00:00:00 watchdog/1
1 S   0    9    2  0 -40   - -    0 migrat ?        00:00:00 migration/2
1 S   0   10    2  0  75  -5 -    0 ksofti ?        00:00:00 ksoftirqd/2
5 S   0   11    2  0  58   - -    0 watchd ?        00:00:00 watchdog/2
1 S   0   12    2  0 -40   - -    0 migrat ?        00:00:00 migration/3
1 S   0   13    2  0  75  -5 -    0 ksofti ?        00:00:00 ksoftirqd/3
5 S   0   14    2  0  58   - -    0 watchd ?        00:00:00 watchdog/3

...

4 S   0  8864    1  0  80   0 -  2249 wait  ttyS0    00:00:00 login
4 S  2002 28073  8864  0  80   0 - 69158 select ttyS0    00:00:00 vsh
4 R   0 28264  3782  0  80   0 - 54790 select ?        00:00:00 in.dcos-telnet
4 S   0 28265 28264  0  80   0 -  2247 wait  pts/0    00:00:00 login
4 S  2002 28266 28265  0  80   0 - 69175 wait  pts/0    00:00:00 vsh
1 S  2002 28413 28266  0  80   0 - 69175 wait  pts/0    00:00:00 vsh
0 R  2002 28414 28413  0  80   0 -   887 -      pts/0    00:00:00 ps
switch#
```

Bash からの Python の実行

次の例は、Python をロードし、Python オブジェクトを使用してスイッチを構成する方法を示しています。

```
switch# run bash
Linux# python
Python 2.7.5 (default, May 16 2014, 10:58:01)
[GCC 4.3.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
Loaded cisco NxOS lib!
>>>
>>> from cisco import *
>>> from cisco.vrf import *
>>> from cisco.interface import *
>>> vrfobj=VRF('myvrf')
>>> vrfobj.get_name()
'myvrf'
>>> vrfobj.add_interface('Ethernet1/3')
True
```

```
>>> intf=Interface('Ethernet1/3')
>>> print intf.config()

!Command: show running-config interface Ethernet1/3
!Time: Thu Aug 21 23:32:25 2014

version 6.0(2)U4(1)

interface Ethernet1/3
  no switchport
  vrf member myvrf

>>>
```

Kstack を介してコピー

Cisco NX-OS リリース 9.3(1) 以降では、ファイル コピー操作には、**use-kstack** オプションを使用して別のネットワーク スタックを介して実行するオプションがあります。**use-kstack** を通じてファイルをコピーすると、コピー時間が短縮されます。このオプションは、スイッチから複数のホップにあるリモート サーバーからファイルをコピーする場合に役立ちます。**use-kstack** オプションは、**scp** や **sftp** などの標準ファイル コピー機能を通じてスイッチに、またはスイッチからファイルをコピー処理します。



- (注) スイッチが FIPS モード機能を実行している場合、**use-kstack** オプションは機能しません。スイッチで FIPS モードが有効になっている場合、コピー操作は引き続き成功しますが、デフォルトのコピー方法が使用されます。

use-kstack を介してコピーするには、NX-OS **copy** コマンドの最後に引数を追加します。たとえば:

```
switch-1# copy scp://test@10.1.1.1/image.bin . vrf management use-kstack
switch-1#
switch-1# copy scp://test@10.1.1.1/image.bin bootflash:// vrf management
use-kstack
switch-1#
switch-1# copy scp://test@10.1.1.1/image.bin . use-kstack
switch-1#
switch-1# copy scp://test@10.1.1.1/image.bin bootflash:// vrf default
use-kstack
switch-1#
```

use-kstack オプションは、すべての NX-OS **copy** コマンドとファイル システムでサポートされています。オプションは OpenSSL (セキュア コピー) 認定済みです。



第 4 章

ゲスト シェル

- [Guest Shell について \(11 ページ\)](#)
- [注意事項と制約事項 \(12 ページ\)](#)
- [Guestshell に関する注意事項と制限事項 \(17 ページ\)](#)
- [Guestshell に関する注意事項と制限事項 \(24 ページ\)](#)
- [Guest Shell へのアクセス \(31 ページ\)](#)
- [ゲスト シェルに使用されるリソース \(32 ページ\)](#)
- [ゲストシェルの機能 \(32 ページ\)](#)
- [仮想サービス ゲスト シェルのセキュリティ ポスチャ \(44 ページ\)](#)
- [ゲスト ファイル システムのアクセス制限 \(48 ページ\)](#)
- [ゲスト シェルの管理 \(49 ページ\)](#)
- [仮想サービスと Guest Shell 情報の検証 \(62 ページ\)](#)
- [ゲスト シェルからのアプリケーションの永続的な起動 \(64 ページ\)](#)
- [Guest Shell からアプリケーションを永続的に起動する手順 \(65 ページ\)](#)
- [ゲスト シェルでのサンプル アプリケーション \(65 ページ\)](#)
- [Guest Shell に関する問題のトラブルシューティング \(67 ページ\)](#)

Guest Shell について

基盤となる Linux 環境での NX-OS CLI および Bash アクセスに加えて、スイッチは、「ゲストシェル」と呼ばれる Linux コンテナ (LXC) 内で実行される分離された実行スペースへのアクセスをサポートします。

ゲスト シェル内から、`network-admin` には次の機能があります。

- Linux ネットワーク インターフェイスを介したネットワークへのアクセス。
- スイッチのブートフラッシュへのアクセス。
- スイッチの揮発性 `tmpfs` へのアクセス。
- スイッチの CLI へのアクセス。
- スイッチのホスト ファイル システムへのアクセス。

- Cisco NX-API REST へのアクセス。
- Python スクリプトをインストールして実行する機能。
- 32 ビットおよび 64 ビットの Linux アプリケーションをインストールして実行する機能。

コンテナ技術によって実行空間を切り離すことで、他の Linux コンテナで実行されているホストシステムやアプリケーションに影響を与えずに、アプリケーションのニーズに合わせて Linux 環境をカスタマイズすることができます。

NX-OS デバイスでは、Linux Containers は `virtual-service` コマンドでインストールと管理されます。Guest Shell は、`virtual-service show` コマンドの出力に表示されます。



- (注) デフォルトでは、ゲストシェルは、有効にすると約 5 MB の RAM と 200 MB のブートフラッシュを占有します。Cisco NX-OS リリース 7.0 (3) I2 (1) 以降、Guest Shell は約 35 MB の RAM を占有します。Guest Shell が使用されていない場合は、**guestshell destroy** コマンドを使用してリソースを再利用します。



- (注) Cisco NX-OS 7.0(3)F3(1)/NX-OS 7.0(3)I7(1)以降、Guest Shell は Cisco Nexus 95083500 スイッチでサポートされます。

注意事項と制約事項

Guest Shell には、次の注意事項と制限事項があります。

すべてのリリースに共通の注意事項



- 重要** Guest Shell のインストール内でカスタム作業を実行した場合は、アップグレードを実行する前に、ブートフラッシュ、オフボックス ストレージ、またはゲスト シェル ルート ファイル システムの外部に変更を保存します。

`guestshell upgrade` コマンドは、本質的に、`guestshell destroy` と `guestshell enable` を連続して実行します。

- `run guestshell CLI` コマンドを使用して、Cisco Nexus デバイスのゲストシェルにアクセスします。`run guestshell` コマンドは、ホスト シェルにアクセスするために使用される `run bash` コマンドに相当します。このコマンドを使用すると、Guest Shell にアクセスして `bash` プロンプトを取得したり、ゲストシェルのコンテキスト内でコマンドを実行したりできます。このコマンドは、パスワードなしの SSH を使用して、デフォルトのネットワーク名前空間にある `localhost` の使用可能なポートに接続します。

- `sshd` ユーティリティは、`localhost` でリッスンして、ネットワーク外からの接続試行を回避することにより、Guest Shell への事前構成された SSH アクセスを保護できます。sshd には次の機能があります。
 - これは、パスワードにフォールバックしないキーベースの認証用に構成されています。
 - Guest Shell の再起動後にゲスト シェルにアクセスするために使用されるキーを読み取ることができるのは `root` だけです。
 - `root` のみがホスト上のキーを含むファイルを読み取ることができ、ホスト `bash` アクセスを持つ非特権ユーザーがキーを使用して Guest Shell に接続できないようにします。ネットワーク管理者ユーザーは、Guest Shell で `sshd` の別のインスタンスを開始して、Guest Shell に直接リモート アクセスできるようにすることができますが、Guest Shell にログインするすべてのユーザーには、ネットワーク管理者権限も与えられます。



(注) ゲスト シェル 2.2 (0.2) で導入されたキー ファイルは、ユーザー アカウントが作成されたユーザーに対して読み取り可能です。

さらに、Guest Shell アカウントは自動的に削除されないため、不要になったときにネットワーク管理者が削除する必要があります。

Cisco Nexus リリース 7.0(3)I5(2)を使用した 2.2 (0.2) より前の Guest Shell インストールでは、個々のユーザー アカウントが動的に作成されません。

- すぐに使用できる新しい Cisco Nexus スイッチに Cisco Nexus シリーズ スイッチ ソフトウェア リリースをインストールすると、Guest Shell が自動的に有効になります。その後の Cisco Nexus シリーズ スイッチ ソフトウェアへのアップグレードでは、Guest Shell は自動的にアップグレードされません。
- Guest Shell のリリースでは、配布または配布のバージョンが変更されると、メジャー番号が増分されます。
- Guest Shell のリリースでは、CVE が解決されるとマイナー番号が増分されます。Guest Shell は、CentOS が公開した場合にのみ CVE を更新します。
- **yum update** を使用して、CentOS リポジトリからサードパーティのセキュリティ脆弱性修正を直接取得することをお勧めします。これにより、Cisco NX-OS ソフトウェアのアップデートを待つことなく、更新が利用可能になったときに入手できる柔軟性が得られます。

または、**guestshell update** コマンドを使用すると、既存のゲストシェル `rootfs` が置き換えられます。カスタマイズとソフトウェアパッケージのインストールは、この新しいゲストシェル `rootfs` のコンテキスト内で再度実行する必要があります。

Guest Shell 1.0 からゲスト シェル 2.x へのアップグレード

Guest Shell 2.x は、CentOS 7 ルート ファイル システムに基づいています。コンテンツを Guest Shell 1.0 にプルダウンした .conf ファイルやユーティリティのオフボックス リポジトリがある場合は、Guest Shell 2.x で同じ展開手順を繰り返す必要があります。CentOS 7 の違いを考慮して、展開スクリプトを調整する必要がある場合があります。

Guest Shell 2.x

Cisco NX-OS は、デフォルトで自動的に Guest Shell のインストールおよび有効化を行います。ただし、Guest Shell をサポートしない Cisco NX-OS イメージでデバイスがリロードされる場合、既存の Guest Shell が自動的に削除され、%VMAN-2-INVALID_PACKAGE が発行されます。



- (注) 4GB の RAM を搭載したシステムでは、デフォルトでは Guest Shell が有効になりません。guestshell enable コマンドを使用して、Guest Shell をインストールして有効にします。

install all コマンドは、現在の Cisco NX-OS イメージとターゲットの Cisco NX-OS イメージとの互換性を検証します。

互換性のないイメージをインストールした場合の出力例を次に示します。

```
switch#
Installer will perform compatibility check first. Please wait.
uri is: /
2014 Aug 29 20:08:51 switch %$ VDC-1 %$ %VMAN-2-ACTIVATION_STATE:
Successfully activated virtual service 'guestshell+'
Verifying image bootflash:/n9kpregs.bin for boot variable "nxos".
[#####] 100% -- SUCCESS
Verifying image type.
[#####] 100% -- SUCCESS
Preparing "" version info using image bootflash:/.
[#####] 100% -- SUCCESS
Preparing "bios" version info using image bootflash:/.
[#####] 100% -- SUCCESS
Preparing "" version info using image bootflash:/.
[#####] 100% -- SUCCESS
Preparing "" version info using image bootflash:/.
[#####] 100% -- SUCCESS
Preparing "nxos" version info using image bootflash:/.
[#####] 100% -- SUCCESS
Preparing "" version info using image bootflash:/.
[#####] 100% -- SUCCESS
Preparing "" version info using image bootflash:/.
[#####] 100% -- SUCCESS
"Running-config contains configuration that is incompatible with the new image (strict
incompatibility).
Please run 'show incompatibility-all nxos <image>' command to find out which feature
needs to be disabled.".
Performing module support checks.
[#####] 100% -- SUCCESS
Notifying services about system upgrade.
[# ] 0% -- FAIL.
Return code 0x42DD0006 ((null)).
"Running-config contains configuration that is incompatible with the new image (strict
incompatibility).
Please run 'show incompatibility-all nxos <image>' command to find out
which feature needs to be disabled."
```

```
Service "vman" in vdc 1: Guest shell not supported, do 'guestshell destroy' to remove
it and then retry ISSU
Pre-upgrade check failed. Return code 0x42DD0006 ((null)).
switch#
```



- (注) ベストプラクティスとして、Guest Shell をサポートしていない古い Cisco Nexus イメージをリロードする前に、**guestshell destroy** コマンドを使用して Guest Shell を削除します。

事前設定された SSHD サービス

Guest Shell は、起動時に OpenSSH サーバーを開始します。サーバーは、localhost IP アドレス インターフェイス 127.0.0.1 でランダムに生成されたポートでのみリスンします。これにより、**guestshell** キーワードが入力されたときに、NX-OS vegas-shell から Guest Shell へのパスワードなしの接続が提供されます。このサーバーが強制終了されるか、その構成 (/etc/ssh/sshd_config-cisco にある) が変更された場合、NX-OS CLI からの Guest Shell へのアクセスが機能しない可能性があります。

次の手順では、Guest Shell 内で root として OpenSSH サーバーをインスタンス化します。

1. SSH 接続を確立するネットワーク名前空間または VRF を決定します。
2. OpenSSH がリスンするポートを決定します。すでに使用されているポートを表示するには、NX-OS コマンドの **show socket connection** を使用します。



- (注) パスワードなしのアクセス用の Guest Shell sshd サービスは、17680 から 49150 までのランダム化されたポートを使用します。ポートの競合を避けるには、この範囲外のポートを選択してください。

次の手順では、OpenSSH サーバーを起動します。例では、IP アドレス 10.122.84.34:2222 で管理 netns の OpenSSH サーバーを起動します。

1. 次のファイルを作成します: /usr/lib/systemd/system/sshd-mgmt.service および /etc/ssh/sshd-mgmt_config。ファイルには次の構成が必要です。

```
-rw-r--r-- 1 root root 394 Apr 7 14:21 /usr/lib/systemd/system/sshd-mgmt.service
-rw----- 1 root root 4478 Apr 7 14:22 /etc/ssh/sshd-mgmt_config
```

2. Unit と Service の内容を /usr/lib/systemd/system/ssh.service ファイルから sshd-mgmt.service にコピーします。
3. sshd-mgmt.service ファイルを次のように編集します。

```
[Unit]
Description=OpenSSH server daemon
After=network.target sshd-keygen.service
Wants=sshd-keygen.service

[Service]
EnvironmentFile=/etc/sysconfig/ssh
ExecStartPre=/usr/sbin/sshd-keygen
```

```

ExecStart=/sbin/ip netns exec management /usr/sbin/sshd -f /etc/ssh/sshd-mgmt_config
-D $OPTIONS
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure
RestartSec=42s
[Install]
WantedBy=multi-user.target

```

4. /etc/ssh/sshd-config の内容を /etc/ssh/sshd-mgmt_config にコピーします。必要に応じて、ListenAddress IP とポートを変更します。

```

Port 2222
ListenAddress 10.122.84.34

```

5. 次のコマンドを使用して、systemctl デーモンを開始します。

```

sudo systemctl daemon-reload
sudo systemctl start sshd-mgmt.service
sudo systemctl status sshd-mgmt.service -l

```

6. (オプション) 構成を確認します。

```
ss -tnldp | grep 2222
```

7. ゲスト シェルへの SSH:

```
ssh -p 2222 guestshell@10.122.84.34
```

8. 複数の Guest Shell またはスイッチの再起動にまたがって構成を保存します。

```
sudo systemctl enable sshd-mgmt.service
```

9. パスワードなしの SSH/SCP およびリモート実行の場合、ssh-keygen -t dsa コマンドを使用して、SSH/SCP に使用するユーザー ID の公開鍵と秘密鍵を生成します。

その後、キーは /.ssh ディレクトリの id_rsa および id_rsa.pub ファイルに保存されます。

```

[root@node01 ~]# cd ~/.ssh
[root@node02 .ssh]# ls -l
total 8
-rw-----. 1 root root 1675 May 5 15:01 id_rsa
-rw-r--r--. 1 root root 406 May 5 15:01 id_rsa.pub

```

10. 公開キーを SSH で接続するマシンにコピーし、アクセス許可を修正します。

```

cat id_rsa.pub >> /root/.ssh/authorized_keys
chmod 700 /root/.ssh
chmod 600 /root/.ssh/*

```

11. パスワードなしでリモート スイッチに SSH または SCP:

```

ssh -p <port#> userid@hostname [<remote command>]
scp -P <port#> userid@hostname/filepath /destination

```

localtime

Guest Shell は、ホストシステムと /etc/localtime を共有します。



- (注) ホストと同じ **localtime** を共有したくない場合は、このシンボリック リンクを切断して、ゲストシェル固有の `/etc/localtime` を作成できます。

```
switch(config)# clock timezone PDT -7 0
switch(config)# clock set 10:00:00 27 Jan 2017
Fri Jan 27 10:00:00 PDT 2017
switch(config)# show clock
10:00:07.554 PDT Fri Jan 27 2017
switch(config)# run guestshell
guestshell:~$ date
Fri Jan 27 10:00:12 PDT 2017
```

Guestshell に関する注意事項と制限事項

すべてのリリースに共通の注意事項



- 重要** Guestshell のインストール内でカスタム作業を実行した場合は、Guestshell のアップグレードを実行する前に、ブートフラッシュ、オフボックスストレージ、または Guestshell ルート ファイルシステムの外部の他の場所に変更を保存します。

guestshell upgrade コマンドは、本質的に、guestshell destroy と guestshell enable を連続して実行します。

- Guest Shell は、4 GB のメモリを搭載した 3500 モデル（3524、3548、3524-X、3548-X）ではサポートされていません。これは、-XL など、より多くのメモリを備えたプラットフォームでサポートされます。
- Guestshell でサードパーティの DHCPD サーバーを実行している場合、SVI と一緒に使用すると、クライアントに到達するオファーに問題が発生する可能性があります。可能な回避策は、ブロードキャスト応答を使用することです。
- run guestshell CLI コマンドを使用して、スイッチの Guestshell にアクセスします。run guestshell コマンドは、ホスト シェルへのアクセスに使用される run bash コマンドに相当します。このコマンドを使用すると、Guestshell にアクセスして Bash プロンプトを取得したり、Guestshell のコンテキスト内でコマンドを実行したりできます。このコマンドは、パスワードなしの SSH を使用して、デフォルトのネットワーク名前空間にある localhost の使用可能なポートに接続します。
- NXOS の 2 つの異なる VRF 間で（静的または動的に）ルートが交換されている場合、対応する VRF/名前空間のルーティング テーブルに、ゲストシェルコンテナの「共有ルート」が入力されることはありません。
- sshd ユーティリティは、ローカルホストでリッスンして、ネットワークの外部からの接続試行を回避することにより、Guestshell への事前構成された SSH アクセスを保護できます。sshd には次の機能があります。

- これは、パスワードにフォールバックしないキーベースの認証用に構成されています。
- Guestshell の再起動後に Guestshell にアクセスするために使用されるキーを読み取ることができるのは root だけです。
- root だけがホスト上のキーを含むファイルを読み取ることができ、ホスト Bash アクセスを持つ非特権ユーザーがキーを使用して Guestshell に接続できないようにします。ネットワーク管理ユーザーは、Guestshell で sshd の別のインスタンスを開始して、Guestshell ネットワーク管理ユーザーは、Guestshell で sshd の別のインスタンスを開始して、ネットワーク管理ユーザーは、Guestshell で sshd の別のインスタンスを開始して、アクセスできるようにすることができますが、Guestshell にログインするユーザーにはネットワーク管理者権限も与えられます。



(注) Guestshell 2.2 (0.2) で導入されたキー ファイルは、ユーザー アカウントが作成されたユーザーに対して読み取り可能です。

さらに、Guestshell アカウントは自動的に削除されないため、不要になったときにネットワーク管理者が削除する必要があります。

2.2 (0.2) より前の Guestshell インストールでは、個々のユーザー アカウントが動的に作成されません。

- すぐに使用できる新しいスイッチに Cisco NX-OS ソフトウェア リリースをインストールすると、Guestshell が自動的に有効になります。その後のスイッチ ソフトウェアのアップグレードでは、Guestshell は自動的にアップグレードされません。
- Guestshell リリースでは、配布または配布バージョンが変更されると、メジャー番号が増分します。
- NX-OS の Guestshell は、前面パネルのポートに、ファーストクラスの Linux インターフェイスとしてアクセスできます。
- NX-OS の Guestshell は、NX-API へのローカル Unix ソケットを使用し、dohost を介してコマンドシェルにアクセスできます。
 1. 9.3(8) 以降の NX-OS の Guestshell において、NX-API ソケットへのアクセスは、root/管理者ユーザー権限でのみ許可されます。
 2. 9.3 (8) 以降の NX-OS の Guestshell において、NX-OS ファイルシステムへのアクセスは、root/管理者ユーザーだけが行います。
- Guestshell リリースでは、CVE が解決されるとマイナー番号が増分します。Guestshell は、CentOS が公開した場合にのみ CVE を更新します。
- **dnf update** を使用して、CentOS リポジトリからサードパーティのセキュリティ脆弱性修正を直接取得することをお勧めします。これにより、Cisco NX-OS ソフトウェアのアップデートを待つことなく、更新が利用可能になったときに入手できる柔軟性が得られます。

または、**guestshell update** コマンドを使用すると、既存の Guestshell rootfs が置き換えられます。カスタマイズとソフトウェア パッケージのインストールは、この新しい Guestshell rootfs のコンテキスト内で再度実行する必要があります。

- bash シェルからの Nexus クロックの設定はサポートされていません。

CentOS のサポート終了と Guestshell への影響

Guestshell は CentOS 環境に基づく LXC コンテナです。オープン ソース コミュニティの更新によると、CentOS 8 プロジェクトは 2021 年 12 月までにサポートが終了します。CentOS 7 プロジェクトは継続され、2024 年 6 月までにサポートが終了する予定です。CentOS 7 のこの長期サポートにより、最新の Cisco NX-OS ソフトウェア 10.2.x は Guestshell 2.11 (CentOS 7 ベース) にパッケージ化されています。これは、10.1.x リリースのデフォルト環境である Guestshell 3.0 (CentOS 8) を置き換えます。

Guestshell 2.11

Cisco NX-OS リリース 10.2(1) 以降、CentOS 7 がデフォルトの Guestshell 環境として再展開されました。理由の詳細については、「CentOS のサポート終了」セクションを参照してください。

Guestshell 2.11 には python2 および python3.6 のサポートが付属しています。Guestshell 2.11 と Guestshell 3.0 の間の機能は同じままです。



(注) Guestshell 2.11 の rootfs サイズは約 200 MB に増加しました。

Guestshell 3.0

Guestshell 3.0 は廃止されており、NX-OS 10.2.x からは利用できません。Guestshell 2.11 を使用することをお勧めします。ただし、10.2.x ソフトウェアは、Guestshell 3.0 コンテナおよび 10.1.x で動作している 3.0 Guestshell コンテナとの互換性を維持しています。ソフトウェアは、10.2.x へのアップグレード後も引き続き機能します。



(注) Guestshell 3.0 の rootfs サイズは、Guestshell 2.0 の 170 MB に対して 220 MB です。

Guestshell 4.x

Guestshell 2.x には Centos 7 が含まれています。Centos 7 のサポート終了は 2024 年初めです。したがって、Guestshell 4.x は RockyLinux 9 ベースとなり、Guestshell 2.x を置き換えることになります。Guestshell 4.x は、次の NX-OS リリースでダウンロード可能になり、デフォルトのオプションとして使用できるようになります。

NX-OS リリース	Guestshell のデフォルトパッケージバージョン	Guestshell のダウンロード可能なオプションが適用可能
9.3(14) 以降	4.1 以降	不要
10.2(6)	2.15	4.0
10.2(7)	2.15	4.1
10.2(8) 以降	4.1 以降	不要
10.3(4)	2.15	4.0
10.3(5) 以降	4.1 以降	不要
10.4(1) および 10.4(2)	2.15	4.0
10.4(3) 以降	4.1 以降	不要
10.5(1) 以降	4.1 以降	不要



(注) Guestshell 4.x の rootfs サイズは 400 MB です。Guestshell 2.x では 350 MB でした。Guestshell 4.x のダウンロード可能な OVA は、デフォルトとして Guestshell 2.x を実行しているすべてのリリースに後方互換性を持ちます。

Guestshell 1.0 から Guestshell 2.x へのアップグレード

Guestshell 2.x は、CentOS 7 ルート ファイル システムに基づいています。コンテンツを Guestshell 1.0 にプルダウンした .conf ファイルまたはユーティリティのオフボックス リポジトリがある場合は、Guestshell 2.x で同じ展開手順を繰り返す必要があります。CentOS 7 の違いを考慮して、展開スクリプトを調整する必要がある場合があります。

Jacksonville リリース Guestshell 3.0 からの NX-OS のダウングレード

Cisco NX-OS リリース 10.1(1) 以降、Guestshell 3.0 サポートのインフラストラクチャバージョンは 1.11 に引き上げられています (show virtual-service コマンドで確認してください)。したがって、Guestshell 3.0 OVA は以前のリリースでは使用できません。**Install all** コマンドを使用すると、バージョンの不一致が検証され、エラーがスローされます。Guestshell 3.0 を以前のリリースにダウングレードする前に、Guestshell 3.0 を破棄して、Guestshell 3.0 が以前のリリースで起動しないようにすることをお勧めします。

Guestshell 2.x から Guestshell 4.x ダウンロード可能 OVA へのアップグレード

Guestshell 4.x は、Cisco の公式ソフトウェアダウンロードページからダウンロードでき、command guestshell upgrade コマンドを使用してインストールできます。

次の表に、ゲスト シェルのリリースを示します。

表 2: ゲストシェル リリース

ゲスト シェル リリース	NX-OS がサポートするリリース	サポートされる Python のバージョン
2.x	10.4.1	Python 2.7 および Python 3.6
3.0	10.1.x	python 3.6
4.x	10.4.1	Python 3.9

Guestshell 4.x にアップグレードするには、次のコマンドを使用します：

- ゲストシェルがインストールされていない場合には、**guestshell enable package <downloaded ova>** コマンドを実行します。
- ゲストシェルがインストールされ、実行されている場合には、**guestshell upgrade package <downloaded ova>** コマンドを実行します。



(注) 4 GB の RAM を搭載したシステムでは、デフォルトでは Guestshell が有効になりません。 **guestshell enable** コマンドを使用して、Guestshell をインストールして有効にします。

install all コマンドは、現在の Cisco NX-OS イメージとターゲットの Cisco NX-OS イメージとの互換性を検証します。

互換性のないイメージをインストールした場合の出力例を次に示します。

```
switch#
Installer will perform compatibility check first. Please wait.
uri is: /
2014 Aug 29 20:08:51 switch %$ VDC-1 %$ %VMAN-2-ACTIVATION_STATE:
Successfully activated virtual service 'guestshell+'
Verifying image bootflash:/n9kpregs.bin for boot variable "nxos".
[#####] 100% -- SUCCESS
Verifying image type.
[#####] 100% -- SUCCESS
Preparing "" version info using image bootflash:/.
[#####] 100% -- SUCCESS
Preparing "bios" version info using image bootflash:/.
[#####] 100% -- SUCCESS
Preparing "" version info using image bootflash:/.
[#####] 100% -- SUCCESS
Preparing "" version info using image bootflash:/.
[#####] 100% -- SUCCESS
Preparing "nxos" version info using image bootflash:/.
[#####] 100% -- SUCCESS
Preparing "" version info using image bootflash:/.
[#####] 100% -- SUCCESS
Preparing "" version info using image bootflash:/.
[#####] 100% -- SUCCESS
"Running-config contains configuration that is incompatible with the new image (strict
incompatibility).
Please run 'show incompatibility-all nxos <image>' command to find out which feature
needs to be disabled.".
Performing module support checks.
```

```
[#####] 100% -- SUCCESS
Notifying services about system upgrade.
[# ] 0% -- FAIL.
Return code 0x42DD0006 ((null)).
"Running-config contains configuration that is incompatible with the new image (strict
incompatibility).
Please run 'show incompatibility-all nxos <image>' command to find out
which feature needs to be disabled."
Service "vman" in vdc 1: Guestshell not supported, do 'guestshell destroy' to remove
it and then retry ISSU
Pre-upgrade check failed. Return code 0x42DD0006 ((null)).
switch#
```



- (注) ベストプラクティスとして、Guestshellをサポートしていない古いCisco NX-OS イメージをリロードする前に、**guestshell destroy** コマンドを使用して Guestshell を削除します。

事前設定された SSHD サービス

Guestshell は、起動時に OpenSSH サーバーを開始します。サーバーは、localhost IP アドレス インターフェイス 127.0.0.1 でランダムに生成されたポートでのみリスンします。これにより、guestshell キーワードが入力されたときに、NX-OS 仮想シェルから Guestshell へのパスワードなしの接続が提供されます。このサーバーが強制終了されるか、その構成(/etc/ssh/sshd_config-cisco にある) が変更された場合、NX-OS CLI からの Guestshell へのアクセスが機能しない可能性があります。

次の手順では、Guestshell 内で root として OpenSSH サーバーをインスタンス化します。

1. SSH 接続を確立するネットワーク名前空間または VRF を決定します。
2. OpenSSH がリッスンするポートを決定します。すでに使用されているポートを表示するには、NX-OS コマンドの **show socket connection** を使用します。



- (注) パスワードなしのアクセス用の Guestshell sshd サービスは、17680 から 49150 までのランダム化されたポートを使用します。ポートの競合を避けるには、この範囲外のポートを選択してください。

次の手順では、OpenSSH サーバーを起動します。例では、IP アドレス 10.122.84.34:2222 で管理 netns の OpenSSH サーバーを起動します。

1. 次のファイルを作成します: /usr/lib/systemd/system/sshd-mgmt.service および /etc/ssh/sshd-mgmt_config。ファイルには次の構成が必要です。

```
-rw-r--r-- 1 root root 394 Apr 7 14:21 /usr/lib/systemd/system/sshd-mgmt.service
-rw----- 1 root root 4478 Apr 7 14:22 /etc/ssh/sshd-mgmt_config
```

2. Unit と Service の内容を /usr/lib/systemd/system/ssh.service ファイルから sshd-mgmt.service にコピーします。
3. sshd-mgmt.service ファイルを次のように編集します。

```
[Unit]
Description=OpenSSH server daemon
```

```

After=network.target sshd-keygen.service
Wants=sshd-keygen.service

[Service]
EnvironmentFile=/etc/sysconfig/sshd
ExecStartPre=/usr/sbin/sshd-keygen
ExecStart=/sbin/ip netns exec management /usr/sbin/sshd -f /etc/ssh/sshd-mgmt_config
-D $OPTIONS
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure
RestartSec=42s
[Install]
WantedBy=multi-user.target

```

4. /etc/ssh/sshd-config の内容を /etc/ssh/sshd-mgmt_config にコピーします。必要に応じて、ListenAddress IP とポートを変更します。

```

Port 2222
ListenAddress 10.122.84.34

```

5. 次のコマンドを使用して、systemctl デーモンを開始します。

```

sudo systemctl daemon-reload
sudo systemctl start sshd-mgmt.service
sudo systemctl status sshd-mgmt.service -l

```

6. (オプション) 構成を確認します。

```
ss -tnldp | grep 2222
```

7. Guestshell への SSH:

```
ssh -p 2222 guestshell@10.122.84.34
```

8. 複数の Guestshell またはスイッチの再起動にわたって構成を保存します。

```
sudo systemctl enable sshd-mgmt.service
```

9. パスワードなしの SSH/SCP およびリモート実行の場合、ssh-keygen -t dsa コマンドを使用して、SSH/SCP に使用するユーザー ID の公開鍵と秘密鍵を生成します。

その後、キーは /.ssh ディレクトリの id_rsa および id_rsa.pub ファイルに保存されます。

```

[root@node01 ~]# cd ~/.ssh
[root@node02 .ssh]# ls -l
total 8
-rw-----. 1 root root 1675 May 5 15:01 id_rsa
-rw-r--r--. 1 root root 406 May 5 15:01 id_rsa.pub

```

10. 公開キーを SSH で接続するマシンにコピーし、アクセス許可を修正します。

```

cat id_rsa.pub >> /root/.ssh/authorized_keys
chmod 700 /root/.ssh
chmod 600 /root/.ssh/*

```

11. パスワードなしでリモート スイッチに SSH または SCP:

```

ssh -p <port#> userid@hostname [<remote command>]
scp -P <port#> userid@hostname/filepath /destination

```

Localtime

Guestshell は、ホスト システムと `/etc/localtime` を共有します。



- (注) ホストと同じ `localtime` を共有したくない場合は、このシンボリック リンクを切断して、Guestshell 固有の `/etc/localtime` を作成できます。

```
switch(config)# clock timezone PDT -7 0
switch(config)# clock set 10:00:00 27 Jan 2017
Fri Jan 27 10:00:00 PDT 2017
switch(config)# show clock
10:00:07.554 PDT Fri Jan 27 2017
switch(config)# run guestshell
guestshell:~$ date
Fri Jan 27 10:00:12 PDT 2017
```

Guestshell に関する注意事項と制限事項

すべてのリリースに共通の注意事項



- 重要** Guestshell のインストール内でカスタム作業を実行した場合は、Guestshell のアップグレードを実行する前に、ブートフラッシュ、オフボックス ストレージ、または Guestshell ルート ファイル システムの外部の他の場所に変更を保存します。

`guestshell upgrade` コマンドは、本質的に、`guestshell destroy` と `guestshell enable` を連続して実行します。

- Guest Shell は、4 GB のメモリを搭載した 3500 モデル（3524、3548、3524-X、3548-X）ではサポートされていません。これは、-XL など、より多くのメモリを備えたプラットフォームでサポートされます。
- Guestshell でサードパーティの DHCPD サーバーを実行している場合、SVI と一緒に使用すると、クライアントに到達するオファーに問題が発生する可能性があります。可能な回避策は、ブロードキャスト応答を使用することです。
- `run guestshell CLI` コマンドを使用して、スイッチの Guestshell にアクセスします。`run guestshell` コマンドは、ホストシェルへのアクセスに使用される `run bash` コマンドに相当します。このコマンドを使用すると、Guestshell にアクセスして Bash プロンプトを取得したり、Guestshell のコンテキスト内でコマンドを実行したりできます。このコマンドは、パスワードなしの SSH を使用して、デフォルトのネットワーク名前空間にある `localhost` の使用可能なポートに接続します。
- NXOS の 2 つの異なる VRF 間で（静的または動的に）ルートが交換されている場合、対応する VRF/名前空間のルーティング テーブルに、ゲストシェルコンテナの「共有ルート」が入力されることはありません。

- `sshd` ユーティリティは、ローカルホストでリッスンして、ネットワークの外部からの接続試行を回避することにより、Guestshell への事前構成された SSH アクセスを保護できます。`sshd` には次の機能があります。
 - これは、パスワードにフォールバックしないキーベースの認証用に構成されています。
 - Guestshell の再起動後に Guestshell にアクセスするために使用されるキーを読み取ることができるのは `root` だけです。
 - `root` だけがホスト上のキーを含むファイルを読み取ることができ、ホスト Bash アクセスを持つ非特権ユーザーがキーを使用して Guestshell に接続できないようにします。ネットワーク管理ユーザーは、Guestshell で `sshd` の別のインスタンスを開始して、Guestshell ネットワーク管理ユーザーは、Guestshell で `sshd` の別のインスタンスを開始して、ネットワーク管理ユーザーは、Guestshell で `sshd` の別のインスタンスを開始して、アクセスできるようにすることができますが、Guestshell にログインするユーザーにはネットワーク管理者権限も与えられます。



(注) Guestshell 2.2 (0.2) で導入されたキー ファイルは、ユーザー アカウントが作成されたユーザーに対して読み取り可能です。

さらに、Guestshell アカウントは自動的に削除されないため、不要になったときにネットワーク管理者が削除する必要があります。

2.2 (0.2) より前の Guestshell インストールでは、個々のユーザー アカウントが動的に作成されません。

- すぐに使用できる新しいスイッチに Cisco NX-OS ソフトウェア リリースをインストールすると、Guestshell が自動的に有効になります。その後のスイッチ ソフトウェアのアップグレードでは、Guestshell は自動的にアップグレードされません。
- Guestshell リリースでは、配布または配布バージョンが変更されると、メジャー番号が増分します。
- NX-OS の Guestshell は、前面パネルのポートに、ファーストクラスの Linux インターフェイスとしてアクセスできます。
- NX-OS の Guestshell は、NX-API へのローカル Unix ソケットを使用し、`dohost` を介してコマンドシェルにアクセスできます。
 1. 9.3(8) 以降の NX-OS の Guestshell において、NX-API ソケットへのアクセスは、`root/管理者` ユーザー権限でのみ許可されます。
 2. 9.3 (8) 以降の NX-OS の Guestshell において、NX-OS ファイルシステムへのアクセスは、`root/管理者` ユーザーだけが行います。
- Guestshell リリースでは、CVE が解決されるとマイナー番号が増分します。Guestshell は、CentOS が公開した場合にのみ CVE を更新します。

- **dnf update** を使用して、CentOS リポジトリからサードパーティのセキュリティ脆弱性修正を直接取得することをお勧めします。これにより、Cisco NX-OS ソフトウェアのアップデートを待つことなく、更新が利用可能になったときに入手できる柔軟性が得られます。

または、**guestshell update** コマンドを使用すると、既存の Guestshell rootfs が置き換えられます。カスタマイズとソフトウェア パッケージのインストールは、この新しい Guestshell rootfs のコンテキスト内で再度実行する必要があります。

- bash シェルからの Nexus クロックの設定はサポートされていません。

CentOS のサポート終了と Guestshell への影響

Guestshell は **CentOS 環境に基づく LXC コンテナ**です。オープンソースコミュニティの更新によると、CentOS 8 プロジェクトは 2021 年 12 月までにサポートが終了します。CentOS 7 プロジェクトは継続され、2024 年 6 月までにサポートが終了する予定です。CentOS 7 のこの長期サポートにより、最新の Cisco NX-OS ソフトウェア 10.2.x は Guestshell 2.11 (CentOS 7 ベース) にパッケージ化されています。これは、10.1.x リリースのデフォルト環境である Guestshell 3.0 (CentOS 8) を置き換えます。

Guestshell 2.11

Cisco NX-OS リリース 10.2(1) 以降、CentOS 7 がデフォルトの Guestshell 環境として再展開されました。理由の詳細については、「CentOS のサポート終了」セクションを参照してください。

Guestshell 2.11 には python2 および python3.6 のサポートが付属しています。Guestshell 2.11 と Guestshell 3.0 の間の機能は同じままです。



(注) Guestshell 2.11 の rootfs サイズは約 200 MB に増加しました。

Guestshell 3.0

Guestshell 3.0 は廃止されており、NX-OS 10.2.x からは利用できません。Guestshell 2.11 を使用することをお勧めします。ただし、10.2.x ソフトウェアは、Guestshell 3.0 コンテナおよび 10.1.x で動作している 3.0 Guestshell コンテナとの互換性を維持しています。ソフトウェアは、10.2.x へのアップグレード後も引き続き機能します。



(注) Guestshell 3.0 の rootfs サイズは、Guestshell 2.0 の 170 MB に対して 220 MB です。

Guestshell 4.x

Guestshell 2.x には CentOS 7 が含まれています。Centos 7 のサポート終了は 2024 年初めです。したがって、Guestshell 4.x は RockyLinux 9 ベースとなり、Guestshell 2.x を置き換えることになります。Guestshell 4.x は、次の NX-OS リリースでダウンロード可能になり、デフォルトのオプションとして使用できるようになります。

NX-OS リリース	Guestshell のデフォルトパッケージバージョン	Guestshell のダウンロード可能なオプションが適用可能
9.3(14) 以降	4.1 以降	不要
10.2(6)	2.15	4.0
10.2(7)	2.15	4.1
10.2(8) 以降	4.1 以降	不要
10.3(4)	2.15	4.0
10.3(5) 以降	4.1 以降	不要
10.4(1) および 10.4(2)	2.15	4.0
10.4(3) 以降	4.1 以降	不要
10.5(1) 以降	4.1 以降	不要



(注) Guestshell 4.x の rootfs サイズは 400 MB です。Guestshell 2.x では 350 MB でした。Guestshell 4.x のダウンロード可能な OVA は、デフォルトとして Guestshell 2.x を実行しているすべてのリリースに後方互換性を持ちます。

Guestshell 1.0 から Guestshell 2.x へのアップグレード

Guestshell 2.x は、CentOS 7 ルート ファイル システムに基づいています。コンテンツを Guestshell 1.0 にプルダウンした .conf ファイルまたはユーティリティのオフボックス リポジトリがある場合は、Guestshell 2.x で同じ展開手順を繰り返す必要があります。CentOS 7 の違いを考慮して、展開スクリプトを調整する必要がある場合があります。

Jacksonville リリース Guestshell 3.0 からの NX-OS のダウングレード

Cisco NX-OS リリース 10.1(1) 以降、Guestshell 3.0 サポートのインフラストラクチャバージョンは 1.11 に引き上げられています (show virtual-service コマンドで確認してください)。したがって、Guestshell 3.0 OVA は以前のリリースでは使用できません。Install all コマンドを使用すると、バージョンの不一致が検証され、エラーがスローされます。Guestshell 3.0 を以前のリリースにダウングレードする前に、Guestshell 3.0 を破棄して、Guestshell 3.0 が以前のリリースで起動しないようにすることをお勧めします。

Guestshell 2.x から Guestshell 4.x ダウンロード可能 OVA へのアップグレード

Guestshell 4.x は、Cisco の公式ソフトウェアダウンロードページからダウンロードでき、command guestshell upgrade コマンドを使用してインストールできます。

次の表に、ゲスト シェルのリリースを示します。

表 3: ゲスト シェル リリース

ゲスト シェル リリース	NX-OS がサポートするリリース	サポートされる Python のバージョン
2.x	10.4.1	Python 2.7 および Python 3.6
3.0	10.1.x	python 3.6
4.x	10.4.1	Python 3.9

Guestshell 4.x にアップグレードするには、次のコマンドを使用します:

- ゲストシェルがインストールされていない場合には、**guestshell enable package <downloaded ova>** コマンドを実行します。
- ゲストシェルがインストールされ、実行されている場合には、**guestshell upgrade package <downloaded ova>** コマンドを実行します。



(注) 4 GB の RAM を搭載したシステムでは、デフォルトでは Guestshell が有効になりません。 **guestshell enable** コマンドを使用して、Guestshell をインストールして有効にします。

install all コマンドは、現在の Cisco NX-OS イメージとターゲットの Cisco NX-OS イメージとの互換性を検証します。

互換性のないイメージをインストールした場合の出力例を次に示します。

```
switch#
Installer will perform compatibility check first. Please wait.
uri is: /
2014 Aug 29 20:08:51 switch %$ VDC-1 %$ %VMAN-2-ACTIVATION_STATE:
Successfully activated virtual service 'guestshell+'
Verifying image bootflash:/n9kpregs.bin for boot variable "nxos".
[#####] 100% -- SUCCESS
Verifying image type.
[#####] 100% -- SUCCESS
Preparing "" version info using image bootflash:/.
[#####] 100% -- SUCCESS
Preparing "bios" version info using image bootflash:/.
[#####] 100% -- SUCCESS
Preparing "" version info using image bootflash:/.
[#####] 100% -- SUCCESS
Preparing "" version info using image bootflash:/.
[#####] 100% -- SUCCESS
Preparing "nxos" version info using image bootflash:/.
[#####] 100% -- SUCCESS
Preparing "" version info using image bootflash:/.
[#####] 100% -- SUCCESS
Preparing "" version info using image bootflash:/.
[#####] 100% -- SUCCESS
"Running-config contains configuration that is incompatible with the new image (strict
incompatibility).
Please run 'show incompatibility-all nxos <image>' command to find out which feature
needs to be disabled.".
Performing module support checks.
```



```
[#####] 100% -- SUCCESS
Notifying services about system upgrade.
[# ] 0% -- FAIL.
Return code 0x42DD0006 ((null)).
"Running-config contains configuration that is incompatible with the new image (strict
incompatibility).
Please run 'show incompatibility-all nxos <image>' command to find out
which feature needs to be disabled."
Service "vman" in vdc 1: Guestshell not supported, do 'guestshell destroy' to remove
it and then retry ISSU
Pre-upgrade check failed. Return code 0x42DD0006 ((null)).
switch#
```



- (注) ベストプラクティスとして、Guestshellをサポートしていない古いCisco NX-OS イメージをリロードする前に、**guestshell destroy** コマンドを使用して Guestshell を削除します。

事前設定された SSHD サービス

Guestshell は、起動時に OpenSSH サーバーを開始します。サーバーは、localhost IP アドレス インターフェイス 127.0.0.1 でランダムに生成されたポートでのみリスンします。これにより、guestshell キーワードが入力されたときに、NX-OS 仮想シェルから Guestshell へのパスワードなしの接続が提供されます。このサーバーが強制終了されるか、その構成(/etc/ssh/sshd_config-ciscoにある)が変更された場合、NX-OS CLI からの Guestshell へのアクセスが機能しない可能性があります。

次の手順では、Guestshell 内で root として OpenSSH サーバーをインスタンス化します。

1. SSH 接続を確立するネットワーク名前空間または VRF を決定します。
2. OpenSSH がリスンするポートを決定します。すでに使用されているポートを表示するには、NX-OS コマンドの **show socket connection** を使用します。



- (注) パスワードなしのアクセス用の Guestshell sshd サービスは、17680 から 49150 までのランダム化されたポートを使用します。ポートの競合を避けるには、この範囲外のポートを選択してください。

次の手順では、OpenSSH サーバーを起動します。例では、IP アドレス 10.122.84.34:2222 で管理 netns の OpenSSH サーバーを起動します。

1. 次のファイルを作成します: /usr/lib/systemd/system/sshd-mgmt.service および /etc/ssh/sshd-mgmt_config。ファイルには次の構成が必要です。

```
-rw-r--r-- 1 root root 394 Apr 7 14:21 /usr/lib/systemd/system/sshd-mgmt.service
-rw----- 1 root root 4478 Apr 7 14:22 /etc/ssh/sshd-mgmt_config
```

2. Unit と Service の内容を /usr/lib/systemd/system/ssh.service ファイルから sshd-mgmt.service にコピーします。
3. sshd-mgmt.service ファイルを次のように編集します。

```
[Unit]
Description=OpenSSH server daemon
```

```

After=network.target sshd-keygen.service
Wants=sshd-keygen.service

[Service]
EnvironmentFile=/etc/sysconfig/sshd
ExecStartPre=/usr/sbin/sshd-keygen
ExecStart=/sbin/ip netns exec management /usr/sbin/sshd -f /etc/ssh/sshd-mgmt_config
-D $OPTIONS
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure
RestartSec=42s
[Install]
WantedBy=multi-user.target

```

4. /etc/ssh/sshd-config の内容を /etc/ssh/sshd-mgmt_config にコピーします。必要に応じて、ListenAddress IP とポートを変更します。

```

Port 2222
ListenAddress 10.122.84.34

```

5. 次のコマンドを使用して、systemctl デーモンを開始します。

```

sudo systemctl daemon-reload
sudo systemctl start sshd-mgmt.service
sudo systemctl status sshd-mgmt.service -l

```

6. (オプション) 構成を確認します。

```
ss -tnldp | grep 2222
```

7. Guestshell への SSH:

```
ssh -p 2222 guestshell@10.122.84.34
```

8. 複数の Guestshell またはスイッチの再起動にわたって構成を保存します。

```
sudo systemctl enable sshd-mgmt.service
```

9. パスワードなしの SSH/SCP およびリモート実行の場合、ssh-keygen -t dsa コマンドを使用して、SSH/SCP に使用するユーザー ID の公開鍵と秘密鍵を生成します。

その後、キーは /.ssh ディレクトリの id_rsa および id_rsa.pub ファイルに保存されます。

```

[root@node01 ~]# cd ~/.ssh
[root@node02 .ssh]# ls -l
total 8
-rw-----. 1 root root 1675 May 5 15:01 id_rsa
-rw-r--r--. 1 root root 406 May 5 15:01 id_rsa.pub

```

10. 公開キーを SSH で接続するマシンにコピーし、アクセス許可を修正します。

```

cat id_rsa.pub >> /root/.ssh/authorized_keys
chmod 700 /root/.ssh
chmod 600 /root/.ssh/*

```

11. パスワードなしでリモート スイッチに SSH または SCP:

```

ssh -p <port#> userid@hostname [<remote command>]
scp -P <port#> userid@hostname/filepath /destination

```

Localtime

Guestshell は、ホスト システムと /etc/localtime を共有します。



- (注) ホストと同じ **localtime** を共有したくない場合は、このシンボリック リンクを切断して、Guestshell 固有の /etc/localtime を作成できます。

```
switch(config)# clock timezone PDT -7 0
switch(config)# clock set 10:00:00 27 Jan 2017
Fri Jan 27 10:00:00 PDT 2017
switch(config)# show clock
10:00:07.554 PDT Fri Jan 27 2017
switch(config)# run guestshell
guestshell:~$ date
Fri Jan 27 10:00:12 PDT 2017
```

Guest Shell へのアクセス

Cisco NX-OS では、デフォルトで **network-admin** ユーザのみがゲストシェルにアクセスできます。これはシステムで自動的に有効になっており、**run guestshell** コマンドを使用してアクセスできます。**run bash** コマンドと一致して、これらのコマンドは、NX-OS CLI コマンドの **run guestshell** コマンド 形式を使用して Guest Shell 内で発行できます。



- (注) Guest Shell は、4 GB を超える RAM を搭載したシステムで自動的に有効になります。

```
switch# run guestshell ls -al /bootflash/*.ova
-rw-rw-rw- 1 2002 503 83814400 Aug 21 18:04 /bootflash/pup.ova
-rw-rw-rw- 1 2002 503 40724480 Apr 15 2012 /bootflash/red.ova
```



- (注) Guest Shell で実行している場合、ネットワーク管理者レベルの権限があります。



- (注) 2.2(0.2) 以降の Guest Shell は、スイッチにログインしているユーザーと同じユーザー アカウントを動的に作成します。ただし、他のすべての情報は、スイッチと Guest Shell のユーザー アカウント間で共有されません。

さらに、Guest Shell アカウントは自動的に削除されないため、不要になったときにネットワーク管理者が削除する必要があります。

ゲスト シェルに使用されるリソース

デフォルトでは、ゲストシェルのリソースは、通常のスイッチ操作に使用できるリソースに小さな影響を与えます。ネットワーク管理者がゲスト シェルに追加のリソースを必要とする場合、**guestshell resize {cpu | memory | rootfs}** コマンドは、これらの制限を変更します

リソース	デフォルト	最小/最大
CPU	1 %	1/620%
メモリ	400 MB	256/3840 MB
ストレージ	200 MB	200/2000 MB

CPU 制限は、システム内の他のコンピューティング負荷との競合がある場合に、ゲストシェル内で実行されているタスクに与えられるシステムコンピューティングキャパシティのパーセンテージです。CPU リソースの競合がない場合、ゲスト シェル内のタスクは制限されません。



(注) リソース割り当てを変更した後は、ゲスト シェルの再起動が必要です。そのために、**guestshell reboot** コマンドを使用できます。

ゲストシェルの機能

Guestshell には、デフォルトで利用可能な多くのユーティリティと機能があります。

ゲストシェルは CentOS 7 Linux 環境であり、この流通向けにビルドされたソフトウェア パッケージを、yum インストールすることができます。Guestshell には、**net-tools**、**iproute**、**tcpdump** と OpenSSH などのネットワークングデバイスで自然に期待される多くの一般的なツールが事前に入力されています。Guestshell 2.x の場合、追加の **python** パッケージをインストールするための PIP と同様に、**python 2.7.5** がデフォルトで含まれています。Guestshell 2.11 では、デフォルトで **python 3.6** も含まれています。

デフォルトでは、ゲストシェルは 64 ビットの実行スペースです。32 ビットのサポートが必要な場合は、**glibc.i686** パッケージを yum でインストールできます。

Guestshell は、スイッチの管理ポートとデータポートを表すために使用される Linux ネットワーク インターフェイスにアクセスできます。**ifconfig** と **ethtool** などの典型的な Linux のメソッドとユーティリティは、カウンターの収集に使用できます。インターフェイスが NX-OS CLI で VRF に配置されると、Linux ネットワーク インターフェイスはその VRF のネットワーク名前空間に配置されます。名前空間は **/var/run/netns** で見ることができ、**ip netns** ユーティリティを使用してさまざまな名前空間のコンテキストで実行できます。いくつかのユーティリティ、**chvrf** と **vrfinfo** は、別の名前空間で実行し、プロセスが実行されている名前空間 **/vrf** に関する情報を取得するために提供されています。

systemd は、ゲストシェルを含む CentOS 8 環境でサービスを管理するために使用されます。

Guest Shell の NX-OS CLI

ゲストシェルは、ユーザーがゲストシェル環境からホスト ネットワーク要素に NX-OS コマンドを発行できるようにするアプリケーションを提供します。**dohost** アプリケーションは、有効な NX-OS 構成または **exec** コマンドを受け入れ、それらをホスト ネットワーク要素に発行します。

dohost コマンドを呼び出すときは、各 NX-OS コマンドを一重引用符または二重引用符で囲むことができます：

```
dohost "<NXOS CLI>"
```

NX-OS CLI は連鎖させることができます：

```
[guestshell@guestshell ~]$ dohost "sh lldp time | in Hold" "show cdp global"
Holdtime in seconds: 120
Global CDP information:
CDP enabled globally
Refresh time is 21 seconds
Hold time is 180 seconds
CDPv2 advertisements is enabled
DeviceID TLV in System-Name(Default) Format
[guestshell@guestshell ~]$
```

NX-OS CLI は、各コマンドの間にセミコロンを追加することにより、NX-OS スタイルのコマンドチェーン技術を使用して一緒にチェーンすることもできます。（セミコロンの両側にスペースが必要です。）：

```
[guestshell@guestshell ~]$ dohost "conf t ; cdp timer 13 ; show run | inc cdp"
Enter configuration commands, one per line. End with CNTL/Z.
cdp timer 13
[guestshell@guestshell ~]$
```



（注） Guest Shell 2.2 (0.2) 以降を使用するリリース 7.0(3)I5(2) の場合、**dohost** コマンドを介してホストで発行されたコマンドは、ゲスト シェル ユーザの有効なロールに基づく特権で実行されます。

以前のバージョンのゲスト シェルは、ネットワーク管理者レベルの権限でコマンドを実行します。

NX-API への UDS 接続の数が最大許容数に達すると、**dohost** コマンドは機能不全になります。

Guest Shell でのネットワーク アクセス

NX-OS スイッチポートは、Guest Shell では Linux ネットワーク インターフェイスとして表されます。ifconfig または ethtool を使用して、/proc/net/dev の表示統計などの一般的な Linux メソッドはすべてサポートされています。

Guest Shell には、多くの一般的なネットワーク ユーティリティがデフォルトで含まれており、**chvrf vrf command** コマンドを使用してさまざまな VRF で使用できます。

```
[guestshell@guestshell bootflash]$ ifconfig Eth1-47
Eth1-47: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 13.0.0.47 netmask 255.255.255.0 broadcast 13.0.0.255
ether 54:7f:ee:8e:27:bc txqueuelen 100 (Ethernet)
RX packets 311442 bytes 21703008 (20.6 MiB)
RX errors 0 dropped 185 overruns 0 frame 0
TX packets 12967 bytes 3023575 (2.8 MiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Guest Shell 内では、ネットワーク状態をモニタリングできますが、変更することはできません。ネットワーク状態を変更するには、ホストの **bash** シェルで NX-OS CLI または適切な Linux ユーティリティを使用します。

この **tcpdump** コマンドは Guest Shell にパッケージ化されており、管理ポートまたはスイッチ ポートでパントされたトラフィックのパケット トレースを可能にします。

この **sudo ip netns exec management ping** ユーティリティは、指定されたネットワーク名前空間のコンテキストでコマンドを実行するための一般的な方法です。これは Guest Shell 内で実行できます。

```
[guestshell@guestshell bootflash]$ sudo ip netns exec management ping 10.28.38.48
PING 10.28.38.48 (10.28.38.48) 56(84) bytes of data.
64 bytes from 10.28.38.48: icmp_seq=1 ttl=48 time=76.5 ms
```

chvrf ユーティリティは便宜のために提供されています。

```
guestshell@guestshell bootflash]$ chvrf management ping 10.28.38.48
PING 10.28.38.48 (10.28.38.48) 56(84) bytes of data.
64 bytes from 10.28.38.48: icmp_seq=1 ttl=48 time=76.5 ms
```



(注) コマンドなしで実行される **chvrf** コマンドは、現在の VRF / ネットワーク名前空間で実行されます。

たとえば、管理 VRF 経由で IP アドレス 10.0.0.1 を ping するには、コマンドは「**chvrf management ping 10.0.0.1**」です。 **scp** または **ssh** などの他のユーティリティも同様です。

例:

```
switch# guestshell
[guestshell@guestshell ~]$ cd /bootflash
[guestshell@guestshell bootflash]$ chvrf management scp foo@10.28.38.48:/foo/index.html
index.html
foo@10.28.38.48's password:
index.html 100% 1804 1.8KB/s 00:00
[guestshell@guestshell bootflash]$ ls -al index.html
-rw-r--r-- 1 guestshe users 1804 Sep 13 20:28 index.html
[guestshell@guestshell bootflash]$
[guestshell@guestshell bootflash]$ chvrf management curl cisco.com
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>301 Moved Permanently</title>
```

```
</head><body>
<h1>Moved Permanently</h1>
<p>The document has moved <a href="http://www.cisco.com/">here</a>.</p>
</body></html>
[guestshell@guestshell bootflash]$
```

システム上の VRF のリストを取得するには、NX-OS からネイティブに、**show vrf** または **dohost** コマンドを介してコマンドを使用します。

例:

```
[guestshell@guestshell bootflash]$ dohost 'sh vrf'
VRF-Name   VRF-ID   State    Reason
default    1        Up       --
management 2        Up       --
red         6        Up       --
```

Guest Shell 内では、VRFに関連付けられたネットワーク名前空間が実際に使用されます。どのネットワーク名前空間が存在するかを確認する方が便利な場合があります。

```
[guestshell@guestshell bootflash]$ ls /var/run/netns
default management red
[guestshell@guestshell bootflash]$
```

Guest Shell 内からドメイン名を解決するには、リゾルバーを構成する必要があります。Guest Shell で `/etc/resolv.conf` ファイルを編集して、ネットワークに適した DNS ネームサーバとドメインを含めます。

例:

```
nameserver 10.1.1.1
domain cisco.com
```

ネームサーバーとドメインの情報は、NX-OS 構成で構成されたものと一致する必要があります。

例:

```
switch(config)# ip domain-name cisco.com
switch(config)# ip name-server 10.1.1.1
switch(config)# vrf context management
switch(config-vrf)# ip domain-name cisco.com
switch(config-vrf)# ip name-server 10.1.1.1
```

スイッチが HTTP プロキシ サーバーを使用するネットワーク内にある場合、**http_proxy** および **https_proxy** 環境変数も Guest Shell 内で設定する必要があります。

例:

```
export http_proxy=http://proxy.esl.cisco.com:8080
export https_proxy=http://proxy.esl.cisco.com:8080
```

これらの環境変数は、`.bashrc` ファイルまたは適切なスクリプトで設定して、永続的であることを確認する必要があります。

ゲストシェルでのブートフラッシュへのアクセス

ネットワーク管理者は、NX-OS CLI コマンドの使用に加えて、Linux コマンドとユーティリティを使用してファイルを管理できます。ゲストシェル環境の `/bootflash` にシステムブートフラッシュをマウントすることにより、`network-admin` は Linux コマンドを使用してこれらのファイルを操作できます。

例:

```
find . -name "foo.txt"
rm "/bootflash/junk/foo.txt"
```



(注) ゲストシェル内のユーザーの名前はホストの場合と同じですが、ゲストシェルは別のユーザー名前空間にあり、`uid` はホスト上のユーザーの名前と一致しません。グループおよびその他のファイルのアクセス許可は、ゲストシェルユーザーがファイルに対して持つアクセスの種類を制御します。

Guest Shell の Python

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。

例:

```
guestshell:~$ python
Python 2.7.5 (default, Jun 24 2015, 00:41:19)
[GCC 4.8.3 20140911 (Red Hat 4.8.3-9)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
guestshell:~$
```

ネットワーク管理者が新しい Python パッケージをインストールできるように、ゲストシェルには `pip python` パッケージマネージャが含まれています。

例:

```
[guestshell@guestshell ~]$ sudo su
[root@guestshell guestshell]# pip install Markdown
Collecting Markdown
Downloading Markdown-2.6.2-py2.py3-none-any.whl (157kB)
100% |#####| 159kB 1.8MB/s
Installing collected packages: Markdown
Successfully installed Markdown-2.6.2
[root@guestshell guestshell]# pip list | grep Markdown
Markdown (2.6.2)
[root@guestshell guestshell]#
```




(注) **pip install** コマンドを入力する前に、**sudo su** コマンドを入力する必要があります。

Guest Shell バージョン 2.10 までの Python 3 (CentOS 7)

ゲストシェル 2.X は、デフォルトで Python 3 がインストールされていない CentOS 7.1 環境を提供します。CentOS 7.1 に Python 3 をインストールするには、サードパーティのリポジトリを使用する、送信元からビルドするなど、複数の方法があります。別のオプションは、同じシステム内に複数のバージョンの Python のインストールをサポートする Red Hat Software Collections を使用することです。

Red Hat Software Collections (SCL) ツールをインストールするには:

1. **scl-utils** パッケージをインストールします。
2. CentOS SCL リポジトリを有効にして、提供されている Python 3 RPM のいずれかをインストールします。

```
[admin@guestshell ~]$ sudo su
[root@guestshell admin]# dnf install -y scl-utils | tail
Running transaction test
Transaction test succeeded
Running transaction
  Installing : scl-utils-20130529-19.el7.x86_64                1/1
  Verifying  : scl-utils-20130529-19.el7.x86_64                1/1

Installed:
  scl-utils.x86_64 0:20130529-19.el7

Complete!

[root@guestshell admin]# dnf install -y centos-release-scl | tail
Verifying : centos-release-scl-2-3.el7.centos.noarch          1/2
Verifying : centos-release-scl-rh-2-3.el7.centos.noarch       2/2

Installed:
  centos-release-scl.noarch 0:2-3.el7.centos

Dependency Installed:
  centos-release-scl-rh.noarch 0:2-3.el7.centos

Complete!

[root@guestshell admin]# dnf install -y rh-python36 | tail
warning: /var/cache/dnf/x86_64/7/centos-scl-rh/packages/rh-python36-2.0-1.el7.x86_64.rpm:
  Header V4 RSA/SHA1 Signature, key ID f2ee9d55: NOKEY
http://centos.sonn.com/7.7.1908/os/x86_64/Packages/groff-base-1.22.2-8.el7.x86_64.rpm:
[Errno 12] Timeout on
http://centos.sonn.com/7.7.1908/os/x86_64/Packages/groff-base-1.22.2-8.el7.x86_64.rpm: (28,
  'Operation too slow. Less than 1000 bytes/sec transferred the last 30 seconds')
Trying other mirror.
Importing GPG key 0xF2EE9D55:
  Userid      : "CentOS SoftwareCollections SIG
(https://wiki.centos.org/SpecialInterestGroup/SClO) <security@centos.org>"
  Fingerprint: c4db d535 b1fb ba14 f8ba 64a8 4eb8 4e71 f2ee 9d55
  Package     : centos-release-scl-rh-2-3.el7.centos.noarch (@extras)
```

```

From      : /etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-SIG-SCLo
rh-python36-python-libs.x86_64 0:3.6.9-2.el7
rh-python36-python-pip.noarch 0:9.0.1-2.el7
rh-python36-python-setuptools.noarch 0:36.5.0-1.el7
rh-python36-python-virtualenv.noarch 0:15.1.0-2.el7
rh-python36-runtime.x86_64 0:2.0-1.el7
scl-utils-build.x86_64 0:20130529-19.el7
xml-common.noarch 0:0.6.3-39.el7
zip.x86_64 0:3.0-11.el7

```

Complete!

SCL を使用すると、Python 3 の環境変数を自動的に設定して、インタラクティブな bash セッションを作成できます。



(注) SCL Python インストールを使用するためにルート ユーザーは必要ありません。

```

[admin@guestshell ~]$ scl enable rh-python36 bash
[admin@guestshell ~]$ python3
Python 3.6.9 (default, Nov 11 2019, 11:24:16)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-39)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>

```

Python SCL のインストールでは、pip ユーティリティも提供されます。

```

[admin@guestshell ~]$ pip3 install requests --user
Collecting requests
  Downloading
https://files.pythonhosted.org/packages/51/bd/23c926cc841e67db02a00ba99ae0f828e89d73b2190f27c11d4b7fb/requests-2.22.0-py2.py3-none-any.whl
(57kB)
100% |#####| 61kB 211kB/s
Collecting idna<2.9,>=2.5 (from requests)
  Downloading
https://files.pythonhosted.org/packages/14/2c/af51d81be15200be1cf41cd03869a46fe7226e7450af7a6545b0c474c9/idna-2.8-py2.py3-none-any.whl
(58kB)
100% |#####| 61kB 279kB/s
Collecting chardet<3.1.0,>=3.0.2 (from requests)
  Downloading
https://files.pythonhosted.org/packages/cc/a9/01ffebfb562e4274b6487b4bb1dbec7ca55ac7510c22e4c51f14098443b8/chardet-3.0.4-py2.py3-none-any.whl
(133kB)
100% |#####| 143kB 441kB/s
Collecting certifi>=2017.4.17 (from requests)
  Downloading
https://files.pythonhosted.org/packages/19/63/cf50cac98a0cf5006c55a39c3f1db9a765a24de7890bc9cf5db8e99/certifi-2019.11.28-py2.py3-none-any.whl
(156kB)
100% |#####| 163kB 447kB/s
Collecting urllib3!=1.25.0,!1.25.1,<1.26,>=1.21.1 (from requests)
  Downloading
https://files.pythonhosted.org/packages/e8/74/6e4f91745020f967d09332b2b89b10090957334692ab88ea4afe91b77f/urllib3-1.25.8-py2.py3-none-any.whl
(125kB)
100% |#####| 133kB 656kB/s
Installing collected packages: idna, chardet, certifi, urllib3, requests
Successfully installed certifi-2019.11.28 chardet-3.0.4 idna-2.8 requests-2.22.0
urllib3-1.25.8
You are using pip version 9.0.1, however version 20.0.2 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
[admin@guestshell ~]$ python3
Python 3.6.9 (default, Nov 11 2019, 11:24:16)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-39)] on linux
Type "help", "copyright", "credits" or "license" for more information.

```

```
>>> import requests
>>> requests.get("https://cisco.com")
<Response [200]>
```

デフォルトの Python 2 インストールは、SCL Python インストールと一緒に使用できます。

```
[admin@guestshell ~]$ which python3
/opt/rh/rh-python36/root/usr/bin/python3
[admin@guestshell ~]$ which python2
/bin/python2
[admin@guestshell ~]$ python2
Python 2.7.5 (default, Aug 7 2019, 00:51:29)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-39)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> print 'Hello world!'
Hello world!
```

Software Collections を使用すると、同じ RPM の複数のバージョンをシステムにインストールできます。この場合、Python 3.6 に加えて Python 3.5 をインストールすることが可能です。

```
[admin@guestshell ~]$ sudo dnf install -y rh-python35 | tail
Dependency Installed:
rh-python35-python.x86_64 0:3.5.1-13.el7
rh-python35-python-devel.x86_64 0:3.5.1-13.el7
rh-python35-python-libs.x86_64 0:3.5.1-13.el7
rh-python35-python-pip.noarch 0:7.1.0-2.el7
rh-python35-python-setuptools.noarch 0:18.0.1-2.el7
rh-python35-python-virtualenv.noarch 0:13.1.2-2.el7
rh-python35-runtime.x86_64 0:2.0-2.el7
```

Complete!

```
[admin@guestshell ~]$ scl enable rh-python35 python3
Python 3.5.1 (default, May 29 2019, 15:41:33)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-36)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```



(注) 複数の Python バージョンが SCL にインストールされているときに新しいインタラクティブ **bash** セッションを作成すると、**libpython** 共有オブジェクトファイルをロードできないという問題が発生する可能性があります。**source scl_source enable python-installation** コマンドを使用して、現在の **bash** セッションで環境を適切にセットアップできる回避策があります。

デフォルトの Guest Shell ストレージのキャパシティが、Python 3 をインストールするのに十分ではありません。**guestshell resize rootfs size-in-MB** コマンドを使用して、ファイルシステムのサイズを増やします。通常、**rootfs** のサイズを 550 MB に設定すれば十分です。

Guest Shell バージョン 2.10 までの Python 3 (CentOS 7)

ゲストシェル 2.X は、デフォルトで Python 3 がインストールされていない CentOS 7.1 環境を提供します。CentOS 7.1 に Python 3 をインストールするには、サードパーティのリポジトリを使用する、送信元からビルドするなど、複数の方法があります。別のオプションは、同じシステム内に複数のバージョンの Python のインストールをサポートする Red Hat Software Collections を使用することです。

Red Hat Software Collections (SCL) ツールをインストールするには:

1. `scl-utils` パッケージをインストールします。
2. CentOS SCL リポジトリを有効にして、提供されている Python 3 RPM のいずれかをインストールします。

```
[admin@guestshell ~]$ sudo su
[root@guestshell admin]# dnf install -y scl-utils | tail
Running transaction test
Transaction test succeeded
Running transaction
  Installing : scl-utils-20130529-19.el7.x86_64                1/1
  Verifying  : scl-utils-20130529-19.el7.x86_64                1/1

Installed:
  scl-utils.x86_64 0:20130529-19.el7

Complete!

[root@guestshell admin]# dnf install -y centos-release-scl | tail
Verifying : centos-release-scl-2-3.el7.centos.noarch          1/2
Verifying : centos-release-scl-rh-2-3.el7.centos.noarch        2/2

Installed:
  centos-release-scl.noarch 0:2-3.el7.centos

Dependency Installed:
  centos-release-scl-rh.noarch 0:2-3.el7.centos

Complete!

[root@guestshell admin]# dnf install -y rh-python36 | tail
warning: /var/cache/dnf/x86_64/7/centos-scl-rh/packages/rh-python36-2.0-1.el7.x86_64.rpm:
Header V4 RSA/SHA1 Signature, key ID f2ee9d55: NOKEY
http://centos.sonn.com/7.7.1908/os/x86_64/Packages/groff-base-1.22.2-8.el7.x86_64.rpm:
[Errno 12] Timeout on
http://centos.sonn.com/7.7.1908/os/x86_64/Packages/groff-base-1.22.2-8.el7.x86_64.rpm: (28,
'Operation too slow. Less than 1000 bytes/sec transferred the last 30 seconds')
Trying other mirror.
Importing GPG key 0xF2EE9D55:
  Userid      : "CentOS SoftwareCollections SIG
(https://wiki.centos.org/SpecialInterestGroup/SCLo) <security@centos.org>"
  Fingerprint: c4db d535 blfb ba14 f8ba 64a8 4eb8 4e71 f2ee 9d55
  Package     : centos-release-scl-rh-2-3.el7.centos.noarch (@extras)
  From        : /etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-SIG-SCLo
  rh-python36-python-libs.x86_64 0:3.6.9-2.el7
  rh-python36-python-pip.noarch 0:9.0.1-2.el7
  rh-python36-python-setuptools.noarch 0:36.5.0-1.el7
  rh-python36-python-virtualenv.noarch 0:15.1.0-2.el7
  rh-python36-runtime.x86_64 0:2.0-1.el7
  scl-utils-build.x86_64 0:20130529-19.el7
  xml-common.noarch 0:0.6.3-39.el7
  zip.x86_64 0:3.0-11.el7
```

Complete!

SCL を使用すると、Python 3 の環境変数を自動的に設定して、インタラクティブな `bash` セッションを作成できます。



(注) SCL Python インストールを使用するためにルート ユーザーは必要ありません。

```
[admin@guestshell ~]$ scl enable rh-python36 bash
[admin@guestshell ~]$ python3
Python 3.6.9 (default, Nov 11 2019, 11:24:16)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-39)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Python SCL のインストールでは、pip ユーティリティも提供されます。

```
[admin@guestshell ~]$ pip3 install requests --user
Collecting requests
  Downloading
https://files.pythonhosted.org/packages/51/bd/23c926cc341e67d0b2a0aba9ae0f828e89d7262190f27c11d4b7fb/requests-2.22.0-py2.py3-none-any.whl
(57kB)
100% |#####| 61kB 211kB/s
Collecting idna<2.9,>=2.5 (from requests)
  Downloading
https://files.pythonhosted.org/packages/14/2c/af51d81d8e15200be1cf41cd03869a46fe7226e7450af7a6545bfc474c9/idna-2.8-py2.py3-none-any.whl
(58kB)
100% |#####| 61kB 279kB/s
Collecting chardet<3.1.0,>=3.0.2 (from requests)
  Downloading
https://files.pythonhosted.org/packages/bc/a9/01ffebf562e427466487b4d1b1d3c7ca55ec7510b22e4c51f140984438/chardet-3.0.4-py2.py3-none-any.whl
(133kB)
100% |#####| 143kB 441kB/s
Collecting certifi>=2017.4.17 (from requests)
  Downloading
https://files.pythonhosted.org/packages/b9/63/d50ca98a05b006c55a3993bf1db9b7b5a24be7890bc9cf5f39e9/certifi-2019.11.28-py2.py3-none-any.whl
(156kB)
100% |#####| 163kB 447kB/s
Collecting urllib3!=1.25.0,!1.25.1,<1.26,>=1.21.1 (from requests)
  Downloading
https://files.pythonhosted.org/packages/e8/74/6e4f91745020f967cd0932b2b28b10090957334692a88e4afe91b77f/urllib3-1.25.8-py2.py3-none-any.whl
(125kB)
100% |#####| 133kB 656kB/s
Installing collected packages: idna, chardet, certifi, urllib3, requests
Successfully installed certifi-2019.11.28 chardet-3.0.4 idna-2.8 requests-2.22.0
urllib3-1.25.8
You are using pip version 9.0.1, however version 20.0.2 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
[admin@guestshell ~]$ python3
Python 3.6.9 (default, Nov 11 2019, 11:24:16)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-39)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import requests
>>> requests.get("https://cisco.com")
<Response [200]>
```

デフォルトの Python 2 インストールは、SCL Python インストールと一緒に使用できます。

```
[admin@guestshell ~]$ which python3
/opt/rh/rh-python36/root/usr/bin/python3
[admin@guestshell ~]$ which python2
/bin/python2
[admin@guestshell ~]$ python2
Python 2.7.5 (default, Aug 7 2019, 00:51:29)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-39)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
```

```
>>> print 'Hello world!'
Hello world!
```

Software Collections を使用すると、同じ RPM の複数のバージョンをシステムにインストールできます。この場合、Python 3.6 に加えて Python 3.5 をインストールすることが可能です。

```
[admin@guestshell ~]$ sudo dnf install -y rh-python35 | tail
Dependency Installed:
  rh-python35-python.x86_64 0:3.5.1-13.el7
  rh-python35-python-devel.x86_64 0:3.5.1-13.el7
  rh-python35-python-libs.x86_64 0:3.5.1-13.el7
  rh-python35-python-pip.noarch 0:7.1.0-2.el7
  rh-python35-python-setuptools.noarch 0:18.0.1-2.el7
  rh-python35-python-virtualenv.noarch 0:13.1.2-2.el7
  rh-python35-runtime.x86_64 0:2.0-2.el7

Complete!

[admin@guestshell ~]$ scl enable rh-python35 python3
Python 3.5.1 (default, May 29 2019, 15:41:33)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-36)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```



(注) 複数の Python バージョンが SCL にインストールされているときに新しいインタラクティブ bash セッションを作成すると、libpython 共有オブジェクトファイルをロードできないという問題が発生する可能性があります。source scl_source enable python-installation コマンドを使用して、現在の bash セッションで環境を適切にセットアップできる回避策があります。

デフォルトの Guest Shell ストレージのキャパシティが、Python 3 をインストールするのに十分ではありません。guestshell resize rootfs size-in-MB コマンドを使用して、ファイルシステムのサイズを増やします。通常、rootfs のサイズを 550 MB に設定すれば十分です。

ゲストシェルでの RPM のインストール

/etc/dnf/repos.d/CentOS-Base.repo ファイルは、デフォルトで CentOS ミラーリストを使用するように設定されています。変更が必要な場合は、そのファイルの指示に従ってください。

dnf は、yumrepo_x86_64.repo ファイルを変更するか、repos.d ディレクトリに new.repo ファイルを追加することによって、いつでも 1 つ以上のリポジトリを指すことができます。

ゲストシェル 3.0 内にインストールするアプリケーションについては、http://mirror.centos.org/centos/8/BaseOS/x86_64/os/Packages/ にある CentOS 8 リポジトリに移動します。

ゲストシェル 2.x 内にインストールするアプリケーションについては、http://mirror.centos.org/centos/7/os/x86_64/Packages/ にある CentOS 7 リポジトリに移動します。

ゲストシェル 4.x 内にインストールするアプリケーションについては、https://mirrors.rockylinux.org/mirrorlist?arch=x86_64&repo=rocky-BaseOS-9.2 にある RockyLinux 9 リポジトリに移動します。ミラーリンクのいずれかを選択し、パッケージを表示します。

Dnf は依存関係を解決し、必要なすべてのパッケージをインストールします。

```
[guestshell@guestshell ~]$ sudo chvrf management dnf -y install glibc.i686
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
* base: bay.uchicago.edu
* extras: pubmirrors.dal.coreospace.com
* updates: mirrors.cmich.edu
Resolving Dependencies
"--> Running transaction check
"---> Package glibc.i686 0:2.17-78.el7 will be installed
"--> Processing Dependency: libfreebl3.so(NSSRAWHASH_3.12.3) for package:
glibc-2.17-78.el7.i686
"--> Processing Dependency: libfreebl3.so for package: glibc-2.17-78.el7.i686
"--> Running transaction check
"---> Package nss-softokn-freebl.i686 0:3.16.2.3-9.el7 will be installed
"--> Finished Dependency Resolution
```

Dependencies Resolved

Package	Arch	Version	Repository	Size

Installing:

glibc i686 2.17-78.el7 base 4.2 M

Installing for dependencies:

nss-softokn-freebl i686 3.16.2.3-9.el7 base 187 k

Transaction Summary

Install 1 Package (+1 Dependent package)

Total download size: 4.4 M

Installed size: 15 M

Downloading packages:

Delta RPMs disabled because /usr/bin/applydeltarpm not installed.

(1/2): nss-softokn-freebl-3.16.2.3-9.el7.i686.rpm | 187 kB 00:00:25

(2/2): glibc-2.17-78.el7.i686.rpm | 4.2 MB 00:00:30

Total 145 kB/s | 4.4 MB 00:00:30

Running transaction check

Running transaction test

Transaction test succeeded

Running transaction

Installing : nss-softokn-freebl-3.16.2.3-9.el7.i686 1/2

Installing : glibc-2.17-78.el7.i686 2/2

error: lua script failed: [string "%triggerin(glibc-common-2.17-78.el7.x86_64)"]:1: attempt to compare number with nil

Non-fatal "<unknown>" scriptlet failure in rpm package glibc-2.17-78.el7.i686

Verifying : glibc-2.17-78.el7.i686 1/2

Verifying : nss-softokn-freebl-3.16.2.3-9.el7.i686 2/2

Installed:

glibc.i686 0:2.17-78.el7

Dependency Installed:

nss-softokn-freebl.i686 0:3.16.2.3-9.el7

Complete!



- (注) パッケージをインストールまたは実行するためにゲスト シェル ルート ファイル システムにより多くのスペースが必要な場合は、**guestshell resize roots size-in-MB** コマンドを使用してファイル システムのサイズを増やします。



- (注) リポジトリからの一部のオープン ソース ソフトウェア パッケージは、ホストシステムの完全性を保護するために設定された制限の結果として、ゲスト シェルで期待どおりにインストールまたは実行されない場合があります。

仮想サービス ゲスト シェルのセキュリティ ポスチャ

スイッチでのゲスト シェルと仮想サービスの使用は、ネットワーク管理者がシステムの機能を管理または拡張できる多くの方法のうちの2つにすぎません。これらのオプションは、ネイティブ ホスト コンテキストから切り離された実行環境を提供することを目的としています。この分離により、ネイティブの実行環境と互換性がない可能性のあるソフトウェアをシステムに導入できます。また、システムの動作、パフォーマンス、またはスケールに影響を与えない環境でソフトウェアを実行することもできます。

スイッチでのゲスト シェルの使用は、ネットワーク管理者がシステムの機能を管理または拡張できる多くの方法の1つにすぎません。ゲスト シェルは、ネイティブ ホスト コンテキストから切り離された実行環境を提供することを目的としています。この分離により、ネイティブの実行環境と互換性がない可能性のあるソフトウェアをシステムに導入できます。また、システムの動作、パフォーマンス、またはスケールに影響を与えない環境でソフトウェアを実行することもできます。

[デジタル署名されたアプリケーション パッケージ (Digitally Signed Application Packages)]

デフォルトでは、Cisco ネットワーク要素は、アプリケーションのランタイムに有効な Cisco デジタル署名を提供することを要求します。Cisco のデジタル署名により、Cisco が開発したパッケージとアプリケーションの完全性が保証されます。ネットワーク管理者が、仮想サービスを作成するために Cisco 署名されていないソフトウェア パッケージ (OVA ファイル) を使用する場合、**virtual-service** と **signing level unsigned** コマンドを使用して、仮想サービス構成サブモード でネットワーク要素のイメージ署名ポリシーを構成できます。スイッチにインストールする前に、SHA1 または MD5 チェックサムなどの代替手段を使用して、ソフトウェア パッケージ (Cisco 署名されていない) の完全性を保証することをお勧めします。



- (注) Cisco は、Cisco が署名していないソフトウェア パッケージの使用を推奨していません。

デフォルトでは、Cisco ネットワーク要素は、アプリケーションのランタイムに有効な Cisco デジタル署名を提供することを要求します。Cisco のデジタル署名により、Cisco が開発したパッケージとアプリケーションの完全性が保証されます。

Cisco Nexus 3000 9000 シリーズ スイッチは、署名されていない OVA ソフトウェア パッケージを許可する署名レベルポリシーの構成をサポートしています。署名されていないパッケージと Cisco 署名されたパッケージで仮想サービスを作成できるようにするために、ネットワーク管理者は次の構成ができます。

```
virtual-service
  signing level unsigned
```



(注) ゲスト シェル ソフトウェア パッケージには Cisco の署名があり、この構成は必要ありません。

[カーネル脆弱性パッチ (Kernel Vulnerability Patches)]

シスコは、既知の脆弱性に対処するプラットフォーム アップデートで、関連する Common Vulnerabilities and Exposures (CVE) に対応します。



(注) シスコは、Guestshell 4.x (Rocky Linux 9) 環境の脆弱性を追跡しており、将来の修正を、Rocky Linux から入手可能になった時点で含めます。

[ASLR および X-Space のサポート (ASLR and X-Space Support)]

Cisco 3000 9000 NX-OS は、ランタイム ディフェンスのためのアドレス空間 Layout Randomization (ASLR) と Executable Space Protection (X-Space) の使用をサポートしています。Cisco が署名したパッケージのソフトウェアは、この機能を利用します。システムに他のソフトウェアがインストールされている場合は、これらのテクノロジーをサポートするホスト OS と開発ツールチェーンを使用して構築することをお勧めします。これにより、ソフトウェアが潜在的な侵入者に提示する潜在的な攻撃対象領域が減少します。

名前空間の分離

ホストと仮想サービスは、別々の名前空間に分けられます。これは、仮想サービスの実行スペースをホストから分離する基礎を提供します。名前空間の分離は、信頼境界間の偶発的または意図的なデータの上書きによるデータの損失やデータの破損から保護するのに役立ちます。また、機密データ漏洩を防止することにより、機密データの完全性を確保するのに役立ちます。ある仮想サービスのアプリケーションは、別の仮想サービスのデータにアクセスできません。

名前空間の分離

Guest Shell 環境は、さまざまな名前空間を使用して Guest Shell の実行スペースをホストの実行スペースから切り離す Linux コンテナ内で実行されます。NX-OS 9.2(1) リリース以降、Guest Shell は別のユーザー名前空間で実行され、Guest Shell 内でルートとして実行されているプロセスはホストのルートではないため、ホストシステムの整合性を保護するのに役立ちます。これらのプロセスは、uid マッピングのために Guest Shell 内で uid 0 として実行されているように見えますが、カーネルはこれらのプロセスの実際の uid を認識しており、適切なユーザー名前空間内の POSIX 機能を評価します。

ユーザーがホストから Guest Shell に入ると、Guest Shell 内に同じ名前のユーザーが作成されます。名前は一致しますが、Guest Shell 内のユーザーの uid は、ホストの uid と同じではありません。Guest Shell 内のユーザーが共有メディア（たとえば、/bootflash または /volatile）上のファイルに引き続きアクセスできるようにするために、ホストで使用される一般的な NX-OS gid（たとえば、network-admin または network-operator）が Guest Shell にマッピングされます。その際に、値は同じになり、ユーザーの Guest Shell インスタンスがホスト上のグループメンバーシップに基づく適切なグループに関連付けられています。

例として、ユーザー bob について考えてみましょう。ホスト上で、bob には次の uid および gid メンバーシップがあります。

```
bash-4.3$ id
uid=2004(bob) gid=503(network-admin) groups=503(network-admin),504(network-operator)
```

ユーザー bob が Guest Shell にある場合、ホストからのグループ メンバーシップが Guest Shell に設定されます。

```
[bob@guestshell ~]$ id
uid=1002(bob) gid=503(network-admin)
groups=503(network-admin),504(network-operator),10(wheel)
```

ホスト Bash シェルと Guest Shell でユーザー bob によって作成されたファイルの所有者識別子は異なります。以下の出力例は、Guest Shell 内から作成されたファイルの所有者識別子が、上記の出力例の 1002 ではなく 12002 であることを示しています。これは、ホスト Bash シェルから発行されたコマンドと、Guest Shell の識別子スペースが識別子 11000 で始まるためです。ファイルのグループ識別子は network-admin で、両方の環境で 503 です。

```
bash-4.3$ ls -ln /bootflash/bob_*
-rw-rw-r-- 1 12002 503 4 Jun 22 15:47 /bootflash/bob_guestshell
-rw-rw-r-- 1 2004 503 4 Jun 22 15:47 /bootflash/bob_host
```

```
bash-4.3$ ls -l /bootflash/bob_*
-rw-rw-r-- 1 12002 network-admin 4 Jun 22 15:47 /bootflash/bob_guestshell
-rw-rw-r-- 1 bob network-admin 4 Jun 22 15:47 /bootflash/bob_host
```

network-admin グループのファイルパーミッション設定と、bob がホストシェルと Guest Shell の両方で network-admin のメンバーであるため、ユーザーはファイルにアクセスできます。

以下の出力例は、Guest Shell 環境内で、bob によってホストから作成されたファイルの所有者識別子が 65534 であることを示しています。これは、実際の識別子が、ユーザーの名前空間にマップ

された識別子の範囲外の範囲にあることを示しています。マップされていない識別子は、この値として表示されます。

```
[bob@guestshell ~]$ ls -ln /bootflash/bob_*
-rw-rw-r-- 1 1002 503 4 Jun 22 15:47 /bootflash/bob_guestshell
-rw-rw-r-- 1 65534 503 4 Jun 22 15:47 /bootflash/bob_host

[bob@guestshell ~]$ ls -l /bootflash/bob_*
-rw-rw-r-- 1 bob network-admin 4 Jun 22 15:47 /bootflash/bob_guestshell
-rw-rw-r-- 1 65534 network-admin 4 Jun 22 15:47 /bootflash/bob_host
```

ルートユーザーの制限

安全なコードを開発するためのベストプラクティスとして、割り当てられたタスクを実行するために必要な最小限の特権でアプリケーションを実行することを推奨します。意図しないアクセスを防ぐために、Guest Shell に追加されたソフトウェアは、このベストプラクティスに従う必要があります。

仮想サービス内のすべてのプロセスで、Guest Shell は Linux の機能が低下したことによる制限の対象となります。アプリケーションで root 権限を必要とする操作を実行する必要がある場合は、root アカウントの使用を、root アクセスが絶対に必要な最小限の操作セットに制限し、そのモードでアプリケーションを実行できる時間のハード制限などの他の制御を課します。

Guest Shell が従う仮想サービス内のルートに対してドロップされる一連の Linux 機能は次のとおりです。

CAP_SYS_BOOT	CAP_MKNOD	CAP_SYS_PACCT
CAP_SYS_MODULE	CAP_MAC_OVERRIDE	CAP_SYS_RESOURCE
CAP_SYS_TIME	CAP_SYS_RAWIO	CAP_AUDIT_WRITE
CAP_AUDIT_CONTROL	CAP_SYS_NICE	CAP_NET_ADMIN
CAP_MAC_ADMIN	CAP_SYS_PTRACE	

- cap_audit_control
- cap_audit_write
- cap_mac_admin
- cap_mac_override
- cap_mknod
- cap_net_broadcast
- cap_sys_boot
- cap_syslog
- cap_sys_module
- cap_sys_nice

- `cap_sys_pacct`
- `cap_sys_ptrace`
- `cap_sys_rawio`
- `cap_sys_resource`
- `cap_sys_time`
- `cap_wake_alarm`

仮想サービス内のルートとして、`tmpfs` と `ramfs` マウントだけでなくバインドマウントも使用できます。他のマウントは防止されます。

`net_admin` 機能は削除されませんが、ユーザー名前空間とネットワーク名前空間のホスト所有権により、Guest Shell ユーザーはインターフェイスの状態を変更できません。Guest Shell 内の `root` として、`tmpfs` と `ramfs` マウントだけでなくバインドマウントも使用できます。他のマウントは防止されます。

リソース管理

DDoS 攻撃は、攻撃対象のユーザがマシンやネットワーク 技術情報を使用できないようにする試みます。不適切な動作または悪意のあるアプリケーション コードは、接続帯域幅、ディスク容量、メモリ、およびその他のリソースの過剰消費の結果として DoS を引き起こす可能性があります。ホストは、ゲストシェルとホスト上のサービス間のすべての仮想サービス間で技術情報を公平に割り当てる技術情報管理機能を提供します。

ゲスト ファイル システムのアクセス制限

仮想サービス内のファイルの完全性を維持するために、仮想サービスのファイル システムには NX-OS CLI からアクセスできません。特定の仮想サービスがファイルの変更を許可している場合、これを実行できる代替手段 (つまり `yum install`、`scp`、`ftp` など) を提供する必要があります。

ゲスト シェル内のファイルの完全性を維持するために、ゲスト シェルのファイル システムには NX-OS CLI からアクセスできません。

ゲスト シェルは、ホスト システムの [ブートフラッシュ (`bootflash`)] を [/ ブートフラッシュ (`/bootflash`)] にマウントします。ネットワーク管理者は、ゲスト シェル内から NX-OS CLI または Linux コマンドを使用してファイルにアクセスできます。

ホストの [ブートフラッシュ: (`bootflash:`)] と [揮発性: (`volatile:`)] は、ゲストシェル内で [/ ブートフラッシュ (`/bootflash`)] および [/ 揮発性 (`/volatile`)] としてマウントされます。ネットワーク管理者は、ホストから NX-OS `exec` コマンドを使用するか、ゲストシェル内から Linux コマンドを使用して、このメディア上のファイルにアクセスできます。

ゲストシェルの管理

以下は、ゲストシェルを管理するためのコマンドです。

表 4: ゲストシェル **CLI** コマンド

コマンド	説明
guestshell enable { package [<i>guest shell OVA file</i> <i>rootfs-file-URI</i>]}	• [<i>ゲストシェルOVAファイル (guest shell OVA file)</i>] 指定時: システムイメージに組み込まれている OVA を使用して、ゲストシェルをインストールしてアクティブ化します。 指定されたソフトウェアパッケージ (OVA ファイル) またはシステムイメージからの組み込みパッケージ (パッケージが指定されていない場合) を使用して、ゲストシェルをインストールしてアクティブ化します。当初、ゲストシェルパッケージは、システムイメージに埋め込むことによってのみ利用できます。 ゲストシェルがすでにインストールされている場合、このコマンドはインストールされているゲストシェルを有効にします。通常、これは guestshell disable コマンドの後に使用されます。 • <i>rootfs-file-URI</i> が指定されている場合: ゲストシェルが破棄された状態のときに、ゲストシェル rootfs をインポートします。このコマンドは、指定されたパッケージでゲストシェルを起動します。
guestshell export rootfs package <i>destination-file-URI</i>	ゲストシェルの rootfs ファイルをローカル URI (ブートフラッシュ、USB1 など) にエクスポートします。[(7.0 (3) I7 (1) 以降のリリース)]
guestshell disable	シャットダウンとゲストシェルの無効化

コマンド	説明
guestshell upgrade {package [<i>guest shell OVA file</i> <i>rootfs-file-URI</i>]}	• [ゲストシェルOVAファイル (<i>guest shell OVA file</i>)] 指定時: 指定されたソフトウェアパッケージ (OVA ファイル) またはシステムイメージからの組み込みパッケージ (パッケージが指定されていない場合) を使用して、ゲストシェルの非アクティブ化してアップグレードします。当初、ゲストシェルパッケージは、システムイメージに埋め込むことによってのみ利用できます。 ゲスト シェルの現在の rootfs は、ソフトウェアパッケージの rootfs に置き換えられます。ゲストシェルは、アップグレード後も持続するセカンダリ ファイルシステムを利用しません。永続的なセカンダリ ファイルシステムがない場合、guestshell destroy コマンドに続けて guestshell enable コマンドを使用して rootfs を置き換えることもできます。アップグレードが成功すると、ゲストシェルがアクティブ化されます。 アップグレードコマンドを実行する前に、確認を求めるプロンプトが表示されます。 • <i>rootfs-file-URI</i> が指定されている場合: ゲストシェルがすでにインストールされている場合、ゲストシェルの rootfs ファイルをインポートします。このコマンドは、既存のゲスト シェルを削除します。 指定されたパッケージにインストールします。

コマンド	説明
guestshell reboot	ゲストシェルを非アクティブ化してから、再度アクティブ化します。 リブートコマンドを実行する前に、確認を求めるプロンプトが表示されます。 (注) これは、exec モードで guestshell disable コマンドの後に guestshell enable コマンドが続くのと同じです。 これは、ゲストシェル内のプロセスが停止しており、再起動する必要がある場合に役立ちます。この run guestshell コマンドは、ゲストシェルで実行されている <code>sshd</code> に依存しています。 コマンドが機能しない場合は、<code>sshd</code> プロセスが誤って停止した可能性があります。NX-OS CLI からゲストシェルの再起動を実行すると、再起動してコマンドを復元できます。
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。ゲストシェルに関連付けられているすべての技術情報がシステムに返されます。この show virtual-service global コマンドは、これらの技術情報がいつ利用可能になるかを示します。 このコマンドを発行すると、destroy コマンドを実行する前に確認を求めるプロンプトが表示されます。
guestshell run guestshell	シェルプロンプトですでに実行されているゲストシェルに接続します。必要なユーザー名 / パスワード
guestshell run command run guestshell command	ゲスト シェル環境のコンテキスト内で Linux / UNIX コマンドを実行します。 コマンドの実行後、スイッチプロンプトに戻ります。

コマンド	説明
guestshell resize [cpu memory rootfs]	ゲストシェルに割り当てられた使用可能な技術情報を変更します。変更は、次にゲストシェルが有効化または再起動されたときに有効になります。 (注) サイズ変更の値は、guestshell destroy コマンドを使用するとクリアされます。
guestshell sync	アクティブ スーパーバイザとスタンバイ スーパーバイザがあるシステムでは、このコマンドはゲストシェルの格納ファイルをアクティブ スーパーバイザからスタンバイ スーパーバイザに同期します。network-admin は、スタンバイ スーパーバイザが現用系スーパーバイザになったときに同じ rootfs を使用するようにゲストシェル rootfs が設定されているときに、このコマンドを発行します。このコマンドを使用しない場合、スタンバイ スーパーバイザがそのスーパーバイザで利用可能なゲストシェルパッケージを使用してアクティブロールに移行するときに、ゲストシェルが新たにインストールされます。
virtual-service reset force	ゲストシェルまたは仮想サービスを管理できない場合は、システムのリロード後でも、reset コマンドを使用してゲストシェルとすべての仮想サービスを強制的に削除します。クリーンアップを実行するには、システムを再ロードする必要があります。このコマンドを発行した後は、システムがリロードされるまで、ゲストシェルまたは追加の仮想サービスをインストールまたは有効にすることはできません。 リセットを開始する前に確認を求められます。



(注) ゲスト シェル環境を有効化 / 無効化し、アクセスするには、管理者権限が必要です。



(注) ゲストシェルは、ホストシステム上の Linux コンテナ (LXC) として導入されます。NX-OS デバイスでは、LXC は **virtual-service** コマンドでインストールと管理されます。ゲストシェルは、**virtual-service** コマンドに **guestshell+** という名前の仮想サービスとして表示されます。



- (注) ゲストシェルに関係のない仮想サービス コマンドは廃止されます。これらのコマンドは NX-OS 9.2 (1) リリースでは非表示になっており、将来のリリースでは削除されます。

次の **exec** キーワードは廃止予定です。

```
# virtual-service ?
connect  Request a virtual service shell
install  Add a virtual service to install database
uninstall Remove a virtual service from the install database
upgrade  Upgrade a virtual service package to a different version
```

```
# show virtual-service ?
detail  Detailed information config)
```

次の構成キーワードは廃止されます。

```
(config) virtual-service ?
WORD    Virtual service name (Max Size 20)

(config-virt-serv)# ?
activate Activate configured virtual service
description Virtual service description
```

Guest Shell の無効化

guestshell disable コマンドはシャットダウンして、Guest Shell を無効化します。

Guest Shell が無効化された状態でシステムをリロードすると、Guest Shell は無効化されたままになります。

例:

```
switch# show virtual-service list
Virtual Service List:
Name                               Status           Package Name
-----
guestshell+                        Activated        guestshell1.ova
switch# guestshell disable
You will not be able to access your guest shell if it is disabled. Are you sure you want
to disable the guest shell? (y/n) [n] y

2014 Jul 30 19:47:23 switch %$ VDC-1 %$ %VMAN-2-ACTIVATION_STATE: Deactivating virtual
service 'guestshell+'
2014 Jul 30 18:47:29 switch %$ VDC-1 %$ %VMAN-2-ACTIVATION_STATE: Successfully deactivated
virtual service 'guestshell+'
switch# show virtual-service list
Virtual Service List:
Name                               Status           Package Name
-----
guestshell+                        Deactivated      guestshell.ova
```



(注) **guestshell enable** コマンドで Guest Shell が再アクティブ化されます。

ゲスト シェルの破棄

guestshell destroy コマンドは、ゲストシェルとそのアーティファクトをアンインストールします。このコマンドでは、ゲスト シェル OVA は削除されません。

ゲストシェルが破棄された状態でシステムをリロードすると、ゲストシェルは破棄されたままになります。

```
switch# show virtual-service list
Virtual Service List:
Name                Status              Package Name
-----
guestshell+         Deactivated         guestshell.ova

switch# guestshell destroy

You are about to destroy the guest shell and all of its contents. Be sure to save your work.
Are you sure you want to continue? (y/n) [n] y
2014 Jul 30 18:49:10 switch %$ VDC-1 %$ %VMAN-2-INSTALL_STATE: Destroying virtual service
'guestshell+'
2014 Jul 30 18:49:10 switch %$ VDC-1 %$ %VMAN-2-INSTALL_STATE: Successfully destroyed
virtual service 'guestshell +'

switch# show virtual-service list
Virtual Service List:
```



(注) **guestshell enable** コマンドを使用して、ゲスト シェルを再度有効にすることができます。



(注) ゲスト シェルを使用しない場合は、**guestshell destroy** コマンドで削除できます。ゲスト シェルが削除されると、その後のリロードのために削除されたままになります。つまり、ゲストシェルコンテナが削除され、スイッチが再ロードされても、ゲストシェルコンテナは自動的に開始されません。

Guest Shell の有効化

この **guestshell enable** コマンドは、Guest Shell ソフトウェア パッケージから Guest Shell をインストールします。デフォルトでは、システム イメージに埋め込まれたパッケージがインストールに使用されます。Guest Shell が無効化されている場合は、このコマンドを使用して、Guest Shell を再アクティブ化することもできます。

Guest Shell が有効化された状態でシステムをリロードすると、Guest Shell は有効化されたままになります。

例:

```
switch# show virtual-service list
Virtual Service List:
switch# guestshell enable
2014 Jul 30 18:50:27 switch %$ VDC-1 %$ %VMAN-2-INSTALL_STATE: Installing virtual service
'guestshell+'
2014 Jul 30 18:50:42 switch %$ VDC-1 %$ %VMAN-2-INSTALL_STATE: Install success virtual
service 'guestshell+'; Activating

2014 Jul 30 18:50:42 switch %$ VDC-1 %$ %VMAN-2-ACTIVATION_STATE: Activating virtual service
'guestshell+'
2014 Jul 30 18:51:16 switch %$ VDC-1 %$ %VMAN-2-ACTIVATION_STATE: Successfully activated
virtual service 'guestshell+'

switch# show virtual-service list
Virtual Service List:
Name                Status              Package Name
guestshell+         Activated           guestshell.ova
```

ベース ブート モードでの Guest Shell の有効化

NX-OS 9.2(1) リリース以降、システムを [基本ブート モード (*base boot mode*)] でブートすることを選択できます。システムを基本ブート モードで起動すると、Guest Shell はデフォルトでは開始されません。このモードで Guest Shell を使用するには、仮想化インフラストラクチャと Guest Shell イメージを含む RPM をアクティブにする必要があります。これを行うと、Guest Shell と virtual-service コマンドが使用できるようになります。

RPM アクティベーション コマンドが次の順序で実行された場合:

1. install activate guestshell
2. install activate virtualization

Guest Shell コンテナは、システムがフル モードで起動した場合と同様に自動的にアクティブ化されます。

RPM アクティベーション コマンドを逆の順序で実行した場合:

1. install activate virtualization
2. install activate guestshell

その後、[**guestshell** を有効化 (**guestshell enable**)] コマンドを実行するまで、Guest Shell は有効になりません。

圧縮されたイメージを使用した Cisco Nexus 3000 での Guest Shell の有効化

Guest Shell ソフトウェアは、1.6 GB のブートフラッシュと 4 GB の RAM を備えた Cisco Nexus 3000 シリーズ スイッチ用に圧縮された Cisco NX-OS イメージでは使用できません。この場合も引き続き Guest Shell を使用できますが、software.cisco.com から Cisco NX-OS リリース用のソフトウェア

パッケージをダウンロードしてから、それを Cisco Nexus 3000 シリーズ スイッチにコピーして有効にする必要があります。それ。

コンパクト イメージの詳細については、『Cisco Nexus 3000 Series NX-OS Software Upgrade and Downgrade Guide, Release 9.2(1)』を参照してください。

Guest Shell ソフトウェアは、スイッチのブートフラッシュにインストールされます。できるだけ多くの空きブートフラッシュスペースを作成するには、ダウンロードした guestshell.ova ファイルを volatile: ストレージメディアに置きます。Guest Shell が正常にアクティブ化されたら、guestshell.ova ファイルを削除できます。ある時点で Guest Shell が破棄され、再インストールする必要がない限り、再度必要になることはありません。

例:

```
switch# copy scp://admin@1.2.3.4/guestshell.ova volatile: vrf management
guestshell.ova 100% 55MB 10.9MB/s 00:05
Copy complete, now saving to disk (please wait)...
Copy complete.

switch# dir volatile: | inc .ova
57251840 Jun 22 11:56:51 2018 guestshell.ova

switch# guestshell enable package volatile:guestshell.ova
2018 Jun 7 19:13:03 n3x-164 %$ VDC-1 %$ %VMAN-2-INSTALL_STATE: Installing virtual service
'guestshell+'
2018 Jun 7 19:13:56 n3x-164 %$ VDC-1 %$ %VMAN-2-INSTALL_STATE: Install success virtual
service 'guestshell+'; Activating
2018 Jun 7 19:13:56 n3x-164 %$ VDC-1 %$ %VMAN-2-ACTIVATION_STATE: Activating virtual service
'guestshell+'
2018 Jun 7 19:15:34 n3x-164 %$ VDC-1 %$ %VMAN-2-ACTIVATION_STATE: Successfully activated
virtual service 'guestshell+'

switch# del volatile:guestshell.ova
Do you want to delete "/guestshell.ova" ? (yes/no/abort) [y] y

switch# guestshell
[admin@guestshell ~]$
```

ゲスト シェルの複製

Cisco NX-OS リリース 7.0 (3) I7 (1) 以降、1 つのスイッチでカスタマイズされたゲスト シェル **rootfs** を複数のスイッチに展開できます。

アプローチは、ゲスト シェル **rootfs** をカスタマイズしてからエクスポートし、ファイルサーバに保存することです。POAP スクリプトは、ゲスト シェル **rootfs** を他のスイッチにダウンロード (インポート) し、特定のゲスト シェルを多数のデバイスに同時にインストールできます。

ゲスト シェル **rootfs** のエクスポート

ゲスト シェル **rootfs** をエクスポートするには、**guestshell export rootfs packagedestination-file-URI** コマンドを使用します。

destination-file-URI パラメータは、ゲスト シェル **rootfs** のコピー先のファイルの名前です。このファイルでは、ローカル URI オプション (ブートフラッシュ、USB1 など) が可能です。

guestshell export rootfs package コマンドでは、次の処理が行われます。

- ゲスト シェルを無効にします（すでに有効になっている場合）。
- ゲスト シェルインポート YAML ファイルを作成し、**rootfs ext4** ファイルの **/cisco** ディレクトリに挿入します。
- **rootfs ext4** ファイルをターゲット URI の場所にコピーします。
- ゲスト シェルが以前に有効になっていた場合は、再度有効にします。

Guest Shell rootfs のインポート

Guest Shell **rootfs** をインポートする場合、考慮すべき 2 つの状況があります。

- Guest Shell が破棄された状態の場合は、**guestshell enable package rootfs-file-URI** コマンドを使用して、Guest Shell **rootfs** をインポートします。このコマンドは、指定されたパッケージで Guest Shell を起動します。
- Guest Shell がすでにインストールされている場合は、**guestshell upgrade package rootfs-file-URI** コマンドを使用して、Guest Shell **rootfs** をインポートします。このコマンドは、既存の Guest Shell を削除し、指定されたパッケージをインストールします。

rootfs-file-URI パラメータは、ローカルストレージ（ブートフラッシュ、USB など）に保存されている **rootfs** ファイルです。

ブートフラッシュにあるファイルでこのコマンドを実行すると、ファイルはブートフラッシュのストレージプールに移動されます。

ベストプラクティスとして、**guestshell upgrade package rootfs-file-URI** コマンドを使用する前に、ファイルをブートフラッシュにコピーし、md5sum を検証する必要があります。



（注） **guestshell upgrade package rootfs-file-URI** コマンドは、Guest Shell 内から実行



（注） **rootfs** ファイルはシスコの署名付きパッケージではありません。例に示すように、有効にする前に、署名されていないパッケージを許可するように設定する必要があります。

```
(config-virt-serv-global)# signing level unsigned
```

```
Note: Support for unsigned packages has been user-enabled. Unsigned packages are not endorsed by Cisco. User assumes all responsibility.
```



(注) rootfs の組み込みバージョンを復元するには:

- Guest Shell が既にインストールされている場合は、**guestshell upgrade** コマンドを（追加のパラメーターなしで）使用します。
- Guest Shell が破棄されたときに、**guestshell enable** コマンドを（追加パラメータなしで）使用します。



(注) Guest Shell 内から、または NX-API を使用してスイッチの外部からこのコマンドを実行する場合は、プロンプトをスキップするように設定する必要があります。**terminal dont-ask**

guestshell enable package rootfs-file-URI コマンド:

- **rootfs** ファイルの基本的な検証を実行します。
- **rootfs** をストレージ プールに移動します。
- **rootfs** をマウントして、/cisco ディレクトリから YAML ファイルを抽出します。
- YAML ファイルを解析して VM 定義（リソース要件を含む）を取得します。
- Guest Shell をアクティブにします。

guestshell enable のワークフローの例:

```
switch# copy scp://user@10.1.1.1/my_storage/gs_rootfs.ext4 bootflash: vrf management
switch# guestshell resize cpu 8
Note: System CPU share will be resized on Guest shell enable
switch# guestshell enable package bootflash:gs_rootfs.ext4
Validating the provided rootfs
switch# 2017 Jul 31 14:58:01 switch %$ VDC-1 %$ %VMAN-2-INSTALL_STATE: Installing virtual
service 'guestshell+'
2017 Jul 31 14:58:09 switch %$ VDC-1 %$ %VMAN-2-INSTALL_STATE: Install success virtual
service 'guestshell+'; Activating
2017 Jul 31 14:58:09 switch %$ VDC-1 %$ %VMAN-2-ACTIVATION_STATE: Activating virtual service
'guestshell+'
2017 Jul 31 14:58:33 switch %$ VDC-1 %$ %VMAN-2-ACTIVATION_STATE: Successfully activated
virtual service 'guestshell+'
```



(注) **guestshell upgrade** のワークフローの前に、既存の Guest Shell が破棄されます。



(注) サイズ変更の値は、**guestshell upgrade** コマンドを使用するとクリアされます。

YAML ファイルのインポート

Guest Shell のユーザーが変更可能ないくつかの特性を定義する YAML ファイルは、エクスポート操作の一部として自動的に作成されます。これは、/cisco ディレクトリの Guest Shell **rootfs** に組み込まれています。これは、Guest Shell コンテナの完全な記述子ではありません。ユーザーが変更できるパラメータの一部のみが含まれています。

Guest Shell インポート YAML ファイルの例:

```
---
import-schema-version: "1.0"
info:
  name: "GuestShell"
  version: "2.2(0.3)"
  description: "Exported GuestShell: 20170216T175137Z"
app:
  apptype: "lxc"
  cpuarch: "x86_64"
  resources:
    cpu: 3
    memory: 307200
    disk:
      - target-dir: "/"
        capacity: 250
...
```

guestshell export rootfs package コマンドを実行すると、YAML ファイルが生成されます。このファイルは、現在実行中の Guest Shell の値をキャプチャします。

情報セクションには、Guest Shell の識別に役立つ非運用データが含まれています。**show guestshell detail** コマンドの出力に一部の情報が表示されます。

説明の値は、YAML ファイルが作成されたときの UTC 時間のエンコーディングです。時刻文字列のフォーマットは、RFC5545 (iCal) の DTSTAMP と同じです。

リソース セクションでは、Guest Shell をホストするために必要な情報技術について説明します。この例の **target-dir** の値「/」は、ディスクを **rootfs** として識別します。



(注) Guest Shell が破棄されたときにサイズ変更された値が指定された場合、**guestshell enable package** コマンドの使用時にそれらの値がインポート YAML ファイルの値よりも優先されます。

cpuarch 値は、コンテナの実行が予想される CPU アーキテクチャを示します。

エクスポート操作が完了した後、YAML ファイルを変更できます（説明などを変更したり、必要に応じて技術情報パラメータを増やしたりできます）。

Cisco は、JSON スキーマを使用して変更された YAML ファイルを検証するために実行できる Python スクリプトを提供しています。完全なテストではありませんが（たとえば、デバイス固有のリソース制限はチェックされません）、一般的なエラーにフラグを付けることができます。例を含む Python スクリプトは、[Guest Shell インポート エクスポート (Guest Shell Import Export)] https://github.com/datacenter/opennxsos/tree/master/guestshell_import_export にあります。次の JSON ファイルは、Guest Shell インポート YAML のバージョン 1.0 のスキーマを記述しています。

```

{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "Guest Shell import schema",
  "description": "Schema for Guest Shell import descriptor file - ver 1.0",
  "copyright": "2017 by Cisco systems, Inc. All rights reserved.",
  "id": "",
  "type": "object",
  "additionalProperties": false,
  "properties": {
    "import-schema-version": {
      "id": "/import-schema-version",
      "type": "string",
      "minLength": 1,
      "maxLength": 20,
      "enum": [
        "1.0"
      ]
    },
    "info": {
      "id": "/info",
      "type": "object",
      "additionalProperties": false,
      "properties": {
        "name": {
          "id": "/info/name",
          "type": "string",
          "minLength": 1,
          "maxLength": 29
        },
        "description": {
          "id": "/info/description",
          "type": "string",
          "minLength": 1,
          "maxLength": 199
        },
        "version": {
          "id": "/info/version",
          "type": "string",
          "minLength": 1,
          "maxLength": 63
        },
        "author-name": {
          "id": "/info/author-name",
          "type": "string",
          "minLength": 1,
          "maxLength": 199
        },
        "author-link": {
          "id": "/info/author-link",
          "type": "string",
          "minLength": 1,
          "maxLength": 199
        }
      }
    },
    "app": {
      "id": "/app",
      "type": "object",
      "additionalProperties": false,
      "properties": {
        "apptype": {
          "id": "/app/apptype",
          "type": "string",

```



```

    "minLength": 1,
    "maxLength": 63,
    "enum": [
      "lxc"
    ]
  },
  "cpuarch": {
    "id": "/app/cpuarch",
    "type": "string",
    "minLength": 1,
    "maxLength": 63,
    "enum": [
      "x86_64"
    ]
  },
  "resources": {
    "id": "/app/resources",
    "type": "object",
    "additionalProperties": false,
    "properties": {
      "cpu": {
        "id": "/app/resources/cpu",
        "type": "integer",
        "multipleOf": 1,
        "maximum": 100,
        "minimum": 1
      },
      "memory": {
        "id": "/app/resources/memory",
        "type": "integer",
        "multipleOf": 1024,
        "minimum": 1024
      },
      "disk": {
        "id": "/app/resources/disk",
        "type": "array",
        "minItems": 1,
        "maxItems": 1,
        "uniqueItems": true,
        "items": {
          "id": "/app/resources/disk/0",
          "type": "object",
          "additionalProperties": false,
          "properties": {
            "target-dir": {
              "id": "/app/resources/disk/0/target-dir",
              "type": "string",
              "minLength": 1,
              "maxLength": 1,
              "enum": [
                "/"
              ]
            }
          },
          "file": {
            "id": "/app/resources/disk/0/file",
            "type": "string",
            "minLength": 1,
            "maxLength": 63
          },
          "capacity": {
            "id": "/app/resources/disk/0/capacity",
            "type": "integer",
            "multipleOf": 1,
            "minimum": 1
          }
        }
      }
    }
  }
}

```

```

    }
  }
}
},
"required": [
  "memory",
  "disk"
]
}
},
"required": [
  "apptype",
  "cpuarch",
  "resources"
]
}
},
"required": [
  "app"
]
}
}

```

show guestshell コマンド

show guestshell detail コマンドの出力には、ゲスト シェルがインポートされたか、OVA からインポートされたかを示す情報が含まれます。

rootfsをインポートした後の **show guestshell detail** コマンドの例。

```

switch# show guestshell detail
Virtual service guestshell+ detail
  State                : Activated
  Package information
    Name                : rootfs_puppet
    Path                 : usb2:/rootfs_puppet
  Application
    Name                 : GuestShell
    Installed version    : 3.0(0.0)
    Description          : Exported GuestShell: 20170613T173648Z
  Signing
    Key type             : Unsigned
    Method                : Unknown
  Licensing
    Name                 : None
    Version               : None

```

仮想サービスと Guest Shell 情報の検証

次のコマンドを使用して、仮想サービスとゲスト シェルの情報を検証できます。

コマンド	説明												
show virtual-service global <pre>switch# show virtual-service global</pre> Virtual Service Global State and Virtualization Limits: Infrastructure version : 1.11 Total virtual services installed : 1 Total virtual services activated : 1 Machine types supported : LXC Machine types disabled : KVM Maximum VCPUs per virtual service : 1 Resource virtualization limits: Name Quota Committed Available ----- <pre>system CPU (%) 20 1 19 memory (MB) 3840 256 3584 bootflash (MB) 8192 200 7992 switch#</pre>	仮想サービスのグローバル状態と制限を表示します。												
show virtual-service list <pre>switch# show virtual-service list *</pre> Virtual Service List: <table><thead><tr><th>Name</th><th>Status</th><th>Package Name</th></tr></thead><tbody><tr><td colspan="3">-----</td></tr><tr><td>guestshell+</td><td>Activated</td><td>guestshell.ova</td></tr><tr><td>chef</td><td>Installed</td><td>chef-0.8.1-n9000-spa-k9.ova</td></tr></tbody></table>	Name	Status	Package Name	-----			guestshell+	Activated	guestshell.ova	chef	Installed	chef-0.8.1-n9000-spa-k9.ova	仮想サービスの概要、仮想サービスのステータス、およびインストールされているソフトウェアパッケージを表示します。
Name	Status	Package Name											

guestshell+	Activated	guestshell.ova											
chef	Installed	chef-0.8.1-n9000-spa-k9.ova											

コマンド	説明
show guestshell detail <pre>switch# show guestshell detail Virtual service guestshell+ detail State : Activated Package information Name : guestshell.ova Path : /isan/bin/guestshell.ova Application Name : GuestShell Installed version : 3.0(0.0) Description : Cisco Systems Guest Shell Signing Key type : Cisco key Method : SHA-1 Licensing Name : None Version : None Resource reservation Disk : 400 MB Memory : 256 MB CPU : 1% system CPU Attached devices Type Name Alias ----- Disk rootfs Disk /cisco/core Serial/shell Serial/aux Serial/Syslog serial2 Serial/Trace serial3</pre>	guestshell パッケージに関する詳細（バージョン、署名リソース、デバイスなど）を表示します。

ゲスト シェルからのアプリケーションの永続的な起動

アプリケーションには、`/usr/lib/systemd/system/application_name.service` にインストールされる `systemd` / `systemctl` サービスファイルが必要です。このサービスファイルは、次の一般的なフォーマットにする必要があります。

```
[Unit]
Description=Put a short description of your application here

[Service]
ExecStart=Put the command to start your application here
Restart=always
RestartSec=10s

[Install]
WantedBy=multi-user.target
```



(注) 特定のユーザーとして systemd を実行するには、サービスの [サービス (Service)] セクションに `User=<username>` を追加します。

Guest Shell からアプリケーションを永続的に起動する手順

手順

- Step 1** 上記で作成したアプリケーションサービスファイルを `/usr/lib/systemd/system/application_name` にインストールします。サービス
- Step 2** `systemctl start application_name` でアプリケーションを開始します
- Step 3** アプリケーションが `systemctl status -l application_name` で実行されていることを確認します
- Step 4** `systemctl enable application_name` でリロード時にアプリケーションを再起動できるようにします
- Step 5** アプリケーションが `systemctl status -l application_name` で実行されていることを確認します

ゲスト シェルでのサンプル アプリケーション

次の例は、ゲスト シェルのアプリケーションを示しています。

```
root@guestshell guestshell]# cat /etc/init.d/hello.sh
#!/bin/bash

OUTPUTFILE=/tmp/hello

rm -f $OUTPUTFILE
while true
do
    echo $(date) >> $OUTPUTFILE
    echo 'Hello World' >> $OUTPUTFILE
    sleep 10
done
[root@guestshell guestshell]#
[root@guestshell guestshell]#
[root@guestshell system]# cat /usr/lib/systemd/system/hello.service
[Unit]
Description=Trivial "hello world" example daemon

[Service]
ExecStart=/etc/init.d/hello.sh &
Restart=always
RestartSec=10s
```

```
[Install]
WantedBy=multi-user.target
[root@guestshell system]#
[root@guestshell system]# systemctl start hello
[root@guestshell system]# systemctl enable hello
[root@guestshell system]# systemctl status -l hello
hello.service - Trivial "hello world" example daemon
   Loaded: loaded (/usr/lib/systemd/system/hello.service; enabled)
   Active: active (running) since Sun 2015-09-27 18:31:51 UTC; 10s ago
 Main PID: 355 (hello.sh)
   CGroup: /system.slice/hello.service
           ##355 /bin/bash /etc/init.d/hello.sh &
           ##367 sleep 10

Sep 27 18:31:51 guestshell hello.sh[355]: Executing: /etc/init.d/hello.sh &
[root@guestshell system]#
[root@guestshell guestshell]# exit
exit
[guestshell@guestshell ~]$ exit
logout
switch# reload
This command will reboot the system. (y/n)? [n] y
```

リロード後

```
[root@guestshell guestshell]# ps -ef | grep hello
root      20      1  0 18:37 ?        00:00:00 /bin/bash /etc/init.d/hello.sh &
root      123    108  0 18:38 pts/4    00:00:00 grep --color=auto hello
[root@guestshell guestshell]#
[root@guestshell guestshell]# cat /tmp/hello
Sun Sep 27 18:38:03 UTC 2015
Hello World
Sun Sep 27 18:38:13 UTC 2015
Hello World
Sun Sep 27 18:38:23 UTC 2015
Hello World
Sun Sep 27 18:38:33 UTC 2015
Hello World
Sun Sep 27 18:38:43 UTC 2015
Hello World
[root@guestshell guestshell]#
```

systemd / systemctl で実行すると、アプリケーションが停止した場合（または強制終了した場合）、アプリケーションは自動的に再起動されます。プロセス識別子はもともと 226 です。アプリケーションを強制終了すると、プロセス 識別子 257 で自動的に再起動されます。

```
[root@guestshell guestshell]# ps -ef | grep hello
root      226      1  0 19:02 ?        00:00:00 /bin/bash /etc/init.d/hello.sh &
root      254    116  0 19:03 pts/4    00:00:00 grep --color=auto hello
[root@guestshell guestshell]#
[root@guestshell guestshell]# kill -9 226
[root@guestshell guestshell]#
[root@guestshell guestshell]# ps -ef | grep hello
root      257      1  0 19:03 ?        00:00:00 /bin/bash /etc/init.d/hello.sh &
root      264    116  0 19:03 pts/4    00:00:00 grep --color=auto hello
[root@guestshell guestshell]#
```

Guest Shell に関する問題のトラブルシューティング

7.0 (3) I7 へのダウングレード後にゲストシェルにアクセスできない

ゲストシェルのアクティブ化または非アクティブ化のプロセス中に、NX-OS 9.2 (1) リリースから NX-OS 7.0 (3) 7 リリース イメージ (ユーザー名前空間のサポートがない) にダウングレードした場合、次のコマンドを実行できます。ゲストシェルは起動しますが、ゲストシェルにアクセスできない次の状態になります。この問題の理由は、ゲストシェルの移行中にリロードが発行された場合、ゲストシェル内のファイルがユーザー名前空間のサポートがない NX-OS リリースで使用可能な識別子範囲に戻されないためです。

```
switch# guestshell
Failed to mkdir .ssh for admin
admin RSA add failed
ERROR: Failed to connect with Virtual-service 'guestshell+'
switch#
switch# sh virt list

Virtual Service List:
Name              Status           Package Name
-----
guestshell+       Activated        guestshell.ova

switch# run bash ls -al /isan/vdc_1/virtual-instance/guestshell+/rootfs/
drwxr-xr-x  24 11000 11000 1024 Apr 11 10:44 .
drwxrwxrwx   4 root  root   80 Apr 27 20:08 ..
-rw-r--r--   1 11000 11000   0 Mar 21 16:24 .autorelabel
lrwxrwxrwx   1 11000 11000   7 Mar 21 16:24 bin -> usr/bin
```

ゲストシェルの格納ファイルを失うことなくこの問題から回復するには、以前に実行されていた NX-OS 9.2 (x) イメージを使用してシステムをリロードし、NX-OS 7.0 (3) I7 イメージでシステムをリロードする前に、ゲストシェルが [アクティブ化 (Activated)] された状態になるようにします。もう 1 つのオプションは、NX-OS 9.2 (x) の実行中にゲストシェルを無効にし、7.0 (3) I7 でリロードした後に再度有効にすることです。

ゲストシェルに保存するものがなく、復元するだけの場合は、イメージを変更せずに破棄して再作成できます。

ゲストシェルのルートからブートフラッシュのファイルにアクセスできない

ゲストシェルのルートからブートフラッシュのファイルにアクセスできない場合があります。

ホストから:

```
root@switch# ls -al /bootflash/try.that
-rw-r--r-- 1 root root 0 Apr 27 20:55 /bootflash/try.that
root@switch#
```

ゲストシェルから:

```
[root@guestshellbootflash]# ls -al /bootflash/try.that
-rw-r--r-- 1 65534 host-root 0 Apr 27 20:55 /bootflash/try.that
[root@guestshellbootflash]# echo "some text" >> /bootflash/try.that
```

```
-bash: /bootflash/try.that: Permission denied  
[root@guestshellbootflash]#
```

これは、ユーザーの名前空間がホスト システムを保護するために使用されているため、ゲスト シェルのルートが実際にはシステムのルートではないことが原因である可能性があります。

この問題から回復するには、ファイルのアクセス許可とファイルのグループ識別子で、ブートフラッシュ上の共有ファイルに期待どおりにアクセスできることを確認します。ホスト Bash セッションからアクセス許可またはグループ 識別子 を変更する必要がある場合があります。



第 5 章

Python API

- [Python API の概要 \(69 ページ\)](#)
- [Python の使用 \(70 ページ\)](#)

Python API の概要

Cisco NX-OS リリース 9.3(5) 以降、Python 3 はサポート対象です：Python 2.7 は引き続きサポートされます。新しいスクリプトには、**python3** コマンドを使用することをお勧めします。

Cisco Nexus 3500 プラットフォーム スイッチは、インタラクティブ モードと非インタラクティブ（スクリプト）モードの両方で Python v2.7.11 [と v3.7.3 (and v3.7.3)] をサポートし、ゲストシェルで使用できます。

Python は簡単に習得できる強力なプログラミング言語です。効率的で高水準なデータ構造を持ち、オブジェクト指向プログラミングに対してシンプルで効果的なアプローチを取っています。Python は、簡潔な構文、動的な型指定、インタープリタ型という特長を持っており、ほとんどのプラットフォームのさまざまな分野でスクリプティングと高速アプリケーション開発が可能な理想的な言語です。

Python インタープリタと広範な標準規格ライブラリが Python Web サイトで送信元形式またはバイナリ形式で自由に利用できます：

<http://www.python.org/>

また、このサイトには、サードパーティが無償で提供している多数の Python モジュール、プログラム、ツールのディストリビューションとそれらへのリンク、さらに追加のドキュメンテーションが掲載されています。

Python スクリプト機能は、さまざまなタスクを実行するためにデバイスのコマンドライン インターフェイス（CLI）Power On Auto Provisioning（POAP）または Embedded Event Manager（EEM）アクションへのプログラムによるアクセスを提供します。Python は Bash シェルからアクセスできます。

Python インタープリタは Cisco NX-OS ソフトウェアで使用できます。

Cisco Nexus デバイスでの Python の使用については、<https://developer.cisco.com/site/nx-os/docs/apis/python/> の URL にある Cisco Nexus 9000 Series Python

SDK User Guide and API Referenceを参照してください。[この情報は、Cisco Nexus 3000 シリーズデバイスにも適用されます。（This information applies also to the Cisco Nexus 3000 Series devices.）]

Python の使用

ここでは、Python スクリプトの作成と実行の方法について説明します。

Cisco Python パッケージ

Cisco NX-OS は、インターフェイス、VLAN、VRF、ACL、ルートなど、多くのコア ネットワーク デバイス モジュールへのアクセスを可能にする Cisco Python パッケージを提供します。**help()** コマンドを入力すると、Cisco Python パッケージの詳細を表示できます。モジュール内のクラスとメソッドに関する追加情報を取得するには、特定のモジュールに対して **help** コマンドを実行します。たとえば、**help(cisco.interface)** は、**cisco.interface** モジュールのプロパティを表示します。

その他の Python モジュールについては、https://devhub.cisco.com/artifactory/open-nxos/9.2.1/x86_64/ から **python-modules-nxos RPM** (**python-modules-nxos-1.0.0-9.2.1.lib32_x86.rpm**) をインストールできます。RPM のインストール手順については、「機能 RPM の管理」セクションを参照してください。

次の例は、Cisco Python パッケージに関する情報を表示する方法を示します。

```
>>> import cisco
>>> help(cisco)
Help on package cisco:

NAME
    cisco

FILE
    /isan/python/scripts/cisco/__init__.py

PACKAGE CONTENTS
    acl
    bgp
    cisco_secret
    cisco_socket
    feature
    interface
    key
    line_parser
    md5sum
    nxcli
    ospf
    routemap
    routes
    section_parser
    ssh
    system
    tacacs
    vrf

CLASSES
    __builtin__.object
        cisco.cisco_secret.CiscoSecret
```

```
cisco.interface.Interface
cisco.key.Key
```

次に、Python 3 用の Cisco Python パッケージに関する情報を表示する方法の例を示します。

```
switch# python3
Python 3.7.3 (default, Nov 20 2019, 14:38:01)
[GCC 5.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import cisco
>>> help(cisco)
Help on package cisco:

NAME
cisco

PACKAGE CONTENTS
acl
bgp
buffer_depth_monitor
check_port_discards
cisco_secret
feature
historys
interface
ipaddress
key
line_parser
mac_address_table
md5sum
nxcli
nxos_cli
ospf
routemap
routes
section_parser
ssh
system
tacacs
transfer
vlan
vrf

CLASSES
builtins.dict(builtins.object)
cisco.history.History
builtins.object
cisco.cisco_secret.CiscoSecret
cisco.interface.Interface
cisco.key.Key
```

CLI コマンド API の使用

Python プログラミング言語は、CLI コマンドを実行できる 3 つの API を使用します。API は Python CLI モジュールから利用できます。

これらの API については、次の表で説明します。* **from cli import** コマンドを使用して API を有効にする必要があります。これらの API 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 API 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。

表 5: CLI コマンド API

API	説明
cli() 例: <pre>string = cli ("cli-command")</pre>	制御文字または特殊文字を含む CLI コマンドの未処理の出力を返します。 (注) インタラクティブな Python インタープリタは、制御文字または特殊文字を「エスケープ」して出力します。キャリッジリターンは '\n' として出力され、結果が読みにくい場合があります。 clip() API は、判読性が高い結果を出力します。
clid() 例: <pre>json_string = clid ("cli-command")</pre>	cli-command コマンドに XML サポートが存在する場合は、JSON 出力を返します。それ以外の場合は、例外がスローされます。 (注) この API は、 show コマンドの出力の検索時に使用すると便利な場合があります。
clip() 例: <pre>clip ("cli-command")</pre>	CLI コマンドの出力を直接 stdout に出力し、Python には何も返されません。 (注) <pre>clip ("cli-command")</pre> と同等です (is equivalent to) <pre>r=cli("cli-command") print r</pre>

2 つ以上のコマンドを個別に実行すると、その状態は 1 つのコマンドから後続のコマンドまで持続しません。

次の例では、最初のコマンドの状態が 2 番目のコマンドで持続しないため、2 番目のコマンドが失敗します。

```
>>> cli("conf t")
>>> cli("interface eth4/1")
```

2 つ以上のコマンドを同時に実行すると、その状態は 1 つのコマンドから後続のコマンドまで持続します。

次の例では、2 番目と 3 番目のコマンドの状態が持続するため、2 番目のコマンドは成功しています。

```
>>> cli("conf t ; interface eth4/1 ; shut")
```



(注) 例に示すように、コマンドは「;」で区切られます。セミコロン(;)は、単一のブランク文字で囲む必要があります。

CLI からの Python インタープリタの呼び出し

次に、CLI から Python 2 を呼び出す方法を表示します:



(注) Python インタープリタのプロンプトは「>>>」または「...」で表示されます。



重要 Python 2.7 はサポート終了です。将来の Cisco NX-OS ソフトウェアは Python 2.7 のサポートを廃止します。新しいスクリプトの代わりに **python3** を使用することをお勧めします。新しいシェルを使用するように **python3** を入力します。

```
switch# python
switch# python
```

```
Warning: Python 2.7 is End of Support, and future NXOS software will deprecate
python 2.7 support. It is recommended for new scripts to use 'python3' instead.
Type "python3" to use the new shell.
```

```
Python 2.7.11 (default, Jun  4 2020, 09:48:24)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> from cli import *
>>> import json
>>> cli('configure terminal ; interface loopback 1 ; no shut')
''
>>> intflist=json.loads(clid('show interface brief'))
>>> i=0
>>> while i < len(intflist['TABLE_interface']['ROW_interface']):
...     intf=intflist['TABLE_interface']['ROW_interface'][i]
...     i=i+1
...     if intf['state'] == 'up':
...         print intf['interface']
...
mgmt0
loopback1
>>>
```

次に、CLI から Python 3 を呼び出す方法を表示します:

```
switch# python3
Python 3.7.3 (default, Nov 20 2019, 14:38:01)
[GCC 5.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> from cli import *
>>> import json
```

```
>>> cli('configure terminal ; interface loopback 1 ; no shut')
''
>>> intflist=json.loads(clid('show interface brief'))
>>> i=0
>>> while i < len(intflist['TABLE_interface']['ROW_interface']):
...     intf=intflist['TABLE_interface']['ROW_interface'][i]
...     i=i+1
...     if intf['state'] == 'up':
...         print(intf['interface'])
...
mgmt0
loopback1
>>>
```

表示フォーマット

次に、Python API を使用したさまざまな表示フォーマットを示します:

例 1:

```
>>> from cli import *
>>> cli("conf ; interface loopback 1")
''
>>> clip('where detail')
mode:
username:          admin
vdc:               switch
routing-context vrf: default
```

例 2:

```
>>> from cli import *
>>> cli("conf ; interface loopback 1")
''
>>> cli('where detail')
' mode:          \n username:          admin\n vdc:
  switch\n routing-context vrf: default\n'
>>>
```

例 3:

```
>>> r = cli('where detail')
>>> print(r)
mode:
username: admin
vdc: switch
routing-context vrf: default

>>>
```

例 4:

```
>>> from cli import *
>>> import json
>>> out=json.loads(clid('show version'))
>>> for k in out.keys():
...     print("%30s - %s" % (k,out[k]))
...
header_str - Cisco Nexus Operating System (NX-OS) Software
```

```

TAC support: http://www.cisco.com/tac
Copyright (C) 2002-2020, Cisco and/or its affiliates.
All rights reserved.
The copyrights to certain works contained in this software are
owned by other third parties and used and distributed under their own
licenses, such as open source. This software is provided "as is," and unless
otherwise stated, there is no warranty, express or implied, including but not
limited to warranties of merchantability and fitness for a particular purpose.
Certain components of this software are licensed under
the GNU General Public License (GPL) version 2.0 or
GNU General Public License (GPL) version 3.0 or the GNU
Lesser General Public License (LGPL) Version 2.1 or
Lesser General Public License (LGPL) Version 2.0.
A copy of each such license is available at
http://www.opensource.org/licenses/gpl-2.0.php and
http://opensource.org/licenses/gpl-3.0.html and
http://www.opensource.org/licenses/lgpl-2.1.php and
http://www.gnu.org/licenses/old-licenses/library.txt.
bios_ver_str - 07.67
kickstart_ver_str - 9.3(5) [build 9.3(4)IIL9(0.879)]
nxos_ver_str - 9.3(5) [build 9.3(4)IIL9(0.879)]
bios_cmpl_time - 01/29/2020
kick_file_name - bootflash:///nxos.9.3.4.IIL9.0.879.bin
nxos_file_name - bootflash:///nxos.9.3.4.IIL9.0.879.bin
kick_cmpl_time - 5/10/2020 21:00:00
nxos_cmpl_time - 5/10/2020 21:00:00
kick_tmstamp - 05/12/2020 07:08:44
nxos_tmstamp - 05/12/2020 07:08:44
chassis_id - Nexus9000 93180YC-EX chassis
cpu_name - Intel(R) Xeon(R) CPU @ 1.80GHz
memory - 24632252
mem_type - kB
proc_board_id - FDO22280FFK
host_name - switch
bootflash_size - 53298520
kern_uptm_days - 0
kern_uptm_hrs - 0
kern_uptm_mins - 19
kern_uptm_secs - 34
rr_usec - 641967
rr_ctime - Tue May 12 09:52:28 2020
rr_reason - Reset Requested by CLI command reload
rr_sys_ver - 9.4(1)
rr_service - None
plugins - Core Plugin, Ethernet Plugin
manufacturer - Cisco Systems, Inc.
>>>

```

非インタラクティブ Python

Python スクリプト名を引数として Python CLI 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、ブートフラッシュまたは揮発性スキームの下に配置する必要があります。Python CLI コマンドでは、Python スクリプトの最大 32 個のコマンドライン引数を使用できます。

Cisco Nexus 3500 プラットフォーム スイッチは、Python スクリプトを実行するための送信元 CLI コマンドもサポートしています。bootflash:scripts ディレクトリは、ソース CLI コマンドのデフォルトのスクリプト ディレクトリです。

この例では、最初にスクリプトを表示してから実行します。保存は、任意のファイルをブートフラッシュに持ってくるようなものです。

```
switch# show file bootflash:scripts/deltaCounters.py
#!/isan/bin/python3
from cli import *
import sys, time
ifName = sys.argv[1]
delay = float(sys.argv[2])
count = int(sys.argv[3])
cmd = 'show interface ' + ifName + ' counters'
out = json.loads(clid(cmd))
rxuc = int(out['TABLE_rx_counters']['ROW_rx_counters'][0]['eth_inucast'])
rxmc = int(out['TABLE_rx_counters']['ROW_rx_counters'][1]['eth_inmcast'])
rxbc = int(out['TABLE_rx_counters']['ROW_rx_counters'][1]['eth_inbcast'])
txuc = int(out['TABLE_tx_counters']['ROW_tx_counters'][0]['eth_outucast'])
txmc = int(out['TABLE_tx_counters']['ROW_tx_counters'][1]['eth_outmcast'])
txbc = int(out['TABLE_tx_counters']['ROW_tx_counters'][1]['eth_outbcast'])
print ('row rx_ucast rx_mcast rx_bcast tx_ucast tx_mcast tx_bcast')
print ('=====')
print (' %8d %8d %8d %8d %8d %8d' % (rxuc, rxmc, rxbc, txuc, txmc, txbc))
print ('=====')
i = 0
while (i < count):
    time.sleep(delay)
    out = json.loads(clid(cmd))
    rxucNew = int(out['TABLE_rx_counters']['ROW_rx_counters'][0]['eth_inucast'])
    rxmcNew = int(out['TABLE_rx_counters']['ROW_rx_counters'][1]['eth_inmcast'])
    rxbcNew = int(out['TABLE_rx_counters']['ROW_rx_counters'][1]['eth_inbcast'])
    txucNew = int(out['TABLE_tx_counters']['ROW_tx_counters'][0]['eth_outucast'])
    txmcNew = int(out['TABLE_tx_counters']['ROW_tx_counters'][1]['eth_outmcast'])
    txbcNew = int(out['TABLE_tx_counters']['ROW_tx_counters'][1]['eth_outbcast'])
    i += 1
    print ('%-3d %8d %8d %8d %8d %8d' % (i, rxucNew - rxuc, rxmcNew - rxmc, rxbcNew - rxbc, txucNew - txuc, txmcNew - txmc, txbcNew - txbc))

switch# python bootflash:scripts/deltaCounters.py mgmt0 1 5
row rx_ucast rx_mcast rx_bcast tx_ucast tx_mcast tx_bcast
=====
          291          8233          1767          185          57          2
=====
1           1           4           1           1           0           0
2           2           5           1           2           0           0
3           3           9           1           3           0           0
4           4          12           1           4           0           0
5           5          17           1           5           0           0
switch#
```

次の例は、**source** コマンドがコマンドライン引数を指定する方法を表示しています。この例では、**policy-map** は **cgrep python** スクリプトへの引数です。この例は、**source** コマンドがパイプ演算子（「|」）の後に続くことも表示しています。

```
switch# show running-config | source sys/cgrep policy-map

policy-map type network-qos nw-pfc
policy-map type network-qos no-drop-2
policy-map type network-qos wred-policy
policy-map type network-qos pause-policy
policy-map type qos foo
policy-map type qos classify
policy-map type qos cos-based
policy-map type qos no-drop-2
```



```
policy-map type qos pfc-tor-port
```

Embedded Event Manager でのスクリプトの実行

Cisco Nexus 3500 プラットフォーム スイッチ上の組み込みイベント 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。

次の例は、EEM アクションとして Python スクリプトを実行する方法を示しています。

- 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。

```
switch# show running-config eem

!Command: show running-config eem
!Running configuration last done at: Thu Jun 25 15:29:38 2020
!Time: Thu Jun 25 15:33:19 2020

version 9.3(5) Bios:version 07.67
event manager applet a1
  event cli match "show clock"
  action 1 cli python bootflash:pydate.py

switch# show file logflash:vdc_1/event_archive_1 | last 33

eem_event_time:06/25/2020,15:34:24 event_type:cli event_id:24 slot:active(1) vdc
:1 severity:minor applets:a1
eem_param_info:command = "exshow clock"
Starting with policy a1
stty: standard input: Inappropriate ioctl for device
Executing the following commands succeeded:
  python bootflash:pydate.py
Completed executing policy a1
Event Id:24 event type:10241 handling completed
```

- **show file logflash:event_archive_1** コマンドを実行して、ログファイル内のイベントによってトリガーされたアクションを検索できます。

```
switch# show file logflash:event_archive_1 | last 33

eem_event_time:05/01/2011,19:40:28 event_type:cli event_id:8 slot:active(1)
vdc:1 severity:minor applets:a1
eem_param_info:command = "exshow clock"
Starting with policy a1
Python

2011-05-01 19:40:28.644891
Executing the following commands succeeded:
  python bootflash:pydate.py

PC_VSH_CMD_TLV(7679) with q
```

Cisco NX-OS ネットワーク インターフェイスとの Python 統合

Cisco Nexus 3500 プラットフォーム スイッチでは、Python が基盤となる Cisco NX-OS ネットワーク インターフェイスと統合されています。cisco.vrf.set_global_vrf () API を介してコンテキストを設定することにより、ある仮想ルーティング コンテキストから別の仮想ルーティング コンテキストに切り替えることができます。

次の例は、デバイスの管理インターフェイスを介して HTML ドキュメントを取得する方法を示しています。目的の仮想ルーティング コンテキストに切り替えることにより、帯域内インターフェイスを介して外部エンティティへの接続を確立することもできます。

```
switch# python
```

```
Warning: Python 2.7 is End of Support, and future NXOS software will deprecate
python 2.7 support. It is recommended for new scripts to use 'python3' instead.
Type "python3" to use the new shell.
```

```
Python 2.7.11 (default, Jun  4 2020, 09:48:24)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import urllib2
>>> from cisco.vrf import *
>>> set_global_vrf('management')
>>> page=urllib2.urlopen('http://172.23.40.211:8000/welcome.html')
>>> print page.read()
Hello Cisco Nexus 9000
>>>
>>> import cisco
>>> help(cisco.vrf.set_global_vrf)
Help on function set_global_vrf in module cisco.vrf:
set_global_vrf(vrf)
Sets the global vrf. Any new sockets that are created (using socket.socket)
will automatically get set to this vrf (including sockets used by other
python libraries).
Arguments:
vrf: VRF name (string) or the VRF ID (int).
Returns: Nothing
>>>
```

Python による Cisco NX-OS セキュリティ

Cisco NX-OS 情報技術は、ソフトウェアの Cisco NX-OS サンドボックス レイヤおよび CLI ロールベース アクセス コントロール (RBAC) によって保護されます。

Cisco NX-OS network-admin または dev-ops ロールに関連付けられているすべてのユーザは、特権ユーザです。カスタム ロールで Python へのアクセスが許可されているユーザは、非特権ユーザと見なされます。非特権ユーザは、ファイル システム、ゲスト シェル、Bash コマンドなどの Cisco NX-OS 情報技術へのアクセスが制限されています。特権ユーザは、Cisco NX-OS のすべての情報技術へのアクセスが向上します。

セキュリティとユーザー権限の例

•

Scheduler でのスクリプトの実行例

.



第 6 章

Python API

- [\[Python API について \(About the Python API\)\]](#) (81 ページ)
- [Python の使用](#) (82 ページ)

[Python API について (About the Python API)]

Pythonは簡単に習得できる強力なプログラミング言語です。効率的で高水準なデータ構造を持ち、オブジェクト指向プログラミングに対してシンプルで効果的なアプローチを取っています。Pythonは、簡潔な構文、動的な型指定、インタープリタ型という特長を持っており、ほとんどのプラットフォームのさまざまな分野でスクリプティングと高速アプリケーション開発が可能な理想的な言語です。

Python インタープリタと広範な標準規格ライブラリが Python Web サイトで送信元形式またはバイナリ形式で自由に利用できます：

<http://www.python.org/>

また、このサイトには、サードパーティが無償で提供している多数の Python モジュール、プログラム、ツールのディストリビューションとそれらへのリンク、さらに追加のドキュメンテーションが掲載されています。

Cisco Nexus 30009000 Series デバイスは、インタラクティブ モードと非インタラクティブ（スクリプト）モードの両方で Python v2.7.5 をサポートし、ゲスト シェルで使用できます。

Python スクリプト機能は、さまざまなタスクを実行するためにデバイスのコマンドライン インターフェイス（CLI）Power On Auto Provisioning（POAP）または Embedded Event Manager（EEM）アクションへのプログラムによるアクセスを提供します。Python は Bash シェルからもアクセスできます。

Python インタプリタは Cisco NX-OS ソフトウェアで利用できます。

Cisco Nexus デバイスでの Python の使用については、<https://developer.cisco.com/site/nx-os/docs/apis/python/> の URL にある Cisco Nexus 9000 Series Python SDK User Guide and API Reference を参照してください。[この情報は、Cisco Nexus 3000 シリーズ デバイスにも適用されます。（This information applies also to the Cisco Nexus 3000 Series devices.）]

Python の使用

ここでは、Python スクリプトの作成と実行の方法について説明します。

Cisco Python パッケージ

Cisco NX-OS は、インターフェイス、VLAN、VRF、ACL、ルートなど、多くのコア ネットワーク デバイス モジュールへのアクセスを可能にする Cisco Python パッケージを提供します。**help()** コマンドを入力すると、Cisco Python パッケージの詳細を表示できます。モジュール内のクラスとメソッドに関する追加情報を取得するには、特定のモジュールに対して **help** コマンドを実行します。たとえば、**help** (*cisco.interface*) は、*cisco.interface* モジュールのプロパティを表示します。

その他の Python モジュールについては、https://devhub.cisco.com/artifactory/open-nxos/9.2.1/x86_64/ から *python-modules-nxos RPM* (*python-modules-nxos-1.0.0-9.2.1.lib32_x86.rpm*) をインストールできます。RPM のインストール手順については、「機能 RPM の管理」セクションを参照してください。

次の例は、Cisco Python パッケージに関する情報を表示する方法を示します。

```
>>> import cisco
>>> help(cisco)
Help on package cisco:

NAME
    cisco

FILE
    /isan/python/scripts/cisco/__init__.py

PACKAGE CONTENTS
    acl
    bgp
    cisco_secret
    cisco_socket
    feature
    interface
    key
    line_parser
    md5sum
    nxcli
    ospf
    routemap
    routes
    section_parser
    ssh
    system
    tacacs
    vrf

CLASSES
    __builtin__.object
    cisco.cisco_secret.CiscoSecret
    cisco.interface.Interface
    cisco.key.Key
```

次に、Python 3 用の Cisco Python パッケージに関する情報を表示する方法の例を示します。

```
switch# python3
Python 3.7.3 (default, Nov 20 2019, 14:38:01)
[GCC 5.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import cisco
>>> help(cisco)
Help on package cisco:

NAME
cisco

PACKAGE CONTENTS
acl
bgp
buffer_depth_monitor
check_port_discards
cisco_secret
feature
historys
interface
ipaddress
key
line_parser
mac_address_table
md5sum
nxcli
nxos_cli
ospf
routemap
routes
section_parser
ssh
system
tacacs
transfer
vlan
vrf

CLASSES
builtins.dict(builtins.object)
cisco.history.History
builtins.object
cisco.cisco_secret.CiscoSecret
cisco.interface.Interface
cisco.key.Key
```

CLI コマンド API の使用

Python プログラミング言語は、CLI コマンドを実行できる 3 つの API を使用します。API は Python CLI モジュールから利用できます。

これらの API については、次の表で説明します。*** from cli import** コマンドを使用して API を有効にする必要があります。これらの API 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 API 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。

表 6: CLI コマンド API

API	説明
cli() 例: <pre>string = cli ("cli-command")</pre>	制御文字または特殊文字を含む CLI コマンドの未処理の出力を返します。 (注) インタラクティブな Python インタープリタは、制御文字または特殊文字を「エスケープ」して出力します。改行は「\n」として出力され、結果が読みにくい場合があります。 cli() API は、判読性が高い結果を出力します。
clid() 例: <pre>json_string = clid ("cli-command")</pre>	コマンドに XML サポートが存在する場合は、 cli-command の JSON 出力を返します。それ以外の場合は、例外がスローされます。 (注) この API は、 show コマンドの出力の検索時に使用すると便利な場合があります。
clip() 例: <pre>clip ("cli-command")</pre>	CLI コマンドの出力を直接 stdout に出力し、Python には何も返されません。 (注) <pre>clip ("cli-command")</pre> と同等です (is equivalent to) <pre>r=cli("cli-command") print r</pre>

2 つ以上のコマンドを個別に実行すると、その状態は 1 つのコマンドから後続のコマンドまで持続しません。

次の例では、最初のコマンドの状態が 2 番目のコマンドで持続しないため、2 番目のコマンドが失敗します。

```
>>> cli("conf t")
>>> cli("interface eth4/1")
```

2 つ以上のコマンドを同時に実行すると、その状態は 1 つのコマンドから後続のコマンドまで持続します。

次の例では、2 番目と 3 番目のコマンドの状態が持続するため、2 番目のコマンドは成功しています。

```
>>> cli("conf t ; interface eth4/1 ; shut")
```




(注) 例に示すように、コマンドは「;」で区切られます。セミコロン(;)は、単一のブランク文字で囲む必要があります。

CLI からの Python インタープリタの呼び出し

次に、CLI から Python 2 を呼び出す方法を表示します:



(注) Python インタープリタのプロンプトは「>>>」または「...」で表示されます。



重要 Python 2.7 のサポートは終了し、将来の NX-OS ソフトウェアは Python 2.7 のサポートを廃止します。新しいスクリプトでは、代わりに **python3** を使用することをお勧めします。新しいシェルを使用するように **python3** を入力します。

次に、CLI から Python 3 を呼び出す方法を表示します:

```
switch# python3
Python 3.7.3 (default, Nov 20 2019, 14:38:01)
[GCC 5.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> from cli import *
>>> import json
>>> cli('configure terminal ; interface loopback 1 ; no shut')
''
>>> intflist=json.loads(clid('show interface brief'))
>>> i=0
>>> while i < len(intflist['TABLE_interface']['ROW_interface']):
...     intf=intflist['TABLE_interface']['ROW_interface'][i]
...     i=i+1
...     if intf['state'] == 'up':
...         print(intf['interface'])
...
mgmt0
loopback1
>>>
```

表示フォーマット

次に、Python API を使用したさまざまな表示フォーマットを示します:

例 1:

```
>>> from cli import *
>>> cli("conf ; interface loopback 1")
''
>>> clip('where detail')
mode:
```

```
username:          admin
vdc:               switch
routing-context vrf: default
```

例 2:

```
>>> from cli import *
>>> cli("conf ; interface loopback 1")
''
>>> cli('where detail')
' mode:          \n username:          admin\n vdc:
  switch\n routing-context vrf: default\n'
>>>
```

例 3:

```
>>> r = cli('where detail')
>>> print(r)
mode:
username: admin
vdc: switch
routing-context vrf: default
>>>
```

例 4:

```
>>> from cli import *
>>> import json
>>> out=json.loads(clid('show version'))
>>> for k in out.keys():
...   print("%30s - %s" % (k,out[k]))
...
header_str - Cisco Nexus Operating System (NX-OS) Software
TAC support: http://www.cisco.com/tac
Copyright (C) 2002-2020, Cisco and/or its affiliates.
All rights reserved.
The copyrights to certain works contained in this software are
owned by other third parties and used and distributed under their own
licenses, such as open source. This software is provided "as is," and unless
otherwise stated, there is no warranty, express or implied, including but not
limited to warranties of merchantability and fitness for a particular purpose.
Certain components of this software are licensed under
the GNU General Public License (GPL) version 2.0 or
GNU General Public License (GPL) version 3.0 or the GNU
Lesser General Public License (LGPL) Version 2.1 or
Lesser General Public License (LGPL) Version 2.0.
A copy of each such license is available at
http://www.opensource.org/licenses/gpl-2.0.php and
http://opensource.org/licenses/gpl-3.0.html and
http://www.opensource.org/licenses/lgpl-2.1.php and
http://www.gnu.org/licenses/old-licenses/library.txt.
bios_ver_str - 07.67
kickstart_ver_str - 9.3(5) [build 9.3(4)IIL9(0.879)]
nxos_ver_str - 9.3(5) [build 9.3(4)IIL9(0.879)]
bios_cmpl_time - 01/29/2020
kick_file_name - bootflash:///nxos.9.3.4.IIL9.0.879.bin
nxos_file_name - bootflash:///nxos.9.3.4.IIL9.0.879.bin
kick_cmpl_time - 5/10/2020 21:00:00
nxos_cmpl_time - 5/10/2020 21:00:00
kick_tmstamp - 05/12/2020 07:08:44
nxos_tmstamp - 05/12/2020 07:08:44
```

```

chassis_id - Nexus9000 93180YC-EX chassis
cpu_name - Intel(R) Xeon(R) CPU @ 1.80GHz
memory - 24632252
mem_type - kB
proc_board_id - FDO22280FFK
host_name - switch
bootflash_size - 53298520
kern_uptm_days - 0
kern_uptm_hrs - 0
kern_uptm_mins - 19
kern_uptm_secs - 34
rr_usecs - 641967
rr_ctime - Tue May 12 09:52:28 2020
rr_reason - Reset Requested by CLI command reload
rr_sys_ver - 9.4(1)
rr_service - None
plugins - Core Plugin, Ethernet Plugin
manufacturer - Cisco Systems, Inc.
>>>

```

非インタラクティブ Python

Python スクリプト名を引数として Python CLI 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、ブートフラッシュまたは揮発性スキームの下に配置する必要があります。Python CLI コマンドでは、Python スクリプトの最大 32 個のコマンドライン引数を使用できます。

スイッチは、Python スクリプトを実行するためのソース CLI コマンドもサポートしています。bootflash:scripts ディレクトリは、ソース CLI コマンドのデフォルトのスクリプトディレクトリです。

この例では、最初にスクリプトを表示してから実行します。保存は、任意のファイルをブートフラッシュに持ってくるようなものです。

```

switch# show file bootflash:scripts/deltaCounters.py
#!/isan/bin/python3
from cli import *
import sys, time
ifName = sys.argv[1]
delay = float(sys.argv[2])
count = int(sys.argv[3])
cmd = 'show interface ' + ifName + ' counters'
out = json.loads(clid(cmd))
rxuc = int(out['TABLE_rx_counters']['ROW_rx_counters'][0]['eth_inucast'])
rxmc = int(out['TABLE_rx_counters']['ROW_rx_counters'][1]['eth_inmcast'])
rxbc = int(out['TABLE_rx_counters']['ROW_rx_counters'][1]['eth_inbcast'])
txuc = int(out['TABLE_tx_counters']['ROW_tx_counters'][0]['eth_outucast'])
txmc = int(out['TABLE_tx_counters']['ROW_tx_counters'][1]['eth_outmcast'])
txbc = int(out['TABLE_tx_counters']['ROW_tx_counters'][1]['eth_outbcast'])
print ('row rx_ucast rx_mcast rx_bcast tx_ucast tx_mcast tx_bcast')
print ('=====')
print (' %8d %8d %8d %8d %8d %8d' % (rxuc, rxmc, rxbc, txuc, txmc, txbc))
print ('=====')
i = 0
while (i < count):
    time.sleep(delay)
    out = json.loads(clid(cmd))
    rxucNew = int(out['TABLE_rx_counters']['ROW_rx_counters'][0]['eth_inucast'])
    rxmcNew = int(out['TABLE_rx_counters']['ROW_rx_counters'][1]['eth_inmcast'])

```

```

rxbcNew = int(out['TABLE_rx_counters']['ROW_rx_counters'][1]['eth_inbcst'])
txucNew = int(out['TABLE_tx_counters']['ROW_tx_counters'][0]['eth_outucast'])
txmcNew = int(out['TABLE_tx_counters']['ROW_tx_counters'][1]['eth_outmcast'])
txbcNew = int(out['TABLE_tx_counters']['ROW_tx_counters'][1]['eth_outbcst'])
i += 1
print ('%-3d %8d %8d %8d %8d %8d %8d' % (i, rxucNew - rxuc, rxmcNew - rxmc, rxbcNew -
rxbc, txucNew - txuc, txmcNew - txmc, txbcNew - txbc))

switch# python bootflash:scripts/deltaCounters.py mgmt0 1 5
row rx_ucast rx_mcast rx_bcast tx_ucast tx_mcast tx_bcast
=====
          291          8233          1767          185          57          2
=====
1           1           4           1           1           0           0
2           2           5           1           2           0           0
3           3           9           1           3           0           0
4           4          12           1           4           0           0
5           5          17           1           5           0           0
switch#

```

次の例は、送信元コマンドでコマンドライン引数を指定する方法を表示しています。この例では、**policy-map** は **cgrep python** スクリプトへの引数です。この例は、送信元コマンドがパイプ演算子 (「|」) の後に続くことも示しています。

```

switch# show running-config | source sys/cgrep policy-map

policy-map type network-qos nw-pfc
policy-map type network-qos no-drop-2
policy-map type network-qos wred-policy
policy-map type network-qos pause-policy
policy-map type qos foo
policy-map type qos classify
policy-map type qos cos-based
policy-map type qos no-drop-2
policy-map type qos pfc-tor-port

```

Embedded Event Manager でのスクリプトの実行

Embedded Event Manager (EEM) のポリシーは、Python スクリプトをサポートします。

次の例は、EEM アクションとして Python スクリプトを実行する方法を示しています。

- 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。

```

switch# show running-config eem

!Command: show running-config eem
!Running configuration last done at: Thu Jun 25 15:29:38 2020
!Time: Thu Jun 25 15:33:19 2020

version 9.3(5) Bios:version 07.67
event manager applet a1
  event cli match "show clock"
  action 1 cli python bootflash:pydate.py

switch# show file logflash:vdc_1/event_archive_1 | last 33

eem_event_time:06/25/2020,15:34:24 event_type:cli event_id:24 slot:active(1) vdc

```

```
:1 severity:minor applets:a1
eem_param_info:command = "exshow clock"
Starting with policy a1
stty: standard input: Inappropriate ioctl for device
Executing the following commands succeeded:
    python bootflash:pydate.py
Completed executing policy a1
Event Id:24 event type:10241 handling completed
```

- **show file logflash:event_archive_1** コマンドを実行して、ログ ファイル内のイベントによってトリガーされたアクションを検索できます。

```
switch# show file logflash:event_archive_1 | last 33

eem_event_time:05/01/2011,19:40:28 event_type:cli event_id:8 slot:active(1)
vdc:1 severity:minor applets:a1
eem_param_info:command = "exshow clock"
Starting with policy a1
Python

2011-05-01 19:40:28.644891
Executing the following commands succeeded:
    python bootflash:pydate.py

PC_VSH_CMD_TLV(7679) with q
```

Cisco NX-OS ネットワーク インターフェイスとの Python 統合

Cisco Nexus スイッチでは、Python が基盤となる Cisco NX-OS ネットワーク インターフェイスと統合されています。cisco.vrf.set_global_vrf() API を介してコンテキストを設定することにより、ある仮想ルーティング コンテキストから別の仮想ルーティング コンテキストに切り替えることができます。

次の例は、デバイスの管理インターフェイスを介して HTML ドキュメントを取得する方法を示しています。目的の仮想ルーティング コンテキストに切り替えることにより、帯域内インターフェイスを介して外部エンティティへの接続を確立することもできます。

```
switch# python

Warning: Python 2.7 is End of Support, and future NXOS software will deprecate
python 2.7 support. It is recommended for new scripts to use 'python3' instead.
Type "python3" to use the new shell.

Python 2.7.11 (default, Jun  4 2020, 09:48:24)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import urllib2
>>> from cisco.vrf import *
>>> set_global_vrf('management')
>>> page=urllib2.urlopen('http://172.23.40.211:8000/welcome.html')
>>> print page.read()
Hello Cisco Nexus 9000
>>>
>>> import cisco
>>> help(cisco.vrf.set_global_vrf)
Help on function set_global_vrf in module cisco.vrf:
set_global_vrf(vrf)
```

```

Sets the global vrf. Any new sockets that are created (using socket.socket)
will automatically get set to this vrf (including sockets used by other
python libraries).
Arguments:
vrf: VRF name (string) or the VRF ID (int).
Returns: Nothing
>>>

```

Python による Cisco NX-OS セキュリティ

Cisco NX-OS 情報技術は、ソフトウェアの Cisco NX-OS サンドボックス レイヤおよび CLI ロールベース アクセス コントロール (RBAC) によって保護されます。

Cisco NX-OS network-admin または dev-ops ロールに関連付けられているすべてのユーザは、特権ユーザです。カスタム ロールで Python へのアクセスが許可されているユーザーは、非特権ユーザーと見なされます。非特権ユーザは、ファイル システム、ゲスト シェル、Bash コマンドなどの Cisco NX-OS 情報技術へのアクセスが制限されています。特権ユーザは、Cisco NX-OS のすべての情報技術へのアクセスが向上します。

セキュリティとユーザー権限の例

次の例は、特権ユーザーがコマンドを実行する方法を示しています：

Python 3 の例。

```

switch# python3
Python 3.7.3 (default, Nov 20 2019, 14:38:01)
[GCC 5.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import os
>>> os.system('whoami')
admin
0
>>> f=open('/tmp/test','w')
>>> f.write('hello from python')
17
>>> f.close()
>>> r=open('/tmp/test','r')
>>> print(r.read())
hello from python
>>> r.close()
>>>

```

次の例は、アクセスを拒否されている非特権ユーザーを示しています：

```

switch# python3
Python 3.7.3 (default, Nov 20 2019, 14:38:01)
[GCC 5.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import os
>>> os.system('whoami')
system(whoami): rejected!
-1
>>> f=open('/tmp/test','w')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
PermissionError: [Errno 13] Permission denied: '/tmp/test'

```

```
>>>
```

RBAC は、ログイン ユーザー権限に基づいて CLI アクセスを制御します。ログイン ユーザーの ID は、CLI シェルまたは **Bash** から呼び出される Python に与えられます。Python は、Python から呼び出されたサブプロセスにログイン ユーザーの ID を渡します。

以下は、特権ユーザーの例です:

```
>>> from cli import *
>>> cli('show clock')
'Warning: No NTP peer/server configured. Time may be out of sync.\n15:39:39.513 UTC Thu Jun
 25 2020\nTime source is NTP\n'
>>> cli('configure terminal ; vrf context myvrf')
''
>>> clip('show running-config l3vm')

!Command: show running-config l3vm
!Running configuration last done at: Thu Jun 25 15:39:49 2020
!Time: Thu Jun 25 15:39:55 2020

version 9.3(5) Bios:version 07.67

interface mgmt0
    vrf member management
vrf context blue
vrf context management
vrf context myvrf
```

以下は、非特権ユーザーの例です:

```
>>> from cli import *
>>> cli('show clock')
'11:18:47.482 AM UTC Sun May 08 2011\n'
>>> cli('configure terminal ; vrf context myvrf2')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/isan/python/scripts/cli.py", line 20, in cli
    raise cmd_exec_error(msg)
errors.cmd_exec_error: '% Permission denied for the role\n\nCmd exec error.\n'
```

次の例は、RBAC 構成を示しています:

```
switch# show user-account
user:admin
    this user account has no expiry date
    roles:network-admin
user:pyuser
    this user account has no expiry date
    roles:network-operator python-role
switch# show role name python-role
```

スケジューラでスクリプトを実行する例

次の例は、スケジューラ機能を使用してスクリプトを実行する Python スクリプトを示しています。

```
#!/bin/env python
from cli import *
from nxos import *
```

```
import os

switchname = cli("show switchname")
try:
    user = os.environ['USER']
except:
    user = "No user"
    pass

msg = user + " ran " + __file__ + " on : " + switchname
print msg
py_syslog(1, msg)
# Save this script in bootflash:///scripts
```

Python 3 の例。

```
#!/bin/env python3
from cli import *
from nxos import *
import os

switchname = cli("show switchname")
try:
    user = os.environ['USER']
except:
    user = "No user"
    pass

msg = user + " ran " + __file__ + " on : " + switchname
print(msg)
py_syslog(1, msg)

# Save this script in bootflash:///scripts

switch# conf t
Enter configuration commands, one per line. End with CNTL/Z.
switch(config)# feature scheduler
switch(config)# scheduler job name testplan
switch(config-job)# python bootflash:///scripts/test.py
switch(config-job)# exit
switch(config)# scheduler schedule name testplan
switch(config-schedule)# job name testplan
switch(config-schedule)# time start now repeat 0:0:4
Schedule starts from Sat Jun 13 04:29:38 2020
switch# 2020 Jun 13 04:29:41 switch %USER-1-SYSTEM_MSG: No user ran /bootflash/scripts/test.py
  on : switch - nxpython
switch# show scheduler schedule
Schedule Name : testplan
-----
User Name : admin
Schedule Type : Run every 0 Days 0 Hrs 4 Mins
Start Time : Sat Jun 13 04:29:38 2020
Last Execution Time : Sat Jun 13 04:29:38 2020
Last Completion Time: Sat Jun 13 04:29:41 2020
Execution count : 1
-----
Job Name Last Execution Status
-----
testplan Success (0)
=====
switch#

switch# conf t
Enter configuration commands, one per line. End with CNTL/Z.
```



```

switch(config)# feature scheduler
switch(config)# scheduler job name testplan
switch(config-job)# python bootflash:///scripts/testplan.py
switch(config-job)# exit
switch(config)# scheduler schedule name testplan
switch(config-schedule)# job name testplan
switch(config-schedule)# time start now repeat 0:0:4
Schedule starts from Mon Mar 14 16:40:03 2011
switch(config-schedule)# end
switch# term mon
2011 Mar 14 16:38:03 switch %VSHD-5-VSHD_SYSLOG_CONFIG_I: Configured from vty by admin on
10.19.68.246@pts/2
switch# show scheduler schedule
Schedule Name          : testplan
-----
User Name              : admin
Schedule Type          : Run every 0 Days 0 Hrs 4 Mins
Start Time             : Mon Mar 14 16:40:03 2011
Last Execution Time    : Yet to be executed
-----
      Job Name          Last Execution Status
-----
      testplan          -NA-
=====
switch#
switch# 2011 Mar 14 16:40:04 switch  %USER-1-SYSTEM_MSG: No user ran
/bootflash/scripts/testplan.py on : switch  - nxpython
2011 Mar 14 16:44:04 switch last message repeated 1 time
switch#

```




第 7 章

tcl によるスクリプティング

- [Tcl について（95 ページ）](#)
- [Tclsh コマンドの実行（98 ページ）](#)
- [Tclsh コマンドからの Cisco NX-OS モード間の移動（99 ページ）](#)
- [tcl の参照（100 ページ）](#)

Tcl について

Tcl（「ティックル」と発音）は、CLI コマンドの柔軟性を高めるスクリプト言語です。Tcl を使用して **show** コマンドの出力の特定の値を抽出したり、スイッチを設定したり、Cisco NX-OS コマンドをループで実行したり、スクリプトで Embedded Event Manager（EEM）ポリシーを定義したりすることができます。

このセクションでは、Tcl スクリプトを実行する方法、またはスイッチで Tcl を対話的に実行する方法について説明します。

tclsh コマンドのヘルプ

Tcl コマンドでは、コマンドのヘルプは使用できません。インタラクティブ tcl シェル内から Cisco NX-OS コマンドのヘルプ機能に引き続きアクセスできます。

次に、インタラクティブ Tcl シェルで Tcl コマンドのヘルプがない場合の例を示します。

```
switch# tclsh
switch-tcl# set x 1
switch-tcl# puts ?
      ^
% Invalid command at '^' marker.
switch-tcl# configure ?
<CR>
  session  Configure the system in a session
  terminal Configure the system from terminal input

switch-tcl#
```



- (注) 上の例では、Cisco NX-OS コマンドのヘルプ機能が引き続き使用できますが、Tcl の **puts** コマンドはヘルプ機能からのエラーを返します。

tclsh コマンドの履歴

端末で矢印キーを使用して、以前にインタラクティブ Tcl シェルで入力したコマンドにアクセスできます。



- (注) インタラクティブ Tcl シェルを終了すると、**tclsh** コマンドの履歴は保存されません。

tclsh のタブ補完

インタラクティブ Tcl シェルを実行している場合は、Cisco NX-OS コマンドのタブ補完を使用できます。Tcl コマンドでは、タブ補完は使用できません。

tclsh の CLI コマンド

インタラクティブ tcl シェル内から直接 Cisco NX-OS コマンドにアクセスできますが、Tcl **cli** コマンドにより付加される場合のみ tcl スクリプト内で Cisco NX-OS コマンドを実行できます。

インタラクティブ Tcl シェルでは、次のコマンドは同じであり、正しく実行されます：

```
switch-tcl# cli show module 1 | incl Mod
switch-tcl# cli "show module 1 | incl Mod"
switch-tcl# show module 1 | incl Mod
```

Tcl スクリプトで、次の例のように、Cisco NX-OS コマンドに Tcl **cli** コマンドを付加する必要があります：

```
set x 1
cli show module $x | incl Mod
cli "show module $x | incl Mod"
```

スクリプトで次のコマンドを使用すると、そのスクリプトは機能不全になり、Tcl シェルにエラーが表示されます：

```
show module $x | incl Mod
"show module $x | incl Mod"
```

tclsh コマンドの区切り

セミコロン (;) は、Cisco NX-OS と Tcl の両方でのコマンド区切りです。Tcl コマンドで複数の Cisco NX-OS コマンドを実行するには、各 Cisco NX-OS コマンドを引用符 (") で囲む必要があります。

双方向性 Tcl シェルでは、次のコマンドは同じであり、正しく実行されます。

```
switch-tcl# cli "configure terminal ; interface loopback 10 ; description loop10"
switch-tcl# cli configure terminal ; cli interface loopback 10 ; cli description loop10
switch-tcl# cli configure terminal
Enter configuration commands, one per line. End with CNTL/Z.

switch(config-tcl)# cli interface loopback 10
switch(config-if-tcl)# cli description loop10
switch(config-if-tcl)#
```

双方向性 Tcl シェルでは、Tcl cli コマンドを付加せずに、直接 Cisco NX-OS コマンドを実行することもできます。

```
switch-tcl# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.

switch(config-tcl)# interface loopback 10
switch(config-if-tcl)# description loop10
switch(config-if-tcl)#
```

tcl 変数

Tcl 変数を Cisco NX-OS コマンドへの引数として使用できます。また、Tcl スクリプトに引数を渡すこともできます。tcl 変数は永続的ではありません。

次の例は、Cisco NX-OS コマンドの引数として Tcl 変数を使用する方法を表示しています。

```
switch# tclsh
switch-tcl# set x loop10
switch-tcl# cli "configure terminal ; interface loopback 10 ; description $x"
switch(config-if-tcl)#
```

tclquit

tclquit コマンドは、どの Cisco NX-OS コマンドモードが現在現用系であるかには関係なく Tcl シェルを終了します。また、Ctrl+C を押して Tcl シェルを終了することもできます。exit と endCisco NX-OS コマンドは、コマンドモードを変更します。exit コマンドは、EXEC コマンドモードからのみ Tcl シェルを終了します。

Tclsh セキュリティ

tcl シェルは、Cisco NX-OS システムの特定の部分への不正アクセスを防止するために、サンドボックスで実行されます。システムは、無限ループや過剰なメモリ使用率などのイベントを検出する

ために、tcl シェルによって使用されている CPU、メモリ、ファイルなどのシステムリソースをモニタリングします。

初期の tcl 環境は、**scripting tcl init init-file** コマンドで設定します。

scripting tcl recursion-limit iterations コマンドを使用して、tcl 環境のループ制限を定義できます。デフォルトの再帰制限は 1000 回の繰り返しです。

Tclsh コマンドの実行

tclsh コマンドを使用すると、スクリプトまたはコマンドラインから tcl コマンドを実行できます。



(注) CLI プロンプトの状態では **tcl** スクリプト ファイルを作成することはできません。スクリプト ファイルをリモート デバイスで作成して、Cisco NX-OS デバイスの **bootflash:** ディレクトリにコピーすることができます。

手順の概要

1. **tclsh [bootflash:filename [argument ...]]**

手順の詳細

手順

	コマンドまたはアクション	目的
Step 1	tclsh [bootflash:filename [argument ...]] 例: <pre>switch# tclsh ? <CR> bootflash: The file to run</pre>	tcl シェルを開始します。 引数を指定せずに tclsh コマンドを実行すると、シェルは対話形式で実行され、標準入力から tcl コマンドを読み込んで、コマンドの結果とエラー メッセージを標準出力に出力します。 tclquit を入力するか、 Ctrl-C を押すとインタラクティブ tcl シェルが終了します。 引数を指定して tclsh コマンドを実行すると、最初の引数は、tcl コマンドが記述されたスクリプト ファイルの名前になり、他の引数をスクリプトで変数として使用できます。

例

次の例は、インタラクティブな Tcl シェルを示しています。

```

switch# tclsh
switch-tcl# set x 1
switch-tcl# cli show module $x | incl Mod
Mod  Ports  Module-Type          Model          Status
1    36      36p 40G Ethernet Module  N9k-X9636PQ   ok
Mod  Sw      Hw
Mod  MAC-Address(es)      Serial-Num

switch-tcl# exit
switch#

```

次に、Tcl スクリプトを実行する方法の例を示します。

```

switch# show file bootflash:showmodule.tcl
set x 1
while {$x < 19} {
  cli show module $x | incl Mod
  set x [expr {$x + 1}]
}

switch# tclsh bootflash:showmodule.tcl
Mod  Ports  Module-Type          Model          Status
1    36      36p 40G Ethernet Module  N9k-X9636PQ   ok
Mod  Sw      Hw
Mod  MAC-Address(es)      Serial-Num

switch#

```

Tclsh コマンドからの Cisco NX-OS モード間の移動

インタラクティブ Tcl シェルの実行中に Cisco NX-OS のモードを変更できます。

手順の概要

1. **tclsh**
2. **configure terminal**
3. **tclquit**

手順の詳細

手順

	コマンドまたはアクション	目的
Step 1	tclsh 例: <pre>switch# tclsh switch-tcl#</pre>	インタラクティブ Tcl シェルを開始します。

	コマンドまたはアクション	目的
Step 2	configure terminal 例: <pre>switch-tcl# configure terminal switch(config-tcl)#</pre>	Tcl シェルで Cisco NX-OS のコマンドを実行して、モードを変更します。 (注) Tcl プロンプトが変化して、Cisco NX-OS コマンドモードになったことが示されます。
Step 3	tclquit 例: <pre>switch-tcl# tclquit switch#</pre>	Tcl シェルを終了し、始めのモードに戻ります。

例

次の例は、対話型 Tcl シェルから Cisco NX-OS モードを変更する方法を示しています。

```
switch# tclsh
switch-tcl# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
switch(config-tcl)# interface loopback 10
switch(config-if-tcl)# ?
  description  Enter description of maximum 80 characters
  inherit      Inherit a port-profile
  ip           Configure IP features
  ipv6         Configure IPv6 features
  logging      Configure logging for interface
  no           Negate a command or set its defaults
  rate-limit   Set packet per second rate limit
  shutdown     Enable/disable an interface
  this         Shows info about current object (mode's instance)
  vrf          Configure VRF parameters
  end          Go to exec mode
  exit         Exit from command interpreter
  pop          Pop mode from stack or restore from name
  push         Push current mode to stack or save it under name
  where        Shows the cli context you are in

switch(config-if-tcl)# description loop10
switch(config-if-tcl)# tclquit
Exiting Tcl
switch#
```

tcl の参照

次のタイトルは、参照のために示されています。

- Mark Harrison (ed)、『Tcl/Tk Tools』、O'Reilly Media、ISBN 1-56592-218-2、1997 年

- Mark Harrison および Michael McLennan、『*Effective Tcl/Tk Programming*』、Addison-Wesley、Reading、MA、USA、ISBN 0-201-63474-0、1998 年
- Brent B. Ousterhout、『*Tcl and the Tk Toolkit*』、Addison-Wesley、Reading、MA、USA、ISBN 0-201-63337-X、1994 年
- Brent B. Welch、『*Practical Programming in Tcl and Tk*』、Prentice Hall、Upper Saddle River、NJ、USA、ISBN 0-13-038560-3、2003 年
- J Adrian Zimmer、『*Tcl/Tk for Programmers*』、IEEE Computer Society、John Wiley and Sons により出版、ISBN 0-8186-8515-8、1998 年



第 8 章

Ansible

- 前提条件（103 ページ）
- アンシブルについて（103 ページ）
- Cisco Ansible モジュール（104 ページ）

前提条件

サポートされている制御環境のインストール要件については、https://docs.ansible.com/ansible/latest/getting_started/index.html を参照してください。

アンシブルについて

Ansible は、クラウドプロビジョニング、構成管理、アプリケーションの展開、サービス内オーケストレーション、およびその他の IT ニーズを自動化するオープンソースの IT 自動化エンジンです。

Ansible は、Ansible モジュールと呼ばれる小さなプログラムを使用してノードへの API 呼び出しを行い、Playbook で定義された構成を適用します。

デフォルトでは、Ansible は、すべての管理対象マシンを独自に選択したグループに入れる単純な INI ファイルを使用して、管理対象のマシンを表します。

詳細については、Ansible から入手できます。

Ansible	https://www.ansible.com/
Ansible 自動化ソリューションインストール手順、プレイブックの手順と例、モジュールリストなどが含まれます。	https://docs.ansible.com/

Cisco Ansible モジュール

次のリンクの表に示すように、Ansible には複数の Cisco NX-OS でサポートされるモジュールとプレイブックがあります。

NX-OS デベロッパーのランディング ページ。	構成管理ツール
Ansible NX-OS プレイブックの例	Ansible nxos Playbook のリポジトリ
Ansible NX-OS ネットワーク モジュール	nxos ネットワーク モジュール



第 9 章

Puppet Agent

この章は、次の項で構成されています。

- [Puppet について（105 ページ）](#)
- [前提条件（106 ページ）](#)
- [Puppet エージェント NX-OS 環境（106 ページ）](#)
- [ciscopuppet モジュール（106 ページ）](#)

Puppet について

Puppet Labs によって開発された Puppet ソフトウェア パッケージは、サーバやその他の技術情報を管理するためのオープンソースの自動化ツールセットです。Puppet ソフトウェアは、構成設定などのデバイス状態を適用することにより、サーバとリソースの管理を実現します。

Puppet コンポーネントには、管理対象デバイス（ノード）および Puppet Primary（サーバ）上で動作する Puppet エージェントが含まれます。通常、Puppet Primary は個別の専用サーバ上で実行され、複数のデバイスにサービスを提供します。Puppet エージェントの操作では、Puppet Primary に定期的に接続する必要があります。そして、Puppet Primary は構成マニフェストをコンパイルしてエージェントに送信します。エージェントは、ノードの現在の状態でこのマニフェストを調整し、相違点に基づいて状態を更新します。

Puppet マニフェストは、デバイスの状態を設定するためのプロパティ定義の集合です。これらのプロパティ状態の確認および設定の詳細は抽象化されているため、マニフェストは複数のオペレーティングシステムまたはプラットフォームで使用できます。マニフェストは、通常、構成時の設定を定義するために使用されますが、ソフトウェアパッケージのインストール、ファイルのコピー、およびサービスの開始にも使用できます。

詳細については、Puppet Labs を参照してください。

Puppet Labs	https://puppetlabs.com
Puppet Labs FAQ	https://puppet.com/blog/how-get-started-puppet-enterprise-faq/ https://puppet.com/products/faq
Puppet Labs ドキュメント	https://puppet.com/docs

前提条件

Puppet エージェントの前提条件は次のとおりです。

- インストールをサポートするスイッチおよびオペレーティングシステムソフトウェアリリースが必要です:
 - Cisco Nexus 3600 プラットフォーム スイッチ。
 - Cisco Nexus 3500 プラットフォーム スイッチ
 - Cisco Nexus 3100 プラットフォーム スイッチ。
 - Cisco Nexus 3000 シリーズ スイッチ。
 - Cisco NX-OS リリース 7.0 (3) I5 (1) 以降。
- 仮想サービスのインストールと Puppet Agent の展開に必要なディスク ストレージをデバイスで使用する必要があります。
 - ブートフラッシュに最低 450MB の空きディスク容量
- Puppet 4.0 以降の Puppet プライマリ サーバが必要です。
- Puppet エージェント 4.0 以降が必要です。

Puppet エージェント NX-OS 環境

Puppet Agent ソフトウェアは、ゲスト シェル (CentOS を実行する Linux コンテナ環境) のスイッチにインストールする必要があります。ゲスト シェルは、ホストから切り離された安全でオープンな実行環境を提供します。

Cisco NX-OS リリース 9.2 (1) 以降、Puppet Agent の Bash-shell (Cisco NX-OS の基盤となるネイティブ WindRiver Linux 環境) インストールはサポートされなくなりました。

次に、エージェント ソフトウェアのダウンロード、インストール、およびセットアップに関する情報を示します:

Puppet Agent: Cisco Nexus スイッチでのインストールとセットアップ (手動セットアップ)

<https://github.com/cisco/cisco-network-puppet-module/blob/develop/docs/README-agent-install.md>

ciscopuppet モジュール

ciscopuppet モジュールは、Cisco が開発したオープン ソース ソフトウェア モジュールです。これは、Puppet マニフェストの抽象技術情報構成と、Cisco NX-OS オペレーティング システムおよび

プラットフォームの特定の実装の詳細との間のインターフェイスとなります。このモジュールは Puppet プライマリにインストールされ、Cisco Nexus スイッチでの Puppet エージェントの操作に必要です。

ciscopuppet モジュールは、Puppet Forge で利用できます。

ここでは、ciscopuppet モジュール インストール手順についての追加情報を提供します：

ciscopuppet モジュールの場所 (Puppet Forge)	Puppet Forge
リソースの種類のカタログ	[Cisco Puppet 技術情報の参照先 (Cisco Puppet Resource Reference)]
ciscopuppet モジュール：送信元 コード リポジトリ	[Cisco Network Puppet モジュール (Cisco Network Puppet Module)]
ciscopuppet モジュール：セット アップと使用法	Cisco Puppet モジュール::README.md
Puppet Labs: モジュールのインス トール	https://puppet.com/docs/puppet/7/modules_installing.html https://docs.puppetlabs.com/puppet/latest/reference/modules_installing.html
Puppet NX-OS マニフェストの例	[Cisco Network Puppet モジュールの例 (Cisco Network Puppet Module Examples)]
NX-OS デベロッパーのランディ ング ページ。	構成管理ツール



第 10 章

SaltStack

この章は次のトピックで構成されています。

- [SaltStack について（109 ページ）](#)
- [注意事項と制約事項（110 ページ）](#)
- [SaltStack の Cisco NX-OS 環境（111 ページ）](#)
- [SaltStack の NX-API の有効化（111 ページ）](#)
- [NX-OS 用の SaltStack のインストール（112 ページ）](#)

SaltStack について

Cisco Nexus スイッチは、NX-OS を介して SaltStack をサポートします。Cisco NX-OS のサポートされているリリースの詳細については、<https://github.com/saltstack/salt/blob/develop/doc/topics/installation/nxos.rst#step-1-verify-platform-and-software-version-support> を参照してください。

SaltStack は、サーバーやその他のネットワーク デバイスの構成、管理、およびリモート実行のための無料のオープン ソース自動化フレームワークです。SaltStack フレームワークは、Salt プライマリと呼ばれるサーバーと、ミニオンと呼ばれるクライアントプログラムを実行する Salt ノードで構成されます。Cisco Nexus スイッチ（スイッチ）は、Salt ノードであり、Salt プライマリではありません。

SaltStack ミニオンは、スイッチに応じてオン ボックスまたはオフ ボックスのいずれかで実行して、構成または管理操作を実行できます。

- オンボックスでは、ミニオンはスイッチの Bash シェルで実行されます。これらのネイティブ ミニオンは、プライマリからリモート コマンドを受信して実行し、コマンドの結果をプライマリに中継します。オンボックス展開では、ミニオンはスイッチの Guest Shell で有効になっています。
- オフボックスでは、別のタイプのミニオンであるプロキシミニオンが、スイッチへの SSH 接続または NX-API を介して実行されます。プロキシミニオン（SSH プロキシミニオンまたは NX-API プロキシミニオン）は、コマンドを受信して実行します。その後、プロキシはコマンドの結果をプライマリに中継します。

鍵は、Salt プライマリと Cisco Nexus スイッチで実行されているミニオンとの間のセキュリティを確保するために使用されます。Salt のプライマリは、Cisco Nexus スイッチで実行されているミニオンとの接続を開始すると、最初にキーを渡します。ミニオンはキーを受け取り、正しい応答を計算し、キーをプライマリに送信します。プライマリは、キーの正しい応答値も計算しました。プライマリがミニオンからキーを受け取ると、キーが一致すると、セッションが開かれます。これにより、Salt プライマリはコマンドを送信できます。セッションは、電源を入れ直したり再起動したりしても持続しません。

SaltStack は、スイッチの CLI、プロパティ、および機能に影響を与える実行モジュールとソルトの状態を通じて、スイッチを管理および構成します。たとえば、モジュールを通じて、SaltStack を使用して Cisco Nexus スイッチをアップグレードできます。Salt プライマリは、自動化とスケーラビリティを活用するためにプログラムでコマンドを送信します。

詳細については、次のドキュメントを参照してください。

SaltStack	https://www.saltstack.com/
SaltStack のドキュメント	https://docs.saltstack.com/en/latest/
Cisco Nexus Salt Minion のインストールおよび構成ガイド	https://github.com/saltstack/salt/blob/develop/doc/topics/installation/nxos.rst

NX-OS と SaltStack について

Salt Open は、Salt 構成管理および分散リモート実行システムのオープンソースのコミュニティ版です。Cisco NX-OS は、物理スイッチと Salt Open ソフトウェアの間に中間層を提供します。Cisco NX-OS と Salt Open は相互運用して、Salt ミニオンと Cisco Nexus スイッチの間に API とコマンド実行レイヤを提供します。Cisco NX-OS はミニオンをホストし、次のように実行できるようにします。

- スイッチでは、Cisco NX-OS ゲストシェルが SaltStack ミニオンをホストし、統合インターフェイスを介して 1 つ以上のスイッチの自動オーケストレーションを提供します。ゲストシェルで実行されているミニオンはネイティブ ミニオンであり、NX-API を介して UNIX ドメインソケット (UDS) に接続します。
- スイッチをオフにすると、Salt プライマリはネットワークデバイス上で Salt Open ソフトウェアを実行し、SSH (SSH プロキシ ミニオン) または NX-API over HTTPS (NX-API プロキシ ミニオン) を介して NX-OS と通信します。Cisco NX-OS はコマンドを解釈し、必要な構成タスクを実行し、成功または失敗を適切なプロキシミニオンに報告します。プロキシミニオンは、このデータを Salt プライマリに送信します。

注意事項と制約事項

Cisco Nexus スイッチに SaltStack の導入の注意事項と制約事項は次のとおりです:

- SSH または NX-API HTTPS を介して SaltStack を実行している場合は、Salt を実行する前に NX-API 機能（**feature nxapi**）を有効にします。
- Salt のプライマリは、ポート 4506 でミニオンをリッスンします。このポートが開いていて（ブロックされていない）、別のサービスで使用されていないことを確認してください。

SaltStack の Cisco NX-OS 環境

Cisco NX-OS 環境は、Salt をオンボックスで実行しているかオフボックスで実行しているかによって異なります。

- スイッチをオンボックスで管理するには、ミニオンのホスティング環境であるゲスト シェルに SaltStack ミニオン RPM をインストールする必要があります。
- スイッチをオフボックスで管理するには、NX-OS で SSH または NX-API を有効にする必要があります。

SaltStack をサポートする Cisco Nexus スイッチなどの詳細については、<https://github.com/saltstack/salt/blob/develop/doc/topics/installation/nxos.rst#step-1-verify-platform-and-software-version-support> にアクセスしてください。

SaltStack の NX-API の有効化

始める前に

SSH または NX-API HTTPS で実行されているプロキシ ミニオンの場合、SaltStack が機能するには NX-API 機能を有効にする必要があります。デフォルトでは、NX-API が有効になっています。再度有効にする必要がある場合に備えて、次の手順を示します。

手順の概要

1. **config terminal**
2. **feature nxapi**

手順の詳細

手順

	コマンドまたはアクション	目的
Step 1	config terminal 例: <pre>switch-1# config terminal Enter configuration commands, one per line. End</pre>	コンフィギュレーション モードに入ります。

	コマンドまたはアクション	目的
	with CNTL/Z. switch-1 (config) #	
Step 2	feature nxapi 例: switch-1# feature nxapi switch-1# (config) #	プロキシ ミニオンの NX-API を有効にします。

次のタスク

ソルトスタックをインストールします。

NX-OS 用の SaltStack のインストール

次の設置ガイドを使用して、Cisco Nexus スイッチに SaltStack をインストールして起動します。

<https://github.com/saltstack/salt/blob/develop/doc/topics/installation/nxos.rst#cisco-nexus-salt-minion-installation-and-configuration-guide>



第 11 章

Cisco NX-OS でのシェフクライアントの使用

この章は、次の項で構成されています。

- [シェフについて](#) (113 ページ)
- [前提条件](#) (114 ページ)
- [Chef クライアント NX-OS 環境](#) (114 ページ)
- [cisco-cookbook](#) (115 ページ)

シェフについて

Chef は、Chef Software、Inc. によって開発されたオープンソース ソフトウェア パッケージです。これは、インフラストラクチャのサイズに関係なく、物理、仮想、またはクラウドの場所にサーバーとアプリケーションを導入する、システムおよびクラウドインフラストラクチャの自動化フレームワークです。各組織は、1 つ以上のワークステーション、単一サーバー、Chef クライアントが設定されていて、維持されているすべてのノードで構成されます。各ノードの設定方法について Chef クライアントに指示するために、クックブックとレシピが使用されます。すべてのノードにインストールされている Chef クライアントが、実際の設定を行います。

Chef クックブックは、設定とポリシーの配布の基本単位です。クックブックではシナリオを定義します。また、そのシナリオをサポートするために必要なすべての内容（ライブラリ、レシピ、ファイルなど）が含まれています。Chef レシピは、デバイスの状態を設定するためのプロパティ定義の集合です。これらのプロパティ状態の確認および設定の詳細は抽象化されているため、レシピは複数のオペレーティングシステムまたはプラットフォームで使用できます。レシピは、通常、構成時の設定を定義するために使用されますが、ソフトウェアパッケージのインストール、ファイルのコピー、およびサービスの開始などにも使用できます。

次のリファレンスは、Chef からの詳細情報を提供します。

トピック	リンク
Chef ホーム	https://www.chef.io
Chef の概要	https://docs.chef.io/chef_overview.html
Chef のドキュメント (すべて)	https://docs.chef.io/

前提条件

シェフの前提条件は次のとおりです：

- インストールをサポートする Cisco スイッチおよびオペレーティングシステム ソフトウェア リリースが必要です：
 - Cisco Nexus 3500 プラットフォーム スイッチ
 - Cisco Nexus NX-OS リリース 6.1 (2) I3 (4)
- シェフの展開に必要なディスク ストレージがデバイス上に用意されている必要があります：
 - ブートフラッシュに最低 500 MB の空きディスク容量
- シェフ 12.4.1 以降のシェフ サーバが必要です。
- シェフ クライアント 12.4.1 以降が必要です。

Chef クライアント NX-OS 環境

Chef クライアント ソフトウェアは、Cisco Nexus スイッチにインストールする必要があります。お客様は、Cisco Nexus スイッチが提供する Linux 環境のいずれかに Chef クライアントをインストールできます。

- Bash Shell — これは、Cisco NX-OS の基礎となるネイティブの WindRiver Linux 環境です。
- Guest Shell — これは、CentOS を実行する安全な Linux コンテナ環境です。その利点は、ホストから切り離された安全でオープンな実行環境です。

両方の使用例のワークフローは似ています。

次のドキュメントには、エージェント ソフトウェアのダウンロード、インストール、およびセットアップに関する段階的なガイダンスが記載されています。

トピック	リンク
Chef クライアント (ネイティブ)	クライアント RPM の最新情報は、 ここで 入手できます。
Chef クライアント (Guest Shell、CentOs7)	クライアント RPM の最新情報は、 ここで 入手できます。
Chef クライアント：Cisco Nexus プラットフォームでのインストールとセットアップ（手動セットアップ）	cisco-cookbook::README-install-agent.md

トピック	リンク
Chef クライアント: Cisco Nexus プラットフォームでのインストールとセットアップ (Chef プロビジョナを使用した自動インストール)	cisco-cookbook::README-chef-provisioning.md

cisco-cookbook

cisco-cookbook は、Chef レシピの抽象リソース構成と、Cisco NX-OS オペレーティング システム および Cisco Nexus スイッチの特定の実装の詳細との間の、シスコが開発したオープン ソース インターフェイスです。このクックブックは ChefServer にインストールされ、Cisco Nexus スイッチでの Chef Client の適切な動作に必要です。

cisco-cookbook は、Chef Supermarket にあります。

次のドキュメントには、cisco-cookbook および一般的なクックブックのインストール手順の詳細が記載されています。

トピック	リンク
cisco-cookbook の場所	https://supermarket.chef.io/cookbooks/cisco-cookbook
リソースの種類のカタログ	https://github.com/cisco/cisco-network-chef-cookbook#resource-by-tech
cisco-cookbook: ソース コード リポジトリ	https://github.com/cisco/cisco-network-chef-cookbook
cisco-cookbook: セットアップと使用法	cisco-cookbook::README.md
Chef Supermarket	https://supermarket.chef.io
NX-OS デベロッパーのランディング ページ。	構成管理ツール



第 12 章

Cisco NX-OS での Docker の使用

この章は次のトピックで構成されています。

- Cisco NX-OS での Docker について (117 ページ)
- 注意事項と制約事項 (118 ページ)
- Cisco NX-OS 内で Docker コンテナを設定するための前提条件 (118 ページ)
- Docker デーモンの開始 (119 ページ)
- 自動的に起動するように Docker を構成する (119 ページ)
- Docker コンテナの開始: ホスト ネットワーク モデル (120 ページ)
- Docker コンテナの開始: ブリッジ型ネットワーク モデル (121 ページ)
- Docker コンテナでのブートフラッシュおよび揮発性パーティションのマウント (122 ページ)
- 拡張 ISSU スイッチオーバーでの Docker デーモンの永続性の有効化 (123 ページ)
- Cisco Nexus Platform Switches Switchover 時に Docker デーモンの永続性を有効にする (124 ページ)
- Docker ストレージバックエンドのサイズ変更 (125 ページ)
- Docker デーモンの停止 (127 ページ)
- Docker コンテナ セキュリティ (128 ページ)
- Kubernetes クラスタにノードを追加 (130 ページ)
- Docker のトラブルシューティング (132 ページ)

Cisco NX-OS での Docker について

Docker は、すべての依存関係とライブラリと共にパッケージ化された、コンテナ内で安全に分離されたアプリケーションを実行する方法を提供します。Docker の詳細にを表示するために <https://docs.docker.com/> を参照してください。

Cisco NX-OS リリース 9.2 (1) 以降、スイッチ上の Cisco NX-OS 内で Docker を使用するためのサポートが追加されました。

スイッチに含まれる Docker のバージョンは CE 18.09.0 です。Docker デーモンはデフォルトでは実行されていません。手動で起動するか、スイッチの起動時に自動的に再起動するように設定する必要があります。

このセクションでは、スイッチ環境の特定のコンテキストで Docker を有効にして使用方法について説明します。一般的な Docker の使用方法と機能の詳細については、<https://docs.docker.com/>にある Docker のドキュメントを参照してください。

注意事項と制約事項

次に、スイッチ上の Cisco NX-OS で Docker を使用するためのガイドラインと制限事項を示します。

- Docker 機能は、少なくとも 8 GB のシステム RAM を備えたスイッチでサポートされています。

Cisco NX-OS 内で Docker コンテナを設定するための前提条件

スイッチの Cisco NX-OS で Docker を使用するための前提条件は次のとおりです：

- ホスト Bash シェルを有効にします。スイッチの Cisco NX-OS で Docker を使用するには、ホスト Bash シェルのルートユーザーである必要があります：

```
switch# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
switch(config)# feature bash-shell
```

- スイッチが HTTP プロキシ サーバを使用するネットワーク内にある場合、`http_proxy` と `https_proxy` 環境変数を `/etc/sysconfig/docker` に構成する必要があります。例：

```
export http_proxy=http://proxy.esl.cisco.com:8080
export https_proxy=http://proxy.esl.cisco.com:8080
```

- スイッチのクロックが正しく設定されていることを確認してください。そうしないと、次のエラー メッセージが表示される場合があります：

```
x509: certificate has expired or is not yet valid
```

- ドメイン名とネーム サーバがネットワークに対して適切に構成されていること、および `/etc/resolv.conf` ファイルに反映されていることを確認します：

```
switch# conf t
Enter configuration commands, one per line. End with CNTL/Z.
switch(config)# vrf context management
switch(config-vrf)# ip domain-name ?
WORD Enter the default domain (Max Size 64)

switch(config-vrf)# ip name-server ?
A.B.C.D Enter an IPv4 address
A:B::C:D Enter an IPv6 address

root@switch# cat /etc/resolv.conf
domain cisco.com #bleed
```

```
nameserver 171.70.168.183 #bleed
root@switch#
```

Docker デーモンの開始

初めて Docker デーモンを開始すると、固定サイズのバックエンドストレージスペースがブートフラッシュの `dockerpart` と呼ばれるファイルに切り出され、次に `/var/lib/docker` にマウントされます。必要に応じて、Docker デーモンを初めて開始する前に `/etc/sysconfig/docker` を編集して、この領域のデフォルトサイズを調整できます。後で説明するように、必要に応じてこのストレージスペースのサイズを変更することもできます。

Docker デーモンを開始するには:

手順

Step 1 Bash を読み込み、スーパーユーザーになります。

```
switch# run bash sudo su -
```

Step 2 Docker デーモンを起動します。

```
root@switch# service docker start
```

Step 3 ステータスをチェックします。

```
root@switch# service docker status
dockerd (pid 3597) is running...
root@switch#
```

(注)

Docker デーモンを起動したら、ブートフラッシュの `dockerpart` ファイルを削除したり、改ざんしたりしないでください。これは、docker の機能にとって重要であるからです。

```
switch# dir bootflash:dockerpart
20000000000 Mar 14 12:50:14 2018 dockerpart
```

自動的に起動するように Docker を構成する

スイッチの起動時に常に自動的に起動するように Docker デーモンを構成できます。

手順

Step 1 Bash をロードしてスーパーユーザーになります。

```
switch# run bash sudo su -
```

Step 2 chkconfig ユーティリティを使用して、Docker サービスを永続化します。

```
root@switch# chkconfig --add docker
root@n9k-2#
```

Step 3 chkconfig ユーティリティを使用して、Docker サービスの設定を確認します。

```
root@switch# chkconfig --list | grep docker
docker 0:off 1:off 2:on 3:on 4:on 5:on 6:off
root@switch#
```

Step 4 Docker が自動的に起動しないように構成を削除するには:

```
root@switch# chkconfig --del docker
root@switch# chkconfig --list | grep docker
root@switch#
```

Docker コンテナの開始: ホスト ネットワーク モデル

Docker コンテナがデータ ポートと管理を含むすべてのホスト ネットワーク インターフェイスにアクセスできるようにする場合は、`--network host` オプションを使用して Docker コンテナを起動します。コンテナ内のユーザーは、`ip netns exec <net_namespace> <cmd>` を使用して、`/var/run/netns`（Cisco NX-OS で設定されたさまざまな VRF に対応）でさまざまなネットワーク名前空間を切り替えることができます。

手順

Step 1 Bash をロードしてスーパーユーザーになります。

```
switch# run bash sudo su -
```

Step 2 Docker コンテナを開始します。

以下は、スイッチで Alpine Docker コンテナを起動し、すべてのネットワーク インターフェイスを表示する例です。コンテナは、デフォルトで管理ネットワークの名前空間で起動されます。

```
root@switch# docker run --name=alpine-run -v /var/run/netns:/var/run/netns:ro,rslave --rm --network
host --cap-add SYS_ADMIN -it alpine
/ # apk --update add iproute2
fetch http://dl-cdn.alpinelinux.org/alpine/v3.7/main/x86_64/APKINDEX.tar.gz
fetch http://dl-cdn.alpinelinux.org/alpine/v3.7/community/x86_64/APKINDEX.tar.gz
(1/6) Installing libelf (0.8.13-r3)
(2/6) Installing libmnl (1.0.4-r0)
(3/6) Installing jansson (2.10-r0)
(4/6) Installing libnftnl-libs (1.0.8-r1)
(5/6) Installing iptables (1.6.1-r1)
```

```
(6/6) Installing iproute2 (4.13.0-r0)
Executing iproute2-4.13.0-r0.post-install
Executing busybox-1.27.2-r7.trigger
OK: 7 MiB in 17 packages
/ #
/ # ip netns list
management
default
/ #
/ # ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
valid_lft forever preferred_lft forever
inet6 ::1/128 scope host
valid_lft forever preferred_lft forever
2: tunl0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default
link/ipip 0.0.0.0 brd 0.0.0.0
3: gre0@NONE: <NOARP> mtu 1476 qdisc noop state DOWN group default
link/gre 0.0.0.0 brd 0.0.0.0
...
/ #
/ # ip netns exec default ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/16 scope host lo
valid_lft forever preferred_lft forever
2: dummy0: <BROADCAST,NOARP> mtu 1500 qdisc noop state DOWN group default
link/ether 42:0d:9b:3c:d4:62 brd ff:ff:ff:ff:ff:ff
3: tunl0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default
link/ipip 0.0.0.0 brd 0.0.0.0
...
```

Docker コンテナの開始: ブリッジ型ネットワーク モデル

Docker コンテナに外部ネットワーク接続（通常は管理インターフェースを介して）のみを許可し、特定のデータポートまたは他のスイッチインターフェースへの可視性を必ずしも気にしない場合は、デフォルトの Docker ブリッジ ネットワーク モデルで Docker コンテナを開始できます。これは、ネットワーク名前空間の分離も提供するため、前のセクションで説明したホスト ネットワーキング モデルよりも安全です。

手順

Step 1 Bash をロードしてスーパーユーザーになります。

```
switch# run bash sudo su -
```

Step 2 Docker コンテナを開始します。

以下は、スイッチで Alpine Docker コンテナを開始し、iproute2 パッケージをインストールする例です。

```

root@switch# docker run -it --rm alpine
/ # apk --update add iproute2
fetch http://dl-cdn.alpinelinux.org/alpine/v3.7/main/x86_64/APKINDEX.tar.gz
fetch http://dl-cdn.alpinelinux.org/alpine/v3.7/community/x86_64/APKINDEX.tar.gz
(1/6) Installing libelf (0.8.13-r3)
(2/6) Installing libmnl (1.0.4-r0)
(3/6) Installing jansson (2.10-r0)
(4/6) Installing libnftnl-libs (1.0.8-r1)
(5/6) Installing iptables (1.6.1-r1)
(6/6) Installing iproute2 (4.13.0-r0)
Executing iproute2-4.13.0-r0.post-install
Executing busybox-1.27.2-r7.trigger
OK: 7 MiB in 17 packages
/ #
/ # ip netns list
/ #

```

Step 3 ユーザー名前空間の分離を設定するかどうかを決定します。

ブリッジネットワーク モデルを使用するコンテナの場合、ユーザー名前空間の分離を設定して、セキュリティをさらに向上させることもできます。詳細については、「[ユーザー\[名前空間 \(namespace\)\]の分離による Docker コンテナの保護 \(128 ページ\)](#)」を参照してください。

標準の Docker ポート オプションを使用して、sshd などのコンテナ内からサービスを公開できます。例:

```
root@switch# docker run -d -p 18877:22 --name sshd_container sshd_ubuntu
```

これにより、コンテナ内のポート 22 がスイッチのポート 18877 にマップされます。次の例に示すように、ポート 18877 を介してサービスに外部からアクセスできるようになりました。

```
root@ubuntu-vm# ssh root@ip_address -p 18887
```

Docker コンテナでのブートフラッシュおよび揮発性パーティションのマウント

Docker コンテナの `run` コマンドで `-v /bootflash:/bootflash` および `-v /volatile:/volatile` オプションを渡すことで、ブートフラッシュおよび揮発性パーティションを Docker コンテナに表示できます。これは、新しい NX-OS システム イメージをブートフラッシュにコピーするなど、コンテナ内のアプリケーションがホストと共有するファイルにアクセスする必要がある場合に役立ちます。



- (注) この `-v` コマンドオプションを使用すると、任意のディレクトリをコンテナにマウントでき、NX-OS システムの動作に影響を与える可能性のある情報漏えいやその他のアクセスが発生する可能性があります。これを、NX-OS CLI を使用してすでにアクセス可能な `/bootflash` や `/volatile` などのリソースに制限します。

手順

Step 1 Bash をロードしてスーパーユーザーになります。

```
switch# run bash sudo su -
```

Step 2 Docker コンテナの実行コマンドに `-v /bootflash:/bootflash` および `-v /volatile:/volatile` オプションを渡します。

```
root@switch# docker run -v /bootflash:/bootflash -v /volatile:/volatile -it --rm alpine
/# ls /
bin          etc          media        root          srv          usr
bootflash    home         mnt          run           sys          var
dev          lib          proc         sbin          tmp          volatile
/ #
```

拡張 ISSU スイッチオーバーでの Docker デーモンの永続性の有効化

Docker デーモンと実行中のコンテナの両方を拡張 ISSU スイッチオーバーで持続させることができます。これが可能なのは、バックエンドの Docker ストレージが存在するブートフラッシュが同じであり、アクティブ スーパーバイザとスタンバイ スーパーバイザの両方で共有されるためです。

Docker コンテナは、切り替え中に中断（再起動）されるため、継続的に実行されません。

手順

Step 1 Bash をロードしてスーパーユーザーになります。

```
switch# run bash sudo su -
```

Step 2 スイッチオーバーを開始する前に、`chkconfig` ユーティリティを使用して Docker サービスを永続化します。

```
root@switch# chkconfig --add docker
root@n9k-2#
```

Step 3 スイッチオーバー後にコンテナが自動的に再起動されるように、`--restart without-stopped` オプションを使用してコンテナを起動します。

次の例では、Alpine コンテナを開始し、明示的に停止するか、Docker を再起動しない限り、常に再起動するように構成します。

```
root@switch# docker run -dit --restart unless-stopped alpine
root@n9k-2#
```

Docker コンテナは、切り替え中に中断（再起動）されるため、継続的に実行されません。

Cisco Nexus Platform Switches Switchover 時に Docker デーモンの永続性を有効にする

Docker デーモンと実行中のコンテナの両方を、個別のブートフラッシュ パーティションを持つ 2 つの個別の物理スーパーバイザ間のスイッチオーバーで持続させることができます。ただし、Cisco Nexus スイッチの場合、両方のスーパーバイザのブートフラッシュ パーティションは物理的に分離されています。したがって、スイッチオーバーを実行する前に、dockerpart ファイルをスタンバイ スーパーバイザに手動でコピーする必要があります。

手順

Step 1 Bash を読み込みしてスーパーユーザーになります。

```
switch# run bash sudo su -
```

Step 2 スイッチオーバー後にコンテナが自動的に再起動されるように、--restart without-stopped オプションを使用してコンテナを起動します。

次の例では、Alpine コンテナを開始し、明示的に停止するか、Docker を再起動しない限り、常に再起動するように構成します。

```
root@switch# docker run -dit --restart unless-stopped alpine
root@n9k-2#
```

Docker コンテナは切り替え中に中断（再起動）されるため、継続的に実行されないことに注意してください。

Step 3 スイッチオーバーを開始する前に、chkconfig ユーティリティを使用して Docker サービスを永続化します。

```
root@switch# chkconfig --add docker
root@n9k-2#
```

Step 4 Docker バックエンドストレージパーティションを現用系からスタンバイ スーパーバイザブートフラッシュにコピーします。

```
root@switch# service docker stop
Stopping dockerd: dockerd shutdown

root@switch# cp /bootflash/dockerpart /bootflash_sup-remote/
```



```
root@switch# service docker start
```

Docker ストレージ バックエンドのサイズ変更

Docker デーモンを起動または使用した後、必要に応じて Docker バックエンド ストレージ スペースのサイズを増やすことができます。

手順

Step 1

Guest Shell を無効にします。

ゲスト シェルを無効にしないと、サイズ変更が妨げられる可能性があります。

```
switch# guestshell disable
You will not be able to access your guest shell if it is disabled. Are you sure you want to disable
the guest shell? (y/n) [n] y
switch# 2018 Mar 15 17:16:55 switch %$ VDC-1 %$ %VMAN-2-ACTIVATION_STATE: Deactivating virtual
service 'guestshell+'
2018 Mar 15 17:16:57 switch %$ VDC-1 %$ %VMAN-2-ACTIVATION_STATE: Successfully deactivated virtual
service 'guestshell+'
```

Step 2

Bash をロードしてスーパーユーザーになります。

```
switch# run bash sudo su -
```

Step 3

現在利用可能なストレージ容量に関する情報を取得します。

```
root@switch# df -kh /var/lib/docker
Filesystem Size Used Avail Use% Mounted on
/dev/loop12 1.9G 7.6M 1.8G 1% /var/lib/docker
root@n9k-2#
```

Step 4

Docker デーモンを停止します。

```
root@switch# service docker stop
Stopping dockerd: dockerd shutdown
```

Step 5

Docker バックエンド ストレージ スペース（/bootflash/dockerpart）の現在のサイズに関する情報を取得します。

```
root@switch# ls -l /bootflash/dockerpart
-rw-r--r-- 1 root root 2000000000 Mar 15 16:53 /bootflash/dockerpart
root@n9k-2#
```

Step 6

Docker バックエンド ストレージ スペースのサイズを変更します。

たとえば、次のコマンドはサイズを 500 メガバイト増やします。

```
root@switch# truncate -s +500MB /bootflash/dockerpart
root@n9k-2#
```

Step 7 Docker バックエンド ストレージ スペースのサイズに関する最新情報を取得して、サイズ変更プロセスが正常に完了したことを確認します。

たとえば、次の出力は、Docker バックエンドストレージのサイズが 500 メガバイト増加したことを確認します。

```
root@switch# ls -l /bootflash/dockerpart
-rw-r--r-- 1 root root 2500000000 Mar 15 16:54 /bootflash/dockerpart
root@n9k-2#
```

Step 8 /bootflash/dockerpart のファイル システムのサイズを確認します。

```
root@switch# e2fsck -f /bootflash/dockerpart
e2fsck 1.42.9 (28-Dec-2013)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/bootflash/dockerpart: 528/122160 files (0.6% non-contiguous), 17794/488281 blocks
```

Step 9 /bootflash/dockerpart のファイル システムのサイズを変更します。

```
root@switch# /sbin/resize2fs /bootflash/dockerpart
resize2fs 1.42.9 (28-Dec-2013)
Resizing the filesystem on /bootflash/dockerpart to 610351 (4k) blocks.
The filesystem on /bootflash/dockerpart is now 610351 blocks long.
```

Step 10 /bootflash/dockerpart のファイル システムのサイズを再度チェックして、ファイル システムのサイズが正常に変更されたことを確認します。

```
root@switch# e2fsck -f /bootflash/dockerpart
e2fsck 1.42.9 (28-Dec-2013)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/bootflash/dockerpart: 528/154736 files (0.6% non-contiguous), 19838/610351 blocks
```

Step 11 Daemon デーモンを再起動します。

```
root@switch# service docker start
Updating certificates in /etc/ssl/certs...
0 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...
done.
Starting dockerd with args '--debug=true':
```

Step 12 使用可能なストレージ領域の大きさを確認します。

```
root@switch# df -kh /var/lib/docker
Filesystem Size Used Avail Use% Mounted on
/dev/loop12 2.3G 7.6M 2.3G 1% /var/lib/docker
```

Step 13 BASH シェルを終了します。

```
root@switch# exit
logout
switch#
```

Step 14 Guest Shell を有効にします。

```
switch# guestshell enable

switch# 2018 Mar 15 17:12:53 switch %$ VDC-1 %$ %VMAN-2-ACTIVATION_STATE: Activating virtual service
'guestshell+'
switch# 2018 Mar 15 17:13:18 switch %$ VDC-1 %$ %VMAN-2-ACTIVATION_STATE: Successfully activated
virtual service 'guestshell+'
```

Docker デーモンの停止

Docker を今後使用しない場合は、このトピックの手順に従って Docker デーモンを停止します。

手順

Step 1 Bash をロードしてスーパーユーザーになります。

```
switch# run bash sudo su -
```

Step 2 Docker デーモンを停止します。

```
root@switch# service docker stop
Stopping dockerd: dockerd shutdown
```

Step 3 Docker デーモンが停止していることを確認します。

```
root@switch# service docker status
dockerd is stopped
root@switch#
```

(注)

必要に応じて、この時点でブートフラッシュの dockerpart ファイルを削除することもできます。

```
switch# delete bootflash:dockerpart
Do you want to delete "/dockerpart" ? (yes/no/abort) y
switch#
```

Docker コンテナ セキュリティ

Docker コンテナのセキュリティに関する推奨事項は次のとおりです。

- 可能であれば、別のユーザー [名前空間 (namespace)] で実行します。
- 可能であれば、別のネットワーク [名前空間 (namespace)] で実行します。
- **cgroup** を使用して技術情報を制限します。既存の **cgroup** (`ext_ser`) が作成され、ホストされているアプリケーションを、プラットフォーム チームがスイッチで実行される追加のソフトウェアに対して妥当と見なしたものに制限します。Docker では、これを使用して、コンテナごとの技術情報を制限できます。
- 不要な POSIX 機能を追加しないでください。

ユーザー[名前空間 (namespace)]の分離による Docker コンテナの保護

ブリッジネットワークモデルを使用するコンテナの場合、ユーザー名前空間の分離を設定して、セキュリティをさらに向上させることもできます。詳細については、「<https://docs.docker.com/engine/security/usersns-remap/>」を参照してください。

手順

Step 1 システムに `dockremap` グループがすでに存在するかどうかを確認します。

`dockremap` ユーザーは、デフォルトでシステムにすでに設定されている必要があります。`dockremap` グループがまだ存在しない場合は、次の手順に従って作成します。

a) 次のコマンドを入力して `dockremap` グループを作成します。

```
root@switch# groupadd dockremap -r
```

b) `dockremap` ユーザーを作成します（まだ存在していない場合）。

```
root@switch# useradd dockremap -r -g dockremap
```

c) `dockremap` グループと `dockremap` ユーザーが正常に作成されたことを確認します。

```
root@switch# id dockremap
uid=999(dockremap) gid=498(dockremap) groups=498(dockremap)
root@switch#
```

Step 2 再マップされた必要な ID と範囲を `/etc/subuid` と `/etc/subgid` に追加します。

例:

```
root@switch# echo "dockremap:123000:65536" >> /etc/subuid
root@switch# echo "dockremap:123000:65536" >> /etc/subgid
```

Step 3 テキスト エディタを使用して、`--userns-remap=default` オプションを `/etc/sysconfig/docker` ファイルの `other_args` フィールドに追加します。

例:

```
other_args="-debug=true --userns-remap=default"
```

Step 4 [サービス ドッカー [re]start (service docker [re]start)] を使用して、Docker デーモンを再起動するか、まだ実行されていない場合は起動します。

例:

```
root@switch# service docker [re]start
```

ユーザー名前空間の分離によるコンテナの構成と使用の詳細については、<https://docs.docker.com/engine/security/userns-remap/>で Docker のドキュメントを参照してください。

cgroup パーティションの移動

サードパーティ サービスの cgroup パーティションは `ext_ser` で、CPU 使用率をコアあたり 25% に制限します。この `ext_ser` パーティションで Docker コンテナを実行することをお勧めします。

`--cgroup-parent=/ext_ser/` オプションを指定せずに Docker コンテナを実行すると、最大 100% のホスト CPU アクセスが可能になり、Cisco NX-OS の通常の動作を妨げる可能性があります。

手順

Step 1 Bash をロードしてスーパーユーザーになります。

```
switch# run bash sudo su -
```

Step 2 `ext_ser` パーティションで Docker コンテナを実行します。

例:

```
root@switch# docker run --name=alpinerrun -v /var/run/netns:/var/run/netns:ro,rslave --rm --network
host --cgroup-parent=/ext_ser/ --cap-add SYS_ADMIN -it alpine
/ #
```

Kubernetes クラスタにノードを追加

このトピックでは、Kubernetes クラスタにノードを追加する方法について説明します。この例では、次のようになります。

- Kubernetes (Ubuntu) プライマリの IP アドレスは 10.122.197.246 です。
- Docker コンテナとして実行されているスイッチソフトウェアの IP アドレスは 10.122.84.24 です。



(注) 次の例では、読みやすくなるように、長い行を \ 文字で分割しています。

手順

Step 1 Kubernetes (Ubuntu) プライマリに対して次のコマンドを (on? for?) 実行します。

a) コマンド

```
root@switch# docker run -d --net=host gcr.io/google_containers/etcd:2.2.1 /usr/local/bin/etcd
--listen-client-urls=http://0.0.0.0:4001 --advertise-client-urls=http://0.0.0.0:4001
--data-dir=/var/etcd/data
```

b) コマンド

```
root@switch# docker run -d --name=api --net=host --pid=host --privileged=true
gcr.io/google_containers/hyperkube:v1.2.2 /hyperkube apiserver --insecure-bind-address=0.0.0.0
--allow-privileged=true --service-cluster-ip-range=10.0.0.1/24
--etcd_servers=http://127.0.0.1:4001 --v=2
```

c) コマンド

```
root@switch# docker run -d --name=kubs --volume=/:/rootfs:ro --volume=/sys:/sys:ro
--volume=/dev:/dev --volume=/var/lib/docker/:/var/lib/docker:rw
--volume=/var/lib/kubelet/:/var/lib/kubelet:rw --volume=/var/run:/var/run:rw --net=host --pid=host
--privileged=true gcr.io/google_containers/hyperkube:v1.2.2 /hyperkube kubelet
--allow-privileged=true --hostname-override="127.0.0.1" --address="0.0.0.0"
--api-servers=http://0.0.0.0:8080 --cluster_dns=10.0.0.10 --cluster_domain=cluster.local
--config=/etc/kubernetes/manifests-multi
```

d) コマンド

```
root@switch# docker run -d --name=proxy --net=host --privileged
gcr.io/google_containers/hyperkube:v1.2.2 /hyperkube proxy --master=http://0.0.0.0:8080 --v=2
```

e) コマンド

```
root@switch# export KUBERNETES_MASTER=http://10.122.197.246:8080
```

f) コマンド

```
root@switch# curl -o /usr/bin/kubectl
http://storage.googleapis.com/kubernetes-release/release/v1.2.2/bin/linux/amd64/kubectl
```

g) コマンド

```
root@switch# kubectl -s $KUBERNETES_MASTER create -f kube-system.json
```

h) コマンド

```
root@switch# kubectl -s $KUBERNETES_MASTER create -f skydns-rc.yaml
```

i) コマンド

```
root@switch# kubectl -s $KUBERNETES_MASTER create -f skydns-svc.yaml
```

j) コマンド

```
root@switch# kubectl -s $KUBERNETES_MASTER create -f dashboard.yaml
kubectl -s $KUBERNETES_MASTER cluster-info
```

k) コマンド

```
root@switch# kubectl -s $KUBERNETES_MASTER cluster-info
```

Step 2 スイッチに対して次の手順 (on? for?) を実行します。

a) コマンド

```
root@switch# docker run -d --name=kubs --net=host --pid=host --privileged=true --volume=:/rootfs:ro
--volume=/sys:/sys:ro --volume=/dev:/dev --volume=/var/lib/docker:/var/lib/docker:rw
--volume=/var/lib/kubelet:/var/lib/kubelet:rw --volume=/var/run:/var/run:rw
gcr.io/google_containers/hyperkube:v1.2.2 /hyperkube kubelet --allow-privileged=true --containerized
--enable-server --cluster_dns=10.0.0.10 --cluster_domain=cluster.local
--config=/etc/kubernetes/manifests-multi --hostname-override="10.122.84.34" --address=0.0.0.0
--api-servers=http://10.122.197.246:8080
```

b) コマンド

```
root@switch# docker run -d --name=proxy --net=host --privileged=true
gcr.io/google_containers/hyperkube:v1.2.2 /hyperkube proxy --master=http://10.122.197.246:8080
--v=2
```

Step 3 Kubernetes (Ubuntu) プライマリで次のコマンド (on? for?) を実行して、レプリケーションコントローラーオブジェクトに nginx アプリをデプロイします。

a) コマンド

```
lab@rnimbalk-ubuntu1:~$ kubectl get node
NAME                STATUS    AGE
10.122.84.34        Ready    16m
127.0.0.1           Ready    22m
```

b) コマンド

```
lab@rnlmbalk-ubuntu1:~$ kubectl apply -f replication.yaml
replicationcontroller "nginx" created
```

c) コマンド

```
lab@rnlmbalk-ubuntu1:~$ kubectl describe -f replication.yaml
Name:          nginx
Namespace:     default
Image(s):      nginx
Selector:      app=nginx
Labels:        app=nginx
Replicas:      3 current / 3 desired
Pods Status:   3 Running / 0 Waiting / 0 Succeeded / 0 Failed
No volumes.
Events:
  FirstSeen    LastSeen    Count   From              SubobjectPath  Type
  Reason
  -----
  17s          17s         1       {replication-controller }
    SuccessfulCreate Created pod: nginx-zqfpz
  17s          17s         1       {replication-controller }
    SuccessfulCreate Created pod: nginx-jij40
  17s          17s         1       {replication-controller }
    SuccessfulCreate Created pod: nginx-loa0g
```

Docker のトラブルシューティング

これらのトピックでは、Docker コンテナで発生する可能性のある問題について説明し、考えられる解決策を提供します。

Docker の起動が機能不全になる

[問題: (Problem:)] Docker の起動に失敗し、次のようなエラー メッセージが表示されます:

```
switch# run bash
bash-4.3$ service docker start
Free bootflash: 39099 MB, total bootflash: 51771 MB
Carving docker bootflash storage: 2000 MB
2000+0 records in
2000+0 records out
2000000000 bytes (2.0 GB) copied, 22.3039 s, 89.7 MB/s
losetup: /dev/loop18: failed to set up loop device: Permission denied
mke2fs 1.42.9 (28-Dec-2013)
mkfs.ext4: Device size reported to be zero. Invalid partition specified, or
partition table wasn't reread after running fdisk, due to
a modified partition being busy and in use. You may need to reboot
to re-read your partition table.

Failed to create docker volume
```


[考えられる原因: (Possible Cause:)] root ユーザーではなく、管理ユーザーとして Bash を実行している可能性があります。

[解決策: (Solution:)] root ユーザーではなく、管理ユーザーとして Bash を実行しているかどうかを確認します。

```
bash-4.3$ whoami
admin
```

Bash を終了し、ルート ユーザーとして Bash を実行します:

```
bash-4.3$ exit
switch# run bash sudo su -
```

ストレージが不足しているため、Docker が起動に失敗する

問題: ブートフラッシュストレージが不足しているため、Docker の起動に失敗し、次のようなエラーメッセージが表示されます。

```
root@switch# service docker start
Free bootflash: 790 MB, total bootflash: 3471 MB
Need at least 2000 MB free bootflash space for docker storage
```

考えられる原因: 十分な空きブートフラッシュ ストレージがない可能性があります。

解決策: スペースを解放するか、必要に応じて `/etc/sysconfig/docker` の変数 `_dockerstrg` 値を調整してから、Docker デーモンを再起動します。

```
root@switch# cat /etc/sysconfig/docker
# Replace the below with your own docker storage backend boundary value (in MB)
# if desired.
boundary_dockerstrg=5000

# Replace the below with your own docker storage backend values (in MB) if
# desired. The smaller value applies to platforms with less than
# $boundary_dockerstrg total bootflash space, the larger value for more than
# $boundary_dockerstrg of total bootflash space.
small_dockerstrg=300
large_dockerstrg=2000
```

Docker Hub からのイメージのプルの失敗（509 証明書失効 エラー メッセージ）

問題: システムが Docker ハブからイメージをプルできず、次のようなエラーメッセージが表示されます。

```
root@switch# docker pull alpine
Using default tag: latest
Error response from daemon: Get https://registry-1.docker.io/v2/: x509: certificate has
expired or is not yet valid
```

[考えられる原因: (Possible Cause:)] システムクロックが正しく設定されていない可能性があります。

[解決策: (Solution:)] クロックが正しく設定されているかどうかを確認します。

```
root@n9k-2# sh clock
15:57:48.963 EST Thu Apr 25 2002
Time source is Hardware Calendar
```

必要に応じて、時計をリセットします:

```
root@n9k-2# clock set hh:mm:ss { day month | month day } year
```

例:

```
root@n9k-2# clock set 14:12:00 10 feb 2018
```

DockerHubからのイメージのプルの失敗（クライアントタイムアウトエラーメッセージ）

問題: システムが Docker ハブからイメージをプルできず、次のようなエラーメッセージが表示されます。

```
root@switch# docker pull alpine
Using default tag: latest
Error response from daemon: Get https://registry-1.docker.io/v2/: net/http: request canceled
while waiting for connection (Client.Timeout exceeded while awaiting headers)
```

考えられる原因: プロキシまたは DNS 設定が正しく設定されていない可能性があります。

解決策: プロキシ設定を確認し、必要に応じて修正してから、Docker デーモンを再起動します。

```
root@switch# cat /etc/sysconfig/docker | grep proxy
#export http_proxy=http://proxy.esl.cisco.com:8080
#export https_proxy=http://proxy.esl.cisco.com:8080
root@switch# service docker [re]start
```

DNS 設定を確認し、必要に応じて修正してから、Docker デーモンを再起動します。

```
root@switch# cat /etc/resolv.conf
domain cisco.com #bleed
nameserver 171.70.168.183 #bleed
root@switch# # conf t
Enter configuration commands, one per line. End with CNTL/Z.
switch(config)# vrf context management
switch(config-vrf)# ip domain-name ?
WORD Enter the default domain (Max Size 64)

switch(config-vrf)# ip name-server ?
A.B.C.D Enter an IPv4 address
A:B::C:D Enter an IPv6 address
root@switch# service docker [re]start
```

スイッチのリロードまたはスイッチオーバーで Docker デーモンまたはコンテナが実行されない

問題: スwitchのリロードまたはスイッチオーバーを実行した後、Docker デーモンまたはコンテナが実行されません。

考えられる原因: Docker デーモンが、スイッチのリロードまたはスイッチオーバーで持続するように構成されていない可能性があります。

解決策: Docker デーモンが `chkconfig` コマンドを使用してスイッチのリロードまたはスイッチオーバーで持続するように構成されていることを確認してから、`--restart unless-stopped` オプションを使用して必要な Docker コンテナを開始します。たとえば、Alpine コンテナを開始するには:

```
root@switch# chkconfig --add docker
root@switch#
root@switch# chkconfig --list | grep docker
docker 0:off 1:off 2:on 3:on 4:on 5:on 6:off
root@switch# docker run -dit --restart unless-stopped alpine
```

Docker ストレージバックエンドのサイズ変更が失敗する

問題: Docker バックエンドストレージのサイズを変更しようとして失敗しました。

考えられる原因: ゲスト シェルが無効になっていない可能性があります。

解決策: 次のコマンドを使用して、ゲスト シェルが無効になっているかどうかを確認します。

```
root@switch# losetup -a | grep dockerpart
root@n9k-2#
```

ゲスト シェルが無効になっている場合、コマンドは出力を表示しません。

必要に応じて、次のコマンドを入力してゲスト シェルを無効にします。

```
switch# guestshell disable
```

それでも Docker バックエンドストレージのサイズを変更できない場合は、`/bootflash/dockerpart` を削除し、`/etc/sysconfig/docker` の `[small_]large_dockerstrg` を調整してから、Docker を再度起動して、必要なサイズの新しい Docker パーティションを取得します。

Docker コンテナがポートで着信トラフィックを受信しない

問題: Docker コンテナがポートで着信トラフィックを受信しません。

考えられる原因: Docker コンテナが `kstack` ポートではなく `netstack` ポートを使用している可能性があります。

解決策: Docker コンテナによって使用されるエフェメラルポートが `kstack` の範囲内にあることを確認します。そうしないと、着信パケットがサービスのために `netstack` に送信され、ドロップされる可能性があります。

```

switch# show socket local-port-range
Kstack local port range (15001 - 58000)
Netstack local port range (58001 - 63535) and nat port range (63536 - 65535)
switch# conf t
Enter configuration commands, one per line. End with CNTL/Z.
switch(config)# sockets local-port-range <start_port> <end_port>
switch# run bash sudo su -
root@switch# cat /proc/sys/net/ipv4/ip_local_port_range
15001 58000
root@switch#

```

Docker コンテナでデータ ポートと / または管理インターフェイスを表示できません

問題: Docker コンテナにデータ ポートまたは管理インターフェイスが表示されません。

解決方法:

- `-v /var/run/netns:/var/run/netns:ro,rslave --network host` オプションを使用して、すべてのホスト名前空間がマップされたホストネットワーク名前空間で Docker コンテナが開始されていることを確認します。
- コンテナに入ると、デフォルトで管理ネットワークの名前空間に入ります。 `ip netns` ユーティリティを使用して、データポートインターフェイスを持つデフォルト (`init`) ネットワーク名前空間に移動できます。 `ip netns` ユーティリティは、`dnf`、`apk` などを使用してコンテナにインストールする必要がある場合があります。

一般的なトラブルシューティングのヒント

[問題: (Problem:)] 他のトラブルシューティングプロセスを使用しても解決されなかった Docker コンテナに関する他の問題があります。

解決方法:

- `/var/log/docker` で `dockerd` デバッグ出力を探して、何が問題なのかの手掛かりを見つけてください。
- スイッチに 8 GB 以上の RAM があることを確認します。 Docker 機能は、RAM が 8 GB 未満のスイッチではサポートされていません。



第 13 章

NX-SDK

この章は、次の項で構成されています。

- [NX-SDK について \(137 ページ\)](#)
- [オンボックス \(ローカル\) アプリケーションについて \(139 ページ\)](#)
- [デフォルト Docker イメージ \(139 ページ\)](#)
- [NX-SDK に関する注意事項と制限事項 \(139 ページ\)](#)
- [オフボックス \(リモート\) アプリケーション \(140 ページ\)](#)

NX-SDK について

Cisco NX-OS SDK (NX-SDK) は、自動化およびカスタム アプリケーションの作成 (カスタムの生成など) のためのインフラストラクチャへのアクセスを合理化する C++ 抽象化およびプラグイン ライブラリ レイヤです。

- CLI
- Syslog
- イベント マネージャとエラー マネージャ
- アプリケーション間通信
- ハイ アベイラビリティ (HA)
- ルート マネージャ

NX-SDK を使用したアプリケーション開発には、C++、Python、または Go (一部の例外あり) を使用できます。

要件

NX-SDK では次の要件があります。

- Docker

- Linux 環境（Ubuntu 14.04 または Centos 6.7 のいずれか）。提供されている NX-SDK Docker コンテナを使用することをお勧めします。詳細については、「[Cisco DevNet NX-SDK](#)」を参照してください。

ローカル（オン スイッチ）およびリモート（オフ スイッチ）アプリケーションのサポート

NX-SDK を使用して開発されたアプリケーションは、NX-SDK が提供する Docker コンテナのスイッチから作成または開発されます。アプリケーションの作成後、アプリケーションを展開できる場所を柔軟に選択できます。

- ローカル（オンボックス）アプリケーションはスイッチ上で実行されます。詳細については、[オンボックス（ローカル）アプリケーションについて（139 ページ）](#)を参照してください。
- リモート（オフボックス）アプリケーションは、スイッチをオフにして実行されます。このオプションは NX-SDK 2.0 以降でサポートされており、アプリケーションをスイッチ以外の場所で実行するようにデプロイできます。詳細については、[リモート アプリケーションについて（141 ページ）](#)を参照してください。

関連情報

Cisco NX-SDK の詳細については、次にアクセスしてください。

- [Cisco DevNet NX-SDK](#)。versions.md リンク（[Cisco DevNet NX-SDK バージョン](#)）をクリックして、サポートされている各リリースの機能と詳細に関する情報を取得します。
- [NX-SDK Readme](#)

必要に応じて、シスコは NX-SDK の情報を github に追加します。

Go バインディングに関する考慮事項

Go バインディングは、NX-SDK のリリースと、アプリがローカルで実行されているかリモートで実行されているかに応じて、さまざまなレベルでサポートされます。

- NX-SDK リモート アプリケーションのすべてのバージョンの Go バインディングは、EFT 前の品質です。
- ローカル NX-SDK 2.0 アプリケーションの Go バインディングは、EFT 前です。
- ローカル NX-SDK 1.7.5 以前のアプリケーションの Go バインディングがサポートされていません。

詳細については、「[NX-SDK アプリケーションの GO バインディング](#)」を参照してください。

オンボックス（ローカル）アプリケーションについて

オンボックス（ローカル）アプリケーションでは、NX-SDK をインストールし、選択したサポート対象言語でアプリケーションをビルドし、スイッチにインストールできる .rpm ファイルとしてアプリをパッケージ化し、スイッチにアプリケーションをインストールして実行します。。.rpm ファイルは手動で生成することも、自動生成することもできます。

アプリケーション開発は、NX-SDK が提供するコンテナで行われます。ローカルアプリケーションには、リモートアプリケーションとは異なるコンテナとツールを使用します。詳細については、[デフォルト Docker イメージ（139 ページ）](#)を参照してください。

ローカルアプリケーションのビルド、インストール、および実行については、[Cisco DevNet NX-SDK](#)を参照してください。

デフォルト Docker イメージ

NX-SDK には、ローカルまたはリモートで使用するための次の Docker イメージとツールがデフォルトに含まれています。

使用方法	目次
スイッチの場合	Cisco ENXOS SDK クロス コンパイル用の Wind River Linux（WRL） ツール チェーン 多言語バインディング ツールキット NX-SDK 1.75 以降、Go コンパイラ
オフ スイッチ（リモート）	事前にビルドされた libnxsdk.so を備えた NX-SDK 多言語バインディング ツールキット Go コンパイラ RapidJSON リモート API サポートのための gRPC

詳細については、<https://github.com/CiscoDevNet/NX-SDK#readme>https://github.com/CiscoDevNet/NX-SDK/tree/master/readmes/nxsdk_docker_images.mdを参照してください。

NX-SDK に関する注意事項と制限事項

NX-SDK には、アプリケーションをローカル（オンボックス）またはリモート（オフボックス）で実行するための使用上の注意事項と制限事項があります。

注意事項と制限事項については、[Cisco DevNet NX-SDK](#) の「役立つメモ」を参照してください。

- リモートアプリケーションの場合、SDK サーバーに接続するポートは 50002 ～ 50100 です。これらのポートが開いていて、他のサービスによって使用されていないことを確認してください。ポート 50051 は、内部アプリケーションによる使用がブロックされており、リモートアプリケーションでは使用できません。
- NX-SDK を使用したカスタム アプリケーションの開発には、次の制限が適用されます。
 - GNU C/C++ ツールチェーンと標準ライブラリを備えた現在の Linux 配布を使用してアプリケーションを開発する必要があります。
 - 提供されている Docker イメージでアプリケーションを開発することをお勧めします。Docker イメージと関連ツールを実行するには、最低 8 MB の空きメモリと 64 ビット ホストが必要です。「[デフォルト Docker イメージ（139 ページ）](#)」を参照してください。
- Cisco は、Bash でアプリケーションをビルドしてテストすることを推奨しています。それらを本番環境に展開する場合は、アプリケーションを RPM としてビルドし、NX-OS VSH（Vshell）で実行します。
- Cisco Nexus スイッチは、Bash および VSH で最大 32 のアプリケーションをサポートします。
- NX-SDK 2.0 の場合、Cisco Nexus スイッチで実行されているリモート NX-SDK アプリケーションおよび NX-SDK サーバーには 1 対 1 の制限があります。
- 最大 10 個のリモート NX-SDK アプリケーションをスイッチで同時に実行できます。
- NX-SDK サーバーは、CLI で構成されたアプリケーションからのリモート要求のみを受け入れ、ランダムなアプリケーションからは受け入れません。
- エラーが発生すると、SDK サーバーは例外をスローします。
- 一部の API は、リモートアプリケーションでは実行されません。詳細については、API ドキュメントを参照してください。
- リモートクライアントライブラリコードはオープンソースではないため、NX-SDK リモートアプリは NX-SDK リモート Docker コンテナで実行する必要があります。自分の OS でリモート NX-SDK を使用するには、使用している OS とコンパイラのバージョンを github でリクエストします。Cisco は、バージョンに基づいてリモート libnxsdk.so を生成できます。

オフボックス（リモート）アプリケーション

NX-SDK2.0 について

NX-SDK バージョン 2.0 は、開発者が必要な場所でアプリケーションを実行できる実行環境の柔軟性を可能にします。このバージョンの NX-SDK では、アプリケーションは引き続きコンテナ内のスイッチをオフにして開発されますが、たとえばクラウドなどで、アプリケーションをスイッチ上またはスイッチ外のいずれかで実行できます。

NX-SDK 2.0 には次のような利点があります。

- スイッチをお客様の環境に簡単に統合できます。
- スイッチがデータセンター、パブリッククラウド、プライベートクラウドでシームレスに動作できるようにする拡張性。
- スイッチ レベルのリソースでの変更がアプリケーションの変更または書き換えを必要としないように、スイッチ リソースから顧客アプリを切り離します。
- アプリケーションがリンクするための使いやすい API を備えた単一のライブラリ。これにより、スイッチの相互作用が簡素化され、アプリケーションをより簡単に記述およびデバッグできる高水準言語で記述できます。
- リモート サービスの実行は、オンボックス アプリケーションよりも安全です。

詳細については、https://github.com/CiscoDevNet/NX-SDK/blob/master/readmes/NXSDK_in_NXOS.md[NX-SDK アプリケーションのリモート展開（Deploying NX-SDK Applications Remotely）] を参照してください。

リモート アプリケーションについて

リモート アプリケーションは、Cisco Nexus スイッチではない別のスイッチに置くことができます。リモートまたはオフボックスのアプリケーションは、NX-SDK レイヤを介して呼び出し、スイッチと対話して情報の読み取り（取得）または情報の書き込み（設定）をします。

ローカルとリモートの両方の NX-SDK アプリケーションは同じ API を使用するため、NX-SDK アプリケーションをオンボックスまたはオフボックスで柔軟に展開できます。

リモートで実行するには、アプリケーションが特定の要件を満たしている必要があります。詳細については、https://github.com/CiscoDevNet/NX-SDK/blob/master/readmes/NXSDK_in_NXOS.md[NX-SDK アプリケーションのリモート展開（Deploying NX-SDK Applications Remotely）] を参照してください。

2.0 より前の NX-SDK アプリケーションの後方互換性

NX-SDK 2.0 には、NX-SDK v1.75 アプリケーションの開発方法に応じて、条件付きの後方互換性があります。

- 通常、NX-SDK は、NX-SDK 2.0 より前に作成したアプリのリモート実行をサポートしており、アプリを完全に書き直す必要はありません。代わりに、API 呼び出しを変更するために変更せずに、同じアプリを再利用できます。新しい NX-SDK 2.0 モデルで古いアプリをサポートするには、API 呼び出しで IP およびポートのパラメータを提供する必要があります。これらのパラメータは NX-SDK 1.75 以前では使用できませんが、アプリが SDK サーバにエクスポートできる環境変数として IP アドレスとポート情報を追加できます。
- ただし、NX-SDK 2.0 より前のアプリの下位互換性がサポートされていない場合があります。古いアプリの一部の API は、リモートでの実行をサポートしていないか、実行できない可能性があります。この場合、API は例外をスローできます。元のアプリケーションに対する例外処理の完全性と堅牢性によっては、アプリケーションが予期しない動作をする可能性があります。最悪の場合、クラッシュする可能性があります。

詳細については、https://github.com/CiscoDevNet/NX-SDK/blob/master/readmes/NXSDK_in_NXOS.md[NX-SDK アプリケーションのリモート展開（Deploying NX-SDK Applications Remotely）] を参照してください。

NX-SDK セキュリティ

NX-OS 9.3（1）以降、NX-SDK 2.0 は次のセキュリティ機能をサポートしています。

- セッションセキュリティ。リモートアプリケーションは、トランスポート レイヤー サービス（TLS）を介してスイッチ上の NX SDK サーバに接続し、アプリケーションとスイッチの NX SDK サーバ間に暗号化されたセッションを提供できます。
- サーバ証明書のセキュリティ。Cisco NX-OS 9.3（1）を使用した新しいスイッチ展開の場合、NX-SDK サーバは 1 日限りの一時証明書を生成して、カスタム証明書をインストールするのに十分な時間を提供します。

たとえば、以前の NX-SDK バージョンから NX-SDK 2.0 にアップグレードする場合など、NX-SDK サーバにカスタム証明書がすでにインストールされている場合、既存の証明書はアップグレード後も保持され、使用されます。
- API 書き込み呼び出し制御。NX-SDK 2.0 では、セキュリティプロファイルが導入されています。これにより、アプリケーションが NX-SDK サーバをどの程度制御できるかを制御するための事前定義されたポリシーを選択できます。セキュリティプロファイルに関する詳細情報を入手するには、[NX SDK 2.0 のセキュリティ プロファイル（142 ページ）](#) を参照します。

NX SDK 2.0 のセキュリティ プロファイル

以前のリリースでは、SDK バージョン 1.75 の API は、イベントのデータの読み取りと取得のみが許可されていました。Cisco NX-OS リリース 9.3(1) 以降、NX-SDK 2.0 は書き込みコールを含むさまざまなタイプの操作をサポートします。

アプリがスイッチを読み書きする機能は、セキュリティプロファイルを介して制御できます。セキュリティプロファイルは、スイッチで実行されているアプリケーションのサービスに付加されるオプションのオブジェクトです。セキュリティプロファイルは、スイッチに書き込むアプリケーションの機能を制御し、スイッチ機能を変更、削除、または構成するアプリケーションの機能を制御します。デフォルトでは、アプリケーションの書き込みは許可されていないため、アプリケーションごとに、スイッチへの書き込みアクセスを有効にするセキュリティプロファイルを作成する必要があります。

Cisco の NX-SDK は、次のセキュリティプロファイルを提供します。

プロファイル	説明	値
拒否	CLI の追加を除き、API 呼び出しがスイッチに書き込まれないようにします。	これはデフォルトプロファイルです。

プロファイル	説明	値
スロットル	スイッチを変更する API を許可しますが、指定された数の呼び出しまでのみ許可します。このセキュリティプロファイルは、スロットリングを適用して API 呼び出しの数を制御します。 アプリケーションは制限まで書き込むことができますが、制限を超えると書き込みが停止し、応答でエラーメッセージが送信されます。	スロットルは 50 回の API 呼び出しであり、スロットルは 5 秒後にリセットされます。
許可	スイッチを変更する API は制限なしで許可されます	

NX-SDK のセキュリティプロファイルの詳細については、[\[NX-SDK アプリケーションのセキュリティプロファイル \(Security Profiles for NX-SDK Applications\)\]](#) を参照してください。

アプリケーションの構築、インストール、および実行の詳細については、[CiscoDevNet NX-SDK](#) にアクセスしてください。



第 14 章

NX-API

- [NX-APIについて（145 ページ）](#)
- [NX-API の使用（148 ページ）](#)
- [XML および JSON でサポートされたコマンド（165 ページ）](#)
- [JSON-RPC、JSON、および XML でサポートされるコマンド（174 ページ）](#)

NX-APIについて

Cisco Nexus スイッチでは、コマンドライン インターフェイス（CLI）はスイッチ上でのみ実行されます。NX-APIはHTTP/HTTPSを使ってスイッチの外部でCLIを使用できるようにすることで、これらの CLI のユーザー補助を改善します。この拡張機能を Cisco Nexus 7000 シリーズ スイッチの既存の Cisco NX-OS CLI システムに使用できます。NX-API は **[表示（NXAPI）]** コマンドと構成をサポートします。

NX-API は、JSON-RPC、JSON、および XML フォーマットをサポートします。

Cisco Nexus スイッチでは、コマンドライン インターフェイス（CLI）はスイッチ上でのみ実行されます。NX-APIはHTTP/HTTPSを使ってスイッチの外部でCLIを使用できるようにすることで、これらの CLI のユーザー補助を改善します。この拡張機能を Cisco Nexus 9000 シリーズ スイッチの既存の Cisco NX-OS CLI システムに使用できます。NX-API は **show** コマンド、構成と Linux Bash をサポートします。

NX-API は JSON-RPC をサポートしています。

Cisco Nexus スイッチでは、コマンドライン インターフェイス（CLI）はスイッチ上でのみ実行されます。NX-API は HTTP / HTTPS を使ってスイッチの外部で CLI を使用できるようにすることで、これらの CLI のユーザー補助を改善します。この拡張機能を Cisco Nexus 3500 プラットフォーム スイッチの既存の Cisco NX-OS CLI システムに使用できます。NX-API は **show** コマンド、構成と Linux Bash をサポートします。

NX-API は、JSON-RPC、JSON、および XML フォーマットをサポートします。

機能 NX-API

- サンドボックスを介してデバイスにアクセスするには、機能 NX-API を有効にする必要があります。
- デバイス上の `|json` は、内部的に Python スクリプトを使用して出力を生成します。
- NX-API は、ipv4 を介して http / https のいずれかで有効にすることができます。

```
BLR-VXLAN-NPT-CR-179# show nxapi
nxapi enabled
HTTP Listen on port 80
HTTPS Listen on port 443
BLR-VXLAN-NPT-CR-179#
```

- NX-API は、サードパーティの NGINX プロセスを内部的に生成しています。このプロセスは、ハンドラ受信 / 送信 / http 要求 の処理 / 応答:

```
nxapi certificate {https crt | https key}
nxapi certificate enable
```

- NX-API 証明書は https で有効にできます
- nginx が動作するデフォルトのポートは、http / https がそれぞれ 80 / 443 です。次の CLI コマンドを使用して変更することもできます:

```
nxapi {http|https} port port-number
```

転送

NX-APIは、転送のように HTTP または HTTPS を使用します。CLI は、HTTP / HTTPS POST 本文にエンコードされます。

NX-API バックエンドは Nginx HTTP サーバを使用します。Nginx プロセスとそのすべての子プロセスは、CPU とメモリの使用量が制限されている Linux cgroup 保護下にあります。Nginx のメモリ使用量が cgroup の制限を超えると、Nginx プロセスが再起動されて復元されます。



(注) [機能なし NXAPI (no feature NXAPI)] コマンドを使用して NX-API を無効にした後でも、Nginx プロセスは引き続き実行されます。これは、他の管理関連プロセスに必要です。

メッセージ形式

NX-API は、XML 出力をサポートする Cisco Nexus 7000 シリーズ CLI システムの拡張機能です。NX-API は、特定のコマンドの JSON 出力フォーマットもサポートしています。

NX-API は、XML 出力をサポートする Cisco Nexus 9000 シリーズ CLI システムの拡張機能です。NX-API は、特定のコマンドの JSON 出力フォーマットもサポートしています。



- (注)
- NX-API XML 出力は、情報を使いやすいフォーマットで表示します。
 - NX-API XML は、Cisco NX-OS NETCONF 導入に直接マッピングされません。
 - NX-API XML 出力は、JSON または JSON-RPC に変換できます。

セキュリティ

NX-API は HTTPS をサポートします。HTTPS を使用すると、デバイスへのすべての通信が暗号化されます。

NX-API は、デバイスの認証システムに統合されています。ユーザーは、NX-API を介してデバイスにアクセスするための適切なアカウントを持っている必要があります。NX-API では HTTP basic 認証が使用されます。すべてのリクエストには、HTTP ヘッダーにユーザー名とパスワードが含まれている必要があります。



- (注) ユーザーのログイン資格情報を保護するには、HTTPS の使用を検討する必要があります。

[機能 (feature)] マネージャ CLI コマンドを使用して、NX-API を有効にすることができます。NX-API はデフォルトで無効になっています。

NX-API は、ユーザーが最初に認証に成功したときに、セッションベースの Cookie、**nxapi_auth** を提供します。セッション Cookie を使用すると、デバイスに送信される後続のすべての NX-API 要求にユーザー名とパスワードが含まれます。ユーザー名とパスワードは、完全な認証プロセスの再実行をバイパスするために、セッション Cookie で使用されます。セッション Cookie が後続の要求に含まれていない場合は、別のセッション Cookie が必要であり、認証プロセスによって提供されます。認証プロセスの不必要な使用を避けることで、デバイスのワークロードを軽減できます。



- (注) **nxapi_auth** cookie は 600 秒 (10 分) で期限切れになります。この値は固定されており、調整できません。

Cookie の有効期限が切れたら、ユーザー名 / パスワードを再送信する必要があります。



- (注) NX-API は、スイッチ上の Programmable Authentication Module (PAM) を使用して認証を行います。cookie を使用して PAM の認証数を減らし、PAM の負荷を減らします。

NX-API の使用

Cisco Nexus 7000 シリーズ スイッチのコマンド、コマンドタイプ、および出力タイプは、CLI を HTTP/HTTPS POST の本文にエンコードすることにより、NX-API を使用して入力されます。要求に対する応答は、XML、JSON、または JSON-RPC 出力形式で返されます。

Cisco Nexus 9000 シリーズ スイッチのコマンド、コマンドタイプ、および出力タイプは、CLI を HTTP/HTTPS POST の本文にエンコードすることにより、NX-API を使用して入力されます。要求に対する応答は、XML または JSON 出力形式で返されます。

Cisco Nexus 3500 プラットフォーム スイッチのコマンド、コマンドタイプ、および出力タイプは、CLI を HTTP/HTTPS POST の本文にエンコードすることにより、NX-API を使用して入力されます。要求に対する応答は、XML、JSON、または JSON-RPC 出力形式で返されます。



(注) NX-API 応答コードの詳細については、「[NX-API レスポンス コードの表](#)」を参照してください。

デバイスで **feature manager** CLI コマンドを使用して NX-API を有効にする必要があります。デフォルトでは、NX-API は無効になっています。

次の例は、NX-API サンドボックスを設定して起動する方法を示しています。

- 管理インターフェイスを有効にします。

```
switch# conf t
switch(config)# interface mgmt 0
switch(config)# ip address 198.51.100.1/24
switch(config)# vrf context management
switch(config)# ip route 203.0.113.1/0 1.2.3.1
```

- NX-API **nxapi** 機能を有効にします。

```
switch# conf t
switch(config)# feature nxapi
```

次の例は、リクエストとそのレスポンスを XML 形式で示しています。

要求:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ins_api>
  <version>0.1</version>
  <type>cli_show</type>
  <chunk>0</chunk>
  <sid>session1</sid>
  <input>show switchname</input>
  <output_format>xml</output_format>
</ins_api>
```

応答:

```
<?xml version="1.0"?>
<ins_api>
  <type>cli_show</type>
```



```

<version>0.1</version>
<sid>eoc</sid>
<outputs>
  <output>
    <body>
      <hostname>switch</hostname>
    </body>
    <input>show switchname</input>
    <msg>Success</msg>
    <code>200</code>
  </output>
</outputs>
</ins_api>

```

次の例は、JSON 形式の要求とその応答を示しています。

要求:

```

{
  "ins_api": {
    "version": "0.1",
    "type": "cli_show",
    "chunk": "0",
    "sid": "session1",
    "input": "show switchname",
    "output_format": "json"
  }
}

```

応答:

```

{
  "ins_api": {
    "type": "cli_show",
    "version": "0.1",
    "sid": "eoc",
    "outputs": {
      "output": {
        "body": {
          "hostname": "switch"
        },
        "input": "show switchname",
        "msg": "Success",
        "code": "200"
      }
    }
  }
}

```

次の例は、リクエストとそのレスポンスを XML 形式で示しています。

要求:

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<ins_api>
  <version>0.1</version>
  <type>cli_show</type>
  <chunk>0</chunk>
  <sid>session1</sid>
  <input>show switchname</input>
  <output_format>xml</output_format>

```

```
</ins_api>
```

応答:

```
<?xml version="1.0"?>
<ins_api>
  <type>cli_show</type>
  <version>0.1</version>
  <sid>eoc</sid>
  <outputs>
    <output>
      <body>
        <hostname>switch</hostname>
      </body>
      <input>show switchname</input>
      <msg>Success</msg>
      <code>200</code>
    </output>
  </outputs>
</ins_api>
```

次の例は、JSON 形式の要求とその応答を示しています。

要求:

```
{
  "ins_api": {
    "version": "0.1",
    "type": "cli_show",
    "chunk": "0",
    "sid": "session1",
    "input": "show switchname",
    "output_format": "json"
  }
}
```

応答:

```
{
  "ins_api": {
    "type": "cli_show",
    "version": "0.1",
    "sid": "eoc",
    "outputs": {
      "output": {
        "body": {
          "hostname": "switch"
        },
        "input": "show switchname",
        "msg": "Success",
        "code": "200"
      }
    }
  }
}
```

NX-API コールの管理インターフェイスの使用

NX-API コールには管理インターフェイスを使用することをお勧めします。

NX-API の非管理インターフェイスとカスタム ポートを使用する場合、NX-API トラフィックが API トラフィックを好ましくない処理する可能性のあるデフォルトの **copp** エントリにヒットしないように、CoPP ポリシーにエントリを作成する必要があります。



- (注) NX-API トラフィックには管理インターフェイスを使用することをお勧めします。それが不可能で、カスタム ポートが使用されている場合は、「**copp-http**」クラスを更新して、カスタム NX-API ポートを含める必要があります。

次の例のポート 9443 は、NX-API トラフィックに使用されています。

このポートは、copp-system-acl-http ACL に追加され、copp-http クラスの下で一致できるようになり、100 pps ポリシングになります。（特定の環境では、これを増やす必要がある場合があります。）

```
!
ip access-list copp-system-acl-http
  10 permit tcp any any eq www
  20 permit tcp any any eq 443
  30 permit tcp any any eq 9443 <-----
!
class-map type control-plane match-any copp-http
  match access-group name copp-system-acl-http
!
!
policy-map type control-plane copp-system-policy
  class copp-http
    police pps 100
!
```

NX-API 管理コマンド

次の表にリストされている CLI コマンドを使用して、NX-API を有効にして管理できます。

表 7: NX-API 管理コマンド

NX-API 管理コマンド	説明
feature nxapi	NX-API を有効化します。
no feature nxapi	NX-API を無効化します。
nxapi {http https} port port	ポートを指定します。
no nxapi {http https}	HTTP / HTTPS を無効化します。
show nxapi	ポート情報を表示します。

NX-API 管理コマンド	説明
nxapi certificate {https crt certfile httpskey keyfile} filename	次のアップロードを指定します: • https crt が指定されている場合の HTTPS 証明書。 • httpskey が指定されている場合の HTTPS キー。 HTTPS 証明書の例: <pre>nxapi certificate https crt certfile bootflash:cert.crt</pre> HTTPS キーの例: <pre>nxapi certificate httpskey keyfile bootflash:privkey.key</pre>
nxapi certificate enable	証明書を有効化します。

以下は、HTTPS 証明書の正常なアップロードの例です:

```
switch(config)# nxapi certificate https crt certfile certificate.crt
Upload done. Please enable. Note cert and key must match.
switch(config)# nxapi certificate enable
switch(config)#
```

以下は、HTTPS キーの正常なアップロードの例です:

```
switch(config)# nxapi certificate httpskey keyfile bootflash:privkey.key
Upload done. Please enable. Note cert and key must match.
switch(config)# nxapi certificate enable
switch(config)#
```

状況によっては、証明書が無効であることを示すエラー メッセージが表示されることがあります:

```
switch(config)# nxapi certificate httpskey keyfile bootflash:privkey.key
Upload done. Please enable. Note cert and key must match.
switch(config)# nxapi certificate enable
Nginx certificate invalid.
switch(config)#
```

これは、キー ファイルが暗号化されている場合に発生する可能性があります。その場合、キー ファイルをインストールする前に復号化する必要があります。次の例に示すように、Guest Shell に移動してキー ファイルを復号化する必要がある場合があります。

```
switch(config)# guestshell
[b3456@guestshell ~]$
[b3456@guestshell bootflash]$ /bin/openssl rsa -in certfilename.net.pem -out clearkey.pem

Enter pass phrase for certfilename.net.pem:
writing RSA key
[b3456@guestshell bootflash]$
[b3456@guestshell bootflash]$ exit
switch(config)#
```

Guest Shell の詳細については、[ゲスト シェル \(11 ページ\)](#) を参照します。

これが問題の原因である場合、証明書を正常にインストールできるはずです。

```
switch(config)# nxapi certificate httpskey keyfile bootflash:privkey.key
Upload done. Please enable. Note cert and key must match.
switch(config)# nxapi certificate enable
switch(config)#
```

NX-API を使用したインタラクティブ コマンドの操作

対話型コマンドの確認プロンプトを無効にし、エラー コード 500 によるタイムアウトを回避するには、対話型コマンドの前に**[端末の dont-ask (terminal dont-ask)]**を追加します。を使用。複数の対話型コマンドを区切るには、それぞれが。は単一のブランク文字で囲まれています。

エラー コード 500 でのタイムアウトを回避するために端末の **dont-ask** を使用する対話型コマンドの例をいくつか次に示します:

```
terminal dont-ask ; reload module 21
terminal dont-ask ; system mode maintenance
```

NX-API リクエスト要素

NX-API リクエスト要素は、XML 形式、JSON 形式、または JSON-RPC 形式でスイッチに送信されます。リクエストの HTTP ヘッダーは、リクエストのコンテンツ タイプを識別する必要があります。

入力リクエストの形式が XML または JSON の場合、次の表にリストされている NX-API 要素を使用して、CLI コマンドを指定します。

表 8: NX-API リクエスト要素

NX-API リクエスト要素	説明
version	NX-API バージョンを指定します。

NX-API リクエスト要素	説明
<i>type</i>	実行するコマンドのタイプを指定します。 次のタイプのコマンドがサポートされています。 • cli_show 構造化された出力が必要な CLI show コマンド。コマンドが XML 出力をサポートしていない場合は、エラーメッセージが返されます。 • cli_show_ascii ASCII 出力が必要な CLI show コマンド。これは、ASCII 出力を解析する既存のスクリプトと一致します。ユーザーは、最小限の変更で既存のスクリプトを使用できます。 • cli_conf CLI 構成コマンド • bash Bash コマンド。ほとんどの非対話型 Bash コマンドは、NX-API でサポートされています。 (注) • 各コマンドは、現在のユーザーの権限でのみ実行可能です。 • メッセージタイプが ASCII の場合、出力でパイプ操作がサポートされます。出力が XML 形式の場合、パイプ操作はサポートされていません。 • 最大 10 個の連続する show コマンドがサポートされています。show コマンドの数が 10 個を超える場合、11 番目以降のコマンドは無視されます。 • 対話型コマンドはサポートされていません。

NX-API リクエスト要素	説明				
チャンク	一部の show コマンドは、大量の出力を返す場合があります。コマンド全体が完了する前に NX-API クライアントが出力の処理を開始するために、NX-API は show コマンドの出力チャンクをサポートしています。 次の設定を有効または無効にできます。 <table border="1"> <tr> <td>0</td><td>チャンク出力しません。</td></tr> <tr> <td>1</td><td>チャンク出力。</td></tr> </table> (注) チャンクをサポートするのは show コマンドだけです。一連の show コマンドが入力されると、最初のコマンドだけがチャンクされて返されます。 出力メッセージの形式は XML です。(XML がデフォルトです。) < または > などの特殊文字は、有効な XML メッセージを形成するために変換されます (< は < に変換されます > は &gt; に変換されます)。 XML SAX を使用して、チャンクされた出力を解析できます。 (注) チャンクが有効な場合、メッセージ形式は XML に制限されます。チャンクが有効な場合、JSON 出力形式はサポートされません。 (注) チャンクが有効になっている場合、現在サポートされている最大メッセージサイズは、チャンク出力の 200MB です。	0	チャンク出力しません。	1	チャンク出力。
0	チャンク出力しません。				
1	チャンク出力。				
sid	セッション ID 要素は、応答メッセージがチャンクされている場合にのみ有効です。メッセージの次のチャンクを取得するには、前の応答メッセージの <i>sid</i> と一致する <i>sid</i> を指定する必要があります。 Cisco NX-OS リリース 9.3(1) では、<i>sid</i> オプション <i>clear</i> が導入されています。<i>sid</i> を <i>clear</i> に設定して新しいチャンク リクエストが開始されると、現在のチャンク リクエストはすべて破棄または破棄されます。 応答コード 429 を受け取った場合: 同時チャンク リクエストの最大数は 2 です。<i>sid clear</i> を使用して、現在のチャンク リクエストを破棄します。<i>sid clear</i> を使用した後、後続の応答コードは、残りのリクエストに対して通常どおり動作します。				

NX-API リクエスト要素	説明						
<i>input</i>	入力1つのコマンドまたは複数のコマンドです。ただし、異なるメッセージタイプに属するコマンドを混在させてはなりません。たとえば、show コマンドは cli_show メッセージタイプであり、cli_conf モードではサポートされません。 (注) bash を除き、複数のコマンドは「;」で区切ります。(; は、単一のブランク文字で囲む必要があります。) bash の場合、複数のコマンドは「;」で区切ります。(; は単一のブランク文字で囲まれていません。) 以下は、複数のコマンドの例です。 <table> <tr> <td>cli_show</td><td>show version ; show interface brief ; show vlan</td></tr> <tr> <td>cli_conf</td><td>interface Eth4/1 ; no shut ; switchport</td></tr> <tr> <td>bash</td><td>cd /bootflash;mkdir new_dir</td></tr> </table>	cli_show	show version ; show interface brief ; show vlan	cli_conf	interface Eth4/1 ; no shut ; switchport	bash	cd /bootflash;mkdir new_dir
cli_show	show version ; show interface brief ; show vlan						
cli_conf	interface Eth4/1 ; no shut ; switchport						
bash	cd /bootflash;mkdir new_dir						
<i>output_format</i>	使用可能な出力メッセージ形式は次のとおりです。 <table> <tr> <td>xml</td><td>XML 形式を指定します。</td></tr> <tr> <td>json</td><td>JSON 形式で出力を指定します。</td></tr> </table> (注) Cisco Nexus 7000 シリーズ CLI は XML 出力をサポートしています。つまり、JSON 出力は XML から変換されます。変換はスイッチで処理されます。 計算のオーバーヘッドを管理するために、JSON 出力は出力の量によって決定されます。出力が 1MB を超える場合、出力は XML 形式で返されます。出力がチャンクされている場合、XML 出力のみがサポートされます。 HTTP/HTTPS ヘッダーの content-type ヘッダーは、応答形式 (XML または JSON) のタイプを示します。	xml	XML 形式を指定します。	json	JSON 形式で出力を指定します。		
xml	XML 形式を指定します。						
json	JSON 形式で出力を指定します。						

JSON-RPC が入力リクエスト形式である場合、次の表にリストされている NX-API 要素を使用して、CLI コマンドを指定します。

表 9: NX-API リクエスト要素

NX-API リクエスト要素	説明
<i>jsonrpc</i>	JSON-RPC プロトコルのバージョンを指定する文字列。 バージョンは 2.0 である必要があります。
<i>method</i>	呼び出されるメソッドの名前を含む文字列。 NX-API は、次のいずれかをサポートします。 • <code>cli show</code> または構成コマンド • <code>cli_ascii show</code> または構成コマンド。フォーマットせずに出力
<i>params</i>	メソッドの呼び出し中に使用されるパラメータ値を保持する構造化された値。 以下が含まれている必要があります。 • <code>cmd</code> CLI コマンド • <code>version</code> NX-API リクエストのバージョン識別子
<i>id</i>	クライアントによって確立されるオプションの識別子。指定されている場合は、文字列、数値、または <code>null</code> 値を含む必要があります。通常、値は <code>null</code> であってはならず、数値に小数部は含まれません。ユーザーが <i>id</i> パラメータを指定しない場合、サーバーはリクエストが単なる通知であり、応答がないと見なします。たとえば、 <i>id</i> : 1

NX-API リクエスト要素は、XML 形式または JSON 形式でスイッチに送信されます。リクエストの HTTP ヘッダーは、リクエストのコンテンツ タイプを識別する必要があります。

次の表にリストされている NX-API 要素を使用して、CLI コマンドを指定します。

表 10: NX-API リクエスト要素

NX-API リクエスト要素	説明
<i>version</i>	NX-API バージョンを指定します。

NX-API リクエスト要素	説明
<i>type</i>	実行するコマンドのタイプを指定します。 次のタイプのコマンドがサポートされています。 • cli_show 構造化された出力が必要な CLI show コマンド。コマンドが XML 出力をサポートしていない場合は、エラーメッセージが返されます。 • cli_show_ascii ASCII 出力が必要な CLI show コマンド。これは、ASCII 出力を解析する既存のスクリプトと一致します。ユーザーは、最小限の変更で既存のスクリプトを使用できます。 • cli_conf CLI 構成コマンド • bash Bash コマンド。ほとんどの非対話型 Bash コマンドは、NX-API でサポートされています。 (注) • 各コマンドは、現在のユーザーの権限でのみ実行可能です。 • メッセージタイプが ASCII の場合、出力でパイプ操作がサポートされます。出力が XML 形式の場合、パイプ操作はサポートされていません。 • 最大 10 の連続する show コマンドがサポートされています。show コマンドの数が 10 を超える場合、11 番目以降のコマンドは無視されます。 • 対話型コマンドはサポートされていません。

NX-API リクエスト要素	説明				
チャンク	一部の show コマンドは、大量の出力を返す場合があります。コマンド全体が完了する前に NX-API クライアントが出力の処理を開始するために、NX-API は show コマンドの出力チャンクをサポートしています。 次の設定を有効または無効にできます。 <table border="1"> <tr> <td>0</td><td>出力をチャンクしません。</td></tr> <tr> <td>1</td><td>チャンク出力。</td></tr> </table> (注) • チャンクをサポートするのは show コマンドだけです。一連の show コマンドが入力されると、最初のコマンドだけがチャンクされて返されます。 • 出力メッセージ形式のオプションは、XML または JSON です。 • XML 出力メッセージ形式の場合<または>などの特殊文字は、有効な XML メッセージを形成するために変換されます (<は<に変換されます>は &gt;に変換されます)。 XML SAX を使用して、チャンクされた出力を解析できます。 • 出力メッセージ形式が JSON の場合、チャンクが連結されて有効な JSON オブジェクトが作成されます。	0	出力をチャンクしません。	1	チャンク出力。
0	出力をチャンクしません。				
1	チャンク出力。				
<i>sid</i>	セッション ID 要素は、応答メッセージがチャンクされている場合にのみ有効です。メッセージの次のチャンクを取得するには、前の応答メッセージの <i>sid</i> と一致する <i>sid</i> を指定する必要があります。 NX-OS リリース 9.3(1) では、<i>sid</i> オプション <code>clear</code> が導入されています。<i>sid</i> を <code>clear</code> に設定して新しいチャンク リクエストが開始されると、現在のチャンク リクエストはすべて破棄または破棄されます。 応答コード 429 を受け取った場合: 同時チャンク リクエストの最大数は 2 です。<i>sid clear</i> を使用して、現在のチャンク リクエストを破棄します。<i>sid clear</i> を使用した後、後続の応答コードは、残りのリクエストに対して通常どおり動作します。				

NX-API リクエスト要素	説明						
<i>input</i>	入力は1つのコマンドまたは複数のコマンドです。ただし、異なるメッセージタイプに属するコマンドを混在させてはなりません。たとえば、show コマンドは cli_show メッセージタイプであり、cli_conf モードではサポートされません。 (注) bash を除き、複数のコマンドは「;」で区切ります。(; は、単一のブランク文字で囲む必要があります。) bash の場合、複数のコマンドは「;」で区切ります。(; は単一のブランク文字で囲まれていません。) 以下は、複数のコマンドの例です。 <table> <tr> <td>cli_show</td><td>show version ; show interface brief ; show vlan</td></tr> <tr> <td>cli_conf</td><td>interface Eth4/1 ; no shut ; switchport</td></tr> <tr> <td>bash</td><td>cd /bootflash;mkdir new_dir</td></tr> </table>	cli_show	show version ; show interface brief ; show vlan	cli_conf	interface Eth4/1 ; no shut ; switchport	bash	cd /bootflash;mkdir new_dir
cli_show	show version ; show interface brief ; show vlan						
cli_conf	interface Eth4/1 ; no shut ; switchport						
bash	cd /bootflash;mkdir new_dir						
<i>output_format</i>	使用可能な出力メッセージ形式は次のとおりです。 <table> <tr> <td>xml</td><td>XML 形式を指定します。</td></tr> <tr> <td>json</td><td>JSON 形式で出力を指定します。</td></tr> </table> (注) Cisco Nexus 9000 シリーズ CLI は XML 出力をサポートしています。つまり、JSON 出力は XML から変換されます。変換はスイッチで処理されます。 計算のオーバーヘッドを管理するために、JSON 出力は出力の量によって決定されます。出力が1MBを超える場合、出力は XML 形式で返されます。出力がチャンクされている場合、XML 出力のみがサポートされます。 HTTP/HTTPS ヘッダーの content-type ヘッダーは、応答形式 (XML または JSON) のタイプを示します。	xml	XML 形式を指定します。	json	JSON 形式で出力を指定します。		
xml	XML 形式を指定します。						
json	JSON 形式で出力を指定します。						

NX-API リクエスト要素は、XML 形式または JSON 形式でスイッチに送信されます。リクエストの HTTP ヘッダーは、リクエストのコンテンツ タイプを識別する必要があります。

次の表にリストされている NX-API 要素を使用して、CLI コマンドを指定します。

表 11: NX-API リクエスト要素

NX-API リクエスト要素	説明
version	NX-API バージョンを指定します。
type	実行するコマンドのタイプを指定します。 次のタイプのコマンドがサポートされています。 • cli_show 構造化された出力が必要な CLI show コマンド。コマンドが XML 出力をサポートしていない場合は、エラーメッセージが返されます。 • cli_show_ascii ASCII 出力が必要な CLI show コマンド。これは、ASCII 出力を解析する既存のスクリプトと一致します。ユーザーは、最小限の変更で既存のスクリプトを使用できます。 • cli_conf CLI 構成コマンド • bash Bash コマンド。ほとんどの非対話型 Bash コマンドは、NX-API でサポートされています。 (注) • 各コマンドは、現在のユーザーの権限でのみ実行可能です。 • メッセージタイプが ASCII の場合、出力でパイプ操作がサポートされます。出力が XML 形式の場合、パイプ操作はサポートされていません。 • 最大 10 の連続する show コマンドがサポートされています。show コマンドの数が 10 を超える場合、11 番目以降のコマンドは無視されます。 • 対話型コマンドはサポートされていません。

NX-API リクエスト要素	説明				
チャンク	一部の show コマンドは、大量の出力を返す場合があります。コマンド全体が完了する前に NX-API クライアントが出力の処理を開始するために、NX-API は show コマンドの出力チャンクをサポートしています。 次の設定を有効または無効にできます。 <table border="1"> <tr> <td>0</td><td>チャンク出力しません。</td></tr> <tr> <td>1</td><td>チャンク出力。</td></tr> </table> (注) チャンクをサポートするのは show コマンドだけです。一連の show コマンドが入力されると、最初のコマンドだけがチャンクされて返されます。 出力メッセージの形式は XML です。(XML がデフォルトです。) < または > などの特殊文字は、有効な XML メッセージを形成するために変換されます(< は < に変換されます> は &gt; に変換されます)。 XML SAX を使用して、チャンクされた出力を解析できます。 (注) チャンクが有効な場合、メッセージ形式は XML に制限されます。チャンクが有効な場合、JSON 出力形式はサポートされません。	0	チャンク出力しません。	1	チャンク出力。
0	チャンク出力しません。				
1	チャンク出力。				
<i>sid</i>	セッション ID 要素は、応答メッセージがチャンクされている場合にのみ有効です。メッセージの次のチャンクを取得するには、前の応答メッセージの <i>sid</i> と一致する <i>sid</i> を指定する必要があります。				

NX-API リクエスト要素	説明						
<i>input</i>	入力には1つのコマンドまたは複数のコマンドです。ただし、異なるメッセージタイプに属するコマンドを混在させてはなりません。たとえば、show コマンドは cli_show メッセージタイプであり、cli_conf モードではサポートされません。 (注) bash を除き、複数のコマンドは「;」で区切ります。(; は、単一のブランク文字で囲む必要があります。) bash の場合、複数のコマンドは「;」で区切ります。(; は単一のブランク文字で囲まれていません。) 以下は、複数のコマンドの例です。 <table> <tr> <td>cli_show</td><td>show version ; show interface brief ; show vlan</td></tr> <tr> <td>cli_conf</td><td>interface Eth4/1 ; no shut ; switchport</td></tr> <tr> <td>bash</td><td>cd /bootflash;mkdir new_dir</td></tr> </table>	cli_show	show version ; show interface brief ; show vlan	cli_conf	interface Eth4/1 ; no shut ; switchport	bash	cd /bootflash;mkdir new_dir
cli_show	show version ; show interface brief ; show vlan						
cli_conf	interface Eth4/1 ; no shut ; switchport						
bash	cd /bootflash;mkdir new_dir						
<i>output_format</i>	使用可能な出力メッセージ形式は次のとおりです。 <table> <tr> <td>xml</td><td>XML 形式を指定します。</td></tr> <tr> <td>json</td><td>JSON 形式で出力を指定します。</td></tr> </table> (注) Cisco Nexus 3500 プラットフォーム シリーズ CLI は XML 出力をサポートしています。つまり、JSON 出力は XML から変換されます。変換はスイッチで処理されます。 計算のオーバーヘッドを管理するために、JSON 出力は出力の量によって決定されます。出力が1MBを超える場合、出力は XML 形式で返されます。出力がチャンクされている場合、XML 出力のみがサポートされます。 HTTP/HTTPS ヘッダーの content-type ヘッダーは、応答形式 (XML または JSON) のタイプを示します。	xml	XML 形式を指定します。	json	JSON 形式で出力を指定します。		
xml	XML 形式を指定します。						
json	JSON 形式で出力を指定します。						

NX-API 応答要素

CLI コマンドに応答する NX-API 要素を次の表に示します。

CLI コマンドに応答する NX-API 要素を次の表に示します：

入力要求が XML または JSON フォーマットの場合、応答には次のものが含まれます：

表 12: NX-API 応答要素

NX-API 応答要素	説明
version	NX-API バージョン。
type	実行するコマンドのタイプ。
sid	応答のセッション識別子。この要素は、応答メッセージがチャンクされている場合にのみ有効です。
outputs	すべてのコマンド出力を囲むタグ。 複数のコマンドが <code>cli_show</code> または <code>cli_show_ascii</code> にある場合、各コマンド出力は単一の出力タグで囲まれます。 メッセージタイプが <code>cli_conf</code> または <code>bash</code> の場合、 <code>cli_conf</code> および <code>bash</code> コマンドにはコンテキストが必要なため、すべてのコマンドに単一の出力タグがあります。
出力	単一のコマンド出力の出力を囲むタグ。 <code>cli_conf</code> と <code>bash</code> メッセージタイプの場合、この要素にはすべてのコマンドの出力が含まれます。
input	リクエストで指定された1つのコマンドを囲むタグ。この要素は、要求入力要素を適切な応答出力要素に関連付けるのに役立ちます。
本文	コマンド応答の本文。
コード	コマンドの実行から返された原因コード。 NX-API は、ハイパーテキスト転送プロトコル（HTTP）ステータスコードレジストリ（ http://www.iana.org/assignments/http-status-codes/http-status-codes.xhtml ）で説明されている標準規格の HTTP 原因コードを使用します。
msg	返された原因コードに関連付けられたエラーメッセージ。

JSON の概要（JavaScript オブジェクト表記）

JSON は、判読可能なデータのために設計された軽量テキストベースのオープンスタンダードで、XML の代替になります。JSON はもともと JavaScript から設計されましたが、言語に依存しないデータ形式です。JSON/CLI 実行は現在、Cisco Nexus 3500 プラットフォームスイッチでサポートされています。



（注） NX-API/JSON 機能は、Cisco Nexus 3500 プラットフォームスイッチで使用できるようになりました。

ほぼすべての最新のプログラミング言語で何らかの方法でサポートされている 2 つの主要なデータ構造は次のとおりです。

- 順序付きリスト :: 配列
- 順序付けられていないリスト（名前/値のペア） :: オブジェクト

show コマンドの JSON/JSON-RPC/XML 出力には、サンドボックス経由でアクセスすることもできます。

JSON に関する注意事項

JSON に関する注意事項は次のとおりです。

- **show run | xml** コマンドと **show run | json** コマンドはサポートされていません（NX-OS 6.0（2）A8（9）以前）。
- すべての JSON 出力は文字列として返されます（NX-OS 6.0（2）A8（9）以前）。
- JSON-RPC 整数は、文字列ではなく常に整数として出力されます（NX-OS 6.0（2）A8（9）以前）。

CLI の実行

Show_Command | json

コード例

```
BLR-VXLAN-NPT-CR-179# show cdp neighbors | json
{"TABLE_cdp_neighbor_brief_info": {"ROW_cdp_neighbor_brief_info": [{"ifindex": "83886080", "device_id": "SW-SPARSHA-SAVBU-F10", "intf_id": "mgmt0", "ttl": "148", "capability": ["switch", "IGMP_cnd_filtering"], "platform_id": "cisco WS-C2960S-48TS-L", "port_id": "GigabitEthernet1/0/24"}, {"ifindex": "436207616", "device_id": "BLR-VXLAN-NPT-CR-178(FOC1745R01W)", "intf_id": "Ethernet1/1", "ttl": "166", "capability": ["router", "switch", "IGMP_cnd_filtering", "Supports-STP-Dispute"], "platform_id": "N3K-C3132Q-40G", "port_id": "Ethernet1/1"}]}}
```

XML および JSON でサポートされたコマンド

NX-OS は、次の構造化された出力フォーマットで、さまざまな **show** コマンドの標準規格出力のリダイレクトをサポートしています。

- XML
- JSON
- JSON フォーマット出力の標準規格ブロックを読みやすくする JSON Pretty
- NX-OS リリース 9.3（1）で導入された JSON Native と JSON Pretty Native は、追加のコマンド解釈レイヤーをバイパスすることにより、JSON 出力をより高速かつ効率的に表示します。JSON Native および JSON Pretty Native は、出力のデータ型を保持します。出力用の文字列に変換する代わりに、整数を整数として表示します。

標準規格の NX-OS 出力を JSON、JSON Pretty、または XML フォーマットに変換することは、出力を JSON または XML インタープリタに「パイプ」することによって、NX-OS CLI で発生します。たとえば、論理パイプ(|)を使用して **show ip access** コマンドを発行し、JSON、JSON Pretty、JSON Native、JSON Native Pretty、または XML を指定すると、NX-OS コマンド出力が適切に構造化され、そのフォーマットでエンコードされます。この機能により、プログラムによるデータの解析が可能になり、ソフトウェアストリーミングテレメトリを介したスイッチからのストリーミングデータがサポートされます。Cisco NX-OS のほとんどのコマンドは、JSON、JSON Pretty、および XML 出力をサポートしています。

この機能の選択された例を以下に表示します。

XML および JSON 出力の例

次の例は、ハードウェア テーブルのユニキャストおよびマルチキャストルーティング エントリを JSON 形式で表示する方法を示しています。

```
switch(config)# show hardware profile status | json
{"total_lpm": ["8191", "1024"], "total_host": "8192", "max_host4_limit": "4096",
 "max_host6_limit": "2048", "max_mcast_limit": "2048", "used_lpm_total": "9", "used_v4_lpm": "6", "used_v6_lpm": "3", "used_v6_lpm_128": "1", "used_host_lpm_total": "0", "used_host_v4_lpm": "0", "used_host_v6_lpm": "0", "used_mcast": "0", "used_mcast_oifl": "2", "used_host_in_host_total": "13", "used_host4_in_host": "12", "used_host6_in_host": "1", "max_ecmp_table_limit": "64", "used_ecmp_table": "0", "mfib_fd_status": "Disabled", "mfib_fd_maxroute": "0", "mfib_fd_count": "0"}
switch(config)#
```

次に、ハードウェアテーブルのユニキャストおよびマルチキャストルーティング エントリを XML 形式で表示する例を示します。

```
switch(config)# show hardware profile status | xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<nf:rpc-reply xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns="http://www.cisco.com/nxos:1.0:fib">
  <nf:data>
    <show>
      <hardware>
        <profile>
          <status>
            <__XML__OPT_Cmd_dynamic_tcam_status>
              <__XML__OPT_Cmd_dynamic_tcam_status__readonly__>
                <__readonly__>
                  <total_lpm>8191</total_lpm>
                  <total_host>8192</total_host>
                  <total_lpm>1024</total_lpm>
                  <max_host4_limit>4096</max_host4_limit>
                  <max_host6_limit>2048</max_host6_limit>
                  <max_mcast_limit>2048</max_mcast_limit>
                  <used_lpm_total>9</used_lpm_total>
                  <used_v4_lpm>6</used_v4_lpm>
                  <used_v6_lpm>3</used_v6_lpm>
                  <used_v6_lpm_128>1</used_v6_lpm_128>
                  <used_host_lpm_total>0</used_host_lpm_total>
                  <used_host_v4_lpm>0</used_host_v4_lpm>
                  <used_host_v6_lpm>0</used_host_v6_lpm>
                </__readonly__>
              </__XML__OPT_Cmd_dynamic_tcam_status__readonly__>
            </__XML__OPT_Cmd_dynamic_tcam_status>
          </status>
        </profile>
      </hardware>
    </show>
  </nf:data>
</nf:rpc-reply>
```

```

        <used_mcast>0</used_mcast>
        <used_mcast_oif1>2</used_mcast_oif1>
        <used_host_in_host_total>13</used_host_in_host_total>
        <used_host4_in_host>12</used_host4_in_host>
        <used_host6_in_host>1</used_host6_in_host>
        <max_ecmp_table_limit>64</max_ecmp_table_limit>
        <used_ecmp_table>0</used_ecmp_table>
        <mfib_fd_status>Disabled</mfib_fd_status>
        <mfib_fd_maxroute>0</mfib_fd_maxroute>
        <mfib_fd_count>0</mfib_fd_count>
    </__readonly__>
</__XML__OPT_Cmd_dynamic_tcam_status__readonly__>
</__XML__OPT_Cmd_dynamic_tcam_status>
</status>
</profile>
</hardware>
</show>
</nf:data>
</nf:rpc-reply>
]]>]]>
switch(config)#

```

この例では、JSON 形式でスイッチ上に LLDP タイマーを表示する例を示します。

```

switch(config)# show lldp timers | json
{"ttl": "120", "reinit": "2", "tx_interval": "30", "tx_delay": "2", "hold_mplier": "4", "notification_interval": "5"}
switch(config)#

```

この例では、XML 形式でスイッチ上に LLDP タイマーを表示する例を示します。

```

switch(config)# show lldp timers | xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<nf:rpc-reply xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns="http://www.cisco.com/nxos:1.0:lldp">
  <nf:data>
    <show>
      <lldp>
        <timers>
          <__XML__OPT_Cmd_lldp_show_timers__readonly__>
            <__readonly__>
              <ttl>120</ttl>
              <reinit>2</reinit>
              <tx_interval>30</tx_interval>
              <tx_delay>2</tx_delay>
              <hold_mplier>4</hold_mplier>
              <notification_interval>5</notification_interval>
            </__readonly__>
          </__XML__OPT_Cmd_lldp_show_timers__readonly__>
        </timers>
      </lldp>
    </show>
  </nf:data>
</nf:rpc-reply>
]]>]]>
switch(config)#

```

この例は、ACL 統計を XML 形式で表示する方法を示しています。

```

switch-1(config-acl)# show ip access-lists acl-test1 | xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<nf:rpc-reply xmlns="http://www.cisco.com/nxos:1.0:aclmgr" xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0">
  <nf:data>
    <show>
      <__XML__OPT_Cmd_show_acl_ip_ipv6_mac>
        <ip_ipv6_mac>ip</ip_ipv6_mac>
        <access-lists>
          <__XML__OPT_Cmd_show_acl_name>
            <name>acl-test1</name>
          <__XML__OPT_Cmd_show_acl_capture>
            <__XML__OPT_Cmd_show_acl_expanded>
              <__XML__OPT_Cmd_show_acl__readonly__>
                <__readonly__>
                  <TABLE_ip_ipv6_mac>
                    <ROW_ip_ipv6_mac>
                      <op_ip_ipv6_mac>ip</op_ip_ipv6_mac>
                      <show_summary>0</show_summary>
                      <acl_name>acl-test1</acl_name>
                      <statistics>enable</statistics>
                      <frag_opt_permit_deny>permit-all</frag_opt_permit_deny>
                    <TABLE_seqno>
                      <ROW_seqno>
                        <seqno>10</seqno>
                        <permitdeny>permit</permitdeny>
                        <ip>ip</ip>
                        <src_ip_prefix>192.0.2.1/24</src_ip_prefix>
                        <dest_any>any</dest_any>
                      </ROW_seqno>
                    </TABLE_seqno>
                  </ROW_ip_ipv6_mac>
                </TABLE_ip_ipv6_mac>
              </__readonly__>
            </__XML__OPT_Cmd_show_acl__readonly__>
          </__XML__OPT_Cmd_show_acl_expanded>
        </__XML__OPT_Cmd_show_acl_capture>
      </__XML__OPT_Cmd_show_acl_name>
    </access-lists>
  </__XML__OPT_Cmd_show_acl_ip_ipv6_mac>
</show>
</nf:data>
</nf:rpc-reply>
]]>]]>
switch-1(config-acl)#

```

この例は、ACL 統計を JSON 形式で表示する方法を示しています。

```

switch-1(config-acl)# show ip access-lists acl-test1 | json
{"TABLE_ip_ipv6_mac": {"ROW_ip_ipv6_mac": {"op_ip_ipv6_mac": "ip", "show_summary": "0", "acl_name": "acl-test1", "statistics": "enable", "frag_opt_permit_deny": "permit-all", "TABLE_seqno": {"ROW_seqno": {"seqno": "10", "permitdeny": "permit", "ip": "ip", "src_ip_prefix": "192.0.2.1/24", "dest_any": "any"}}}}}
switch-1(config-acl)#

```

この例は、スイッチの冗長性情報を JSON Pretty Native 形式で表示する方法を示しています。

```

switch-1# show system redundancy status | json-pretty native
{
  "rdn_mode_admin":      "HA",
  "rdn_mode_oper":       "None",
  "this_sup":            "(sup-1)",
  "this_sup_rdn_state":  "Active, SC not present",
  "this_sup_sup_state":  "Active",
  "this_sup_internal_state": "Active with no standby",
}

```

```

        "other_sup":      "(sup-1)",
        "other_sup_rdn_state": "Not present"
    }
switch-1#

```

次の例は、スイッチの冗長ステータスを JSON 形式で表示する方法を示しています。

```

switch-1# show system redundancy status | json
{"rdn_mode_admin": "HA", "rdn_mode_oper": "None", "this_sup": "(sup-1)", "this_sup_rdn_state": "Active, SC not present", "this_sup_sup_state": "Active", "this_sup_internal_state": "Active with no standby", "other_sup": "(sup-1)", "other_sup_rdn_state": "Not present"}
nxosv2#
switch-1#

```

次に、XML 形式で IP ルート要約を表示する例を示します。

```

switch-1# show ip route summary | xml
<?xml version="1.0" encoding="ISO-8859-1"?> <nf:rpc-reply
xmlns="http://www.cisco.com/nxos:1.0:urib" xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0">

  <nf:data>
    <show>
      <ip>
        <route>
          <__XML__OPT_Cmd_urib_show_ip_route_command_ip>
            <__XML__OPT_Cmd_urib_show_ip_route_command_unicast>
              <__XML__OPT_Cmd_urib_show_ip_route_command_topology>
                <__XML__OPT_Cmd_urib_show_ip_route_command_l3vm-info>
                  <__XML__OPT_Cmd_urib_show_ip_route_command_rpf>
                    <__XML__OPT_Cmd_urib_show_ip_route_command_ip-addr>
                      <__XML__OPT_Cmd_urib_show_ip_route_command_protocol>
                        <__XML__OPT_Cmd_urib_show_ip_route_command_summary>
                          <__XML__OPT_Cmd_urib_show_ip_route_command_vrf>
                            <__XML__OPT_Cmd_urib_show_ip_route_command__readonly__>
                              <__readonly__>
                                <TABLE_vrf>
                                  <ROW_vrf>
                                    <vrf-name-out>default</vrf-name-out>
                                    <TABLE_addrf>
                                      <ROW_addrf>
                                        <addrf>ipv4</addrf>
                                        <TABLE_summary>
                                          <ROW_summary>
                                            <routes>938</routes>
                                            <paths>1453</paths>
                                            <TABLE_unicast>
                                              <ROW_unicast>
                                                <clientnameuni>am</clientnameuni>
                                                <best-paths>2</best-paths>
                                              </ROW_unicast>
                                              <ROW_unicast>
                                                <clientnameuni>local</clientnameuni>
                                                <best-paths>105</best-paths>
                                              </ROW_unicast>
                                              <ROW_unicast>
                                                <clientnameuni>direct</clientnameuni>
                                                <best-paths>105</best-paths>
                                              </ROW_unicast>
                                              <ROW_unicast>
                                                <clientnameuni>broadcast</clientnameuni>
                                                <best-paths>203</best-paths>
                                              </ROW_unicast>
                                              <ROW_unicast>
                                                <clientnameuni>ospf-10</clientnameuni>

```

```

        <best-paths>1038</best-paths>
      </ROW_unicast>
    </TABLE_unicast>
    <TABLE_route_count>
      <ROW_route_count>
        <mask_len>8</mask_len>
        <count>1</count>
      </ROW_route_count>
      <ROW_route_count>
        <mask_len>24</mask_len>
        <count>600</count>
      </ROW_route_count>
      <ROW_route_count>
        <mask_len>31</mask_len>
        <count>13</count>
      </ROW_route_count>
      <ROW_route_count>
        <mask_len>32</mask_len>
        <count>324</count>
      </ROW_route_count>
    </TABLE_route_count>
  </ROW_summary>
</TABLE_summary>
</ROW_addrf>
</TABLE_addrf>
</ROW_vrf>
</TABLE_vrf>
</__readonly__>
</__XML__OPT_Cmd_urib_show_ip_route_command__readonly__>
</__XML__OPT_Cmd_urib_show_ip_route_command_vrf>
</__XML__OPT_Cmd_urib_show_ip_route_command_summary>
</__XML__OPT_Cmd_urib_show_ip_route_command_protocol>
</__XML__OPT_Cmd_urib_show_ip_route_command_ip-addr>
</__XML__OPT_Cmd_urib_show_ip_route_command_rpf>
</__XML__OPT_Cmd_urib_show_ip_route_command_l3vm-info>
</__XML__OPT_Cmd_urib_show_ip_route_command_topology>
</__XML__OPT_Cmd_urib_show_ip_route_command_unicast>
</__XML__OPT_Cmd_urib_show_ip_route_command_ip>
</route>
</ip>
</show>
</nf:data>
</nf:rpc-reply>
]]>]]>
switch-1#

```

次の例は、スイッチの OSPF ルーティング パラメータを JSON ネイティブ形式で表示する方法を示しています。

```

switch-1# show ip ospf | json native
{"TABLE_ctx":{"ROW_ctx":[{"ptag":"Blah","instance_number":4,"cname":"default","rid":"0.0.0.0","stateful_ha":"true","gr_ha":"true","gr_planned_only":"true","gr_grace_period":"PT60S","gr_state":"inactive","gr_last_status":"None","support_tos0_only":"true","support_opaque_lsa":"true","is_abr":"false","is_asbr":"false","admin_dist":110,"ref_bw":40000,"spf_start_time":"PT0S","spf_hold_time":"PT1S","spf_max_time":"PT5S","lsa_start_time":"PT0S","lsa_hold_time":"PT5S","lsa_max_time":"PT5S","min_lsa_arr_time":"PT1S","lsa_aging_pace":10,"spf_max_paths":8,"max_metric_adver":"false","asext_lsa_cnt":0,"asext_lsa_crc":0,"asopaque_lsa_cnt":0,"asopaque_lsa_crc":0,"area_total":0,"area_normal":0,"area_stub":0,"area_nssa":0,"act_area_total":0,"act_area_normal":0,"act_area_stub":0,"act_area_nssa":0,"no_discard_rt_ext":"false","no_discard_rt_int":"false"},{"ptag":"100","instance_number":3,"cname":"default","rid":"0.0.0.0","stateful_ha":"true","gr_ha":"true","gr_planned_only":"true","gr_grace_period":"PT60S","gr_state":"inactive","gr_last_status":"None","support_tos0_only":"true","support_opaque_lsa":"true"}]}

```

```
, "is_abr": "false", "is_asbr": "false", "admin_dist": 110, "ref_bw": 40000, "spf_start_time": "PT0S", "spf_hold_time": "PT1S", "spf_max_time": "PT5S", "lsa_start_time": "PT0S", "lsa_hold_time": "PT5S", "lsa_max_time": "PT5S", "min_lsa_arr_time": "PT1S", "lsa_aging_pace": 10, "spf_max_paths": 8, "max_metric_adver": "false", "asext_lsa_cnt": 0, "asext_lsa_crc": "0", "asopaque_lsa_cnt": 0, "asopaque_lsa_crc": "0", "area_total": 0, "area_normal": 0, "area_stub": 0, "area_nssa": 0, "act_area_total": 0, "act_area_normal": 0, "act_area_stub": 0, "act_area_nssa": 0, "no_discard_rt_ext": "false", "no_discard_rt_int": "false"}, {"ptag": "111", "instance_number": 1, "cname": "default", "rid": "0.0.0.0", "stateful_ha": "true", "gr_ha": "true", "gr_planned_only": "true", "gr_grace_period": "PT60S", "gr_state": "inactive", "gr_last_status": "None", "support_tos0_only": "true", "support_opaque_lsa": "true", "is_abr": "false", "is_asbr": "false", "admin_dist": 110, "ref_bw": 40000, "spf_start_time": "PT0S", "spf_hold_time": "PT1S", "spf_max_time": "PT5S", "lsa_start_time": "PT0S", "lsa_hold_time": "PT5S", "lsa_max_time": "PT5S", "min_lsa_arr_time": "PT1S", "lsa_aging_pace": 10, "spf_max_paths": 8, "max_metric_adver": "false", "asext_lsa_cnt": 0, "asext_lsa_crc": "0", "asopaque_lsa_cnt": 0, "asopaque_lsa_crc": "0", "area_total": 0, "area_normal": 0, "area_stub": 0, "area_nssa": 0, "act_area_total": 0, "act_area_normal": 0, "act_area_stub": 0, "act_area_nssa": 0, "no_discard_rt_ext": "false", "no_discard_rt_int": "false"}, {"ptag": "112", "instance_number": 2, "cname": "default", "rid": "0.0.0.0", "stateful_ha": "true", "gr_ha": "true", "gr_planned_only": "true", "gr_grace_period": "PT60S", "gr_state": "inactive", "gr_last_status": "None", "support_tos0_only": "true", "support_opaque_lsa": "true", "is_abr": "false", "is_asbr": "false", "admin_dist": 110, "ref_bw": 40000, "spf_start_time": "PT0S", "spf_hold_time": "PT1S", "spf_max_time": "PT5S", "lsa_start_time": "PT0S", "lsa_hold_time": "PT5S", "lsa_max_time": "PT5S", "min_lsa_arr_time": "PT1S", "lsa_aging_pace": 10, "spf_max_paths": 8, "max_metric_adver": "false", "asext_lsa_cnt": 0, "asext_lsa_crc": "0", "asopaque_lsa_cnt": 0, "asopaque_lsa_crc": "0", "area_total": 0, "area_normal": 0, "area_stub": 0, "area_nssa": 0, "act_area_total": 0, "act_area_normal": 0, "act_area_stub": 0, "act_area_nssa": 0, "no_discard_rt_ext": "false", "no_discard_rt_int": "false"}]}}
switch-1#
```

次の例は、OSPF ルーティング パラメータを JSON Pretty Native 形式で表示する方法を示しています。

```
switch-1# show ip ospf | json-pretty native
{
  "TABLE_ctx": {
    "ROW_ctx": [{
      "ptag": "Blah",
      "instance_number": 4,
      "cname": "default",
      "rid": "0.0.0.0",
      "stateful_ha": "true",
      "gr_ha": "true",
      "gr_planned_only": "true",
      "gr_grace_period": "PT60S",
      "gr_state": "inactive",
      "gr_last_status": "None",
      "support_tos0_only": "true",
      "support_opaque_lsa": "true",
      "is_abr": "false",
      "is_asbr": "false",
      "admin_dist": 110,
      "ref_bw": 40000,
      "spf_start_time": "PT0S",
      "spf_hold_time": "PT1S",
      "spf_max_time": "PT5S",
      "lsa_start_time": "PT0S",
      "lsa_hold_time": "PT5S",
      "lsa_max_time": "PT5S",
      "min_lsa_arr_time": "PT1S",
      "lsa_aging_pace": 10,
      "spf_max_paths": 8,
      "max_metric_adver": "false",
```

```

"asext_lsa_cnt": 0,
"asext_lsa_crc": "0",
"asopaque_lsa_cnt": 0,
"asopaque_lsa_crc": "0",
"area_total": 0,
"area_normal": 0,
"area_stub": 0,
"area_nssa": 0,
"act_area_total": 0,
"act_area_normal": 0,
"act_area_stub": 0,
"act_area_nssa": 0,
"no_discard_rt_ext": "false",
"no_discard_rt_int": "false"
}, {
  "ptag": "100",
  "instance_number": 3,
  "cname": "default",
  "rid": "0.0.0.0",
  "stateful_ha": "true",
  "gr_ha": "true",
  "gr_planned_only": "true",
  "gr_grace_period": "PT60S",
  "gr_state": "inactive",

  ... content deleted for brevity ...

  "max_metric_adver": "false",
  "asext_lsa_cnt": 0,
  "asext_lsa_crc": "0",
  "asopaque_lsa_cnt": 0,
  "asopaque_lsa_crc": "0",
  "area_total": 0,
  "area_normal": 0,
  "area_stub": 0,
  "area_nssa": 0,
  "act_area_total": 0,
  "act_area_normal": 0,
  "act_area_stub": 0,
  "act_area_nssa": 0,
  "no_discard_rt_ext": "false",
  "no_discard_rt_int": "false"
}]
}
switch-1#

```

次の例は、JSON 形式で IP ルート要約を表示する例を示します。

```
switch-1# show ip route summary | json
{"TABLE_vrf": {"ROW_vrf": {"vrf-name-out": "default", "TABLE_addrf": {"ROW_addrf": {"addrf": "ipv4", "TABLE_summary": {"ROW_summary": {"routes": "938", "paths": "1453", "TABLE_unicast": {"ROW_unicast": [{"clientnameuni": "am", "best-paths": "2"}, {"clientnameuni": "local", "best-paths": "105"}, {"clientnameuni": "direct", "best-paths": "105"}, {"clientnameuni": "broadcast", "best-paths": "203"}, {"clientnameuni": "ospf-10", "best-paths": "1038"}]}}, "TABLE_route_count": {"ROW_route_count": [{"mask_len": "8", "count": "1"}, {"mask_len": "24", "count": "600"}, {"mask_len": "31", "count": "13"}, {"mask_len": "32", "count": "324"}]}}}}}}}}}
switch-1#
```

次の例は、JSON Pretty 形式で IP ルート要約を表示する例を示します。

```
switch-1# show ip route summary | json-pretty
{
  "TABLE vrf": {
```



```

"ROW_vrf": {
  "vrf-name-out": "default",
  "TABLE_addrf": {
    "ROW_addrf": {
      "addrf": "ipv4",
      "TABLE_summary": {
        "ROW_summary": {
          "routes": "938",
          "paths": "1453",
          "TABLE_unicast": {
            "ROW_unicast": [
              {
                "clientnameuni": "am",
                "best-paths": "2"
              },
              {
                "clientnameuni": "local",
                "best-paths": "105"
              },
              {
                "clientnameuni": "direct",
                "best-paths": "105"
              },
              {
                "clientnameuni": "broadcast",
                "best-paths": "203"
              },
              {
                "clientnameuni": "ospf-10",
                "best-paths": "1038"
              }
            ]
          }
        },
        "TABLE_route_count": {
          "ROW_route_count": [
            {
              "mask_len": "8",
              "count": "1"
            },
            {
              "mask_len": "24",
              "count": "600"
            },
            {
              "mask_len": "31",
              "count": "13"
            },
            {
              "mask_len": "32",
              "count": "324"
            }
          ]
        }
      }
    }
  }
}
switch-1#

```

JSON-RPC、JSON、および XML でサポートされるコマンド

Cisco NX-OS リリース 6.0 (2) U4 (1) 以降、次のコマンドの JSON-RPC、JSON、および XML 出力がサポートされます。

- show bgp all
- show bgp process
- show bgp convergence
- bgp ip ユニキャスト / マルチキャストを表示します
- bgp ipv4 ユニキャスト / マルチキャストを表示します
- bgp ipv6 ユニキャスト / マルチキャストを表示します
- show bgp paths
- show bgp peer-policy
- show bgp vrf
- show bgp sessions
- show bgp statistics
- consistency-checker が ipv4 の転送を表示
- consistency-checker が ipv6 の転送を表示
- show lldp neighbors
- show lldp neighbors detail
- lldp ネイバー インターフェイス イーサネット x / x を表示
- lldp ネイバー インターフェイス イーサネット x / x の詳細を表示
- lldp portid-subtype を表示します
- show lldp timers
- show lldp tlv-select
- show lldp traffic
- lldp トラフィック インターフェイス イーサネット x / x を表示
- show process memory
- プロセス cpu とプロセスを表示します
- ルーティング vrf 全てを表示

- システム内部転送ルートの要約を表示
- `show system resources`



(注) NX-API がサポートする最大データ サイズは 20MB です。20MB の制限を超えた場合、次のコマンドはサポートされません (NX-OS 6.0 (2) A8 (9) 以前)。

- ハードウェア プロファイル バッファ モニタの詳細を表示します
- ハードウェア プロファイル バッファ モニタ マルチキャスト 1 詳細を表示します
- ハードウェア プロファイル バッファ モニタ マルチキャスト 2 の詳細を表示します
- ハードウェア プロファイル バッファ モニタ マルチキャスト 3 の詳細を表示します

XML および JSON 出力の例

次の例は、ハードウェア テーブルのユニキャストおよびマルチキャスト ルーティング エントリを JSON 形式で表示する方法を示しています。

```
switch(config)# show hardware profile status | json
{"total_lpm": ["8191", "1024"], "total_host": "8192", "max_host4_limit": "4096",
 "max_host6_limit": "2048", "max_mcast_limit": "2048", "used_lpm_total": "9", "used_v4_lpm": "6", "used_v6_lpm": "3", "used_v6_lpm_128": "1", "used_host_lpm_total": "0", "used_host_v4_lpm": "0", "used_host_v6_lpm": "0", "used_mcast": "0", "used_mcast_oif1": "2", "used_host_in_host_total": "13", "used_host4_in_host": "12", "used_host6_in_host": "1", "max_ecmp_table_limit": "64", "used_ecmp_table": "0", "mfib_fd_status": "Disabled", "mfib_fd_maxroute": "0", "mfib_fd_count": "0"}
switch(config)#
```

次に、ハードウェア テーブルのユニキャストおよびマルチキャスト ルーティング エントリを XML 形式で表示する例を示します。

```
switch(config)# show hardware profile status | xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<nf:rpc-reply xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns="http://www.cisco.com/nxos:1.0:fib">
  <nf:data>
    <show>
      <hardware>
        <profile>
          <status>
            <__XML__OPT_Cmd_dynamic_tcam_status>
              <__XML__OPT_Cmd_dynamic_tcam_status__readonly__>
                <__readonly__>
                  <total_lpm>8191</total_lpm>
                  <total_host>8192</total_host>
                  <total_lpm>1024</total_lpm>
                  <max_host4_limit>4096</max_host4_limit>
                  <max_host6_limit>2048</max_host6_limit>
                  <max_mcast_limit>2048</max_mcast_limit>
```

```

<used_lpm_total>9</used_lpm_total>
<used_v4_lpm>6</used_v4_lpm>
<used_v6_lpm>3</used_v6_lpm>
<used_v6_lpm_128>1</used_v6_lpm_128>
<used_host_lpm_total>0</used_host_lpm_total>
<used_host_v4_lpm>0</used_host_v4_lpm>
<used_host_v6_lpm>0</used_host_v6_lpm>
<used_mcast>0</used_mcast>
<used_mcast_oifl>2</used_mcast_oifl>
<used_host_in_host_total>13</used_host_in_host_total>
<used_host4_in_host>12</used_host4_in_host>
<used_host6_in_host>1</used_host6_in_host>
<max_ecmp_table_limit>64</max_ecmp_table_limit>
<used_ecmp_table>0</used_ecmp_table>
<mfib_fd_status>Disabled</mfib_fd_status>
<mfib_fd_maxroute>0</mfib_fd_maxroute>
<mfib_fd_count>0</mfib_fd_count>
</__readonly__>
</__XML__OPT_Cmd_dynamic_tcam_status__readonly__>
</__XML__OPT_Cmd_dynamic_tcam_status>
</status>
</profile>
</hardware>
</show>
</nf:data>
</nf:rpc-reply>
]]>]]>
switch(config)#

```

この例では、JSON 形式でスイッチ上に LLDP タイマーを表示する例を示します。

```

switch(config)# show lldp timers | json
{"ttl": "120", "reinit": "2", "tx_interval": "30", "tx_delay": "2", "hold_mplier": "4", "notification_interval": "5"}
switch(config)#

```

この例では、XML 形式でスイッチ上に LLDP タイマーを表示する例を示します。

```

switch(config)# show lldp timers | xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<nf:rpc-reply xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns="http://www.cisco.com/nxos:1.0:lldp">
  <nf:data>
    <show>
      <lldp>
        <timers>
          <__XML__OPT_Cmd_lldp_show_timers__readonly__>
            <__readonly__>
              <ttl>120</ttl>
              <reinit>2</reinit>
              <tx_interval>30</tx_interval>
              <tx_delay>2</tx_delay>
              <hold_mplier>4</hold_mplier>
              <notification_interval>5</notification_interval>
            </__readonly__>
          </__XML__OPT_Cmd_lldp_show_timers__readonly__>
        </timers>
      </lldp>
    </show>
  </nf:data>
</nf:rpc-reply>

```

```
</nf:data>  
</nf:rpc-reply>  
]]>]]>  
switch(config)#
```




第 15 章

NX-API 応答コード

• NX-API 応答コードの表 (179 ページ)

NX-API 応答コードの表

次に、NX-API 応答の考えられる NX-API エラー、エラーコード、およびメッセージを示します。

次に、NX-API 応答の考えられる NX-API エラー、エラーコード、およびメッセージを示します。

リクエスト形式が XML または JSON フォーマットの場合、NX-API エラー、エラーコード、および NX-API 応答のメッセージは次のとおりです。



(注) 標準の HTTP エラーコードは、ハイパーテキスト転送プロトコル (HTTP) ステータスコードレジストリ (<http://www.iana.org/assignments/http-status-codes/http-status-codes.xhtml>) にあります。

表 13: NX-API 応答コード

[NX-API 応答 (NX-API Response)]	コード	メッセージ
成功	200	成功。
CUST_OUTPUT_PIPED	204	要求により、出力は別の場所にパイプされます。
BASH_CMD_ERR	400	入力 Bash コマンドエラー。
CHUNK_ALLOW_ONE_CMD_ERR	400	チャンクは 1 つのコマンドにのみ許可されます。
CLI_CLIENT_ERR	400	CLI の実行エラー
CLI_CMD_ERR	400	CLI コマンドエラーの入力。

EOC_NOT_ALLOWED_ERR	400	eoc 値は、リクエストのセッション ID として許可されていません。
IN_MSG_ERR	400	要求メッセージが無効です。
MSG_VER_MISMATCH	400	メッセージバージョンの不一致
NO_INPUT_CMD_ERR	400	入力コマンドがありません。
SID_NOT_ALLOWED_ERR	400	セッション ID として入力された文字が無効です。
PERM_DENY_ERR	401	権限が拒否されました。
CONF_NOT_ALLOW_SHOW_ERR	405	構成モードは [表示 (show)] を許可しません。
SHOW_NOT_ALLOW_CONF_ERR	405	表示モードでは構成できません。
EXCEED_MAX_SHOW_ERR	413	連続する show コマンドの最大数を超過しました。最大値は 10 です。
MSG_SIZE_LARGE_ERR	413	応答サイズが大きすぎます。
RESP_SIZE_LARGE_ERR	413	応答サイズが最大メッセージサイズを超えたため、処理を停止しました。最大サイズは 200 MB です。
EXCEED_MAX_INFLIGHT_CHUNK_REQ_ERR	429	同時チャンク リクエストの最大数は超過しています。最大は 2 です。
OBJ_NOT_EXIST	432	要求したオブジェクトが存在しません。
BACKEND_ERR	500	バックエンド処理エラー。
CREATE_CHECKPOINT_ERR	500	チェックポイントの作成をするエラー。
DELETE_CHECKPOINT_ERR	500	チェックポイントの削除中にエラーが発生しました。
FILE_OPER_ERR	500	システム内部ファイル操作エラー。
LIBXML_NS_ERR	500	システムの内部 LIBXML NS エラー。
LIBXML_PARSE_ERR	500	システムの内部 LIBXML 解析エラー。
LIBXML_PATH_CTX_ERR	500	システムの内部 LIBXML パス コンテキストエラー。
MEM_ALLOC_ERR	500	システムの内部メモリ割り当てエラー。

ROLLBACK_ERR	500	ロールバックの実行中にエラーが発生しました。
SERVER_BUSY_ERR	500	サーバーがビジー状態のため、リクエストは拒否されました。
USER_NOT_FOUND_ERR	500	入力またはキャッシュからユーザーが見つかりません。
VOLATILE_FULL	500	揮発性メモリは一杯です。メモリ スペースを解放して、再試行してください。
XML_TO_JSON_CONVERT_ERR	500	XML から JSON への変換エラー。
BASH_CMD_NOT_SUPPORTED_ERR	501	Bash コマンドはサポートされていません。
CHUNK_ALLOW_XML_ONLY_ERR	501	チャンクは XML 出力のみを許可します。
CHUNK_ONLY_ALLOWED_IN_SHOW_ERR	501	応答のチャンクは、show コマンドでのみ許可されます。
CHUNK_TIMEOUT	501	チャンク応答の生成中にタイムアウトしました。
CLI_CMD_NOT_SUPPORTED_ERR	501	CLI コマンドはサポートされていません。
JSON_NOT_SUPPORTED_ERR	501	大量の出力のため、JSONはサポートされていません。
MALFORMED_XML	501	不正な XML 出力。
MSG_TYPE_UNSUPPORTED_ERR	501	メッセージタイプはサポートされていません
OUTPUT_REDIRECT_NOT_SUPPORTED_ERR	501	出力リダイレクトはサポートされていません。
PIPE_OUTPUT_NOT_SUPPORTED_ERR	501	パイプ操作はサポートされていません。
PIPE_OUTPUT_NOT_SUPPORTED_ERR	501	パイプ操作はサポートされていません。
PIPE_XML_NOT_ALLOWED_IN_INPUT	501	入力でパイプXMLは許可されていません。
PIPE_NOT_ALLOWED_IN_INPUT	501	この入力タイプにはパイプを使用できません。
RESP_BIG_JSON_NOT_ALLOWED_ERR	501	応答の出力量が多い。JSONはサポートされません。

RESP_BIG_USE_CHUNK_ERR	501	応答が許容最大値を超えています。最大は 10 MB です。チャンクを有効にして XML または JSON 出力を使用します。
RESP_BIG_JSON_NOT_ALLOWED_ERR	501	応答の出力量が多い。JSONはサポートされません。
STRUCT_NOT_SUPPORTED_ERR	501	構造化出力はサポートされていません。
ERR_UNDEFINED	600	未定義。



第 16 章

NX-API 開発者サンドボックス

- [NX-API 開発者サンドボックス: 9.2 \(2\) より前の NX-OS リリース \(183 ページ\)](#)
- [NX-API 開発者サンドボックス: NX-OS リリース 9.2 \(2\) 以降 \(190 ページ\)](#)

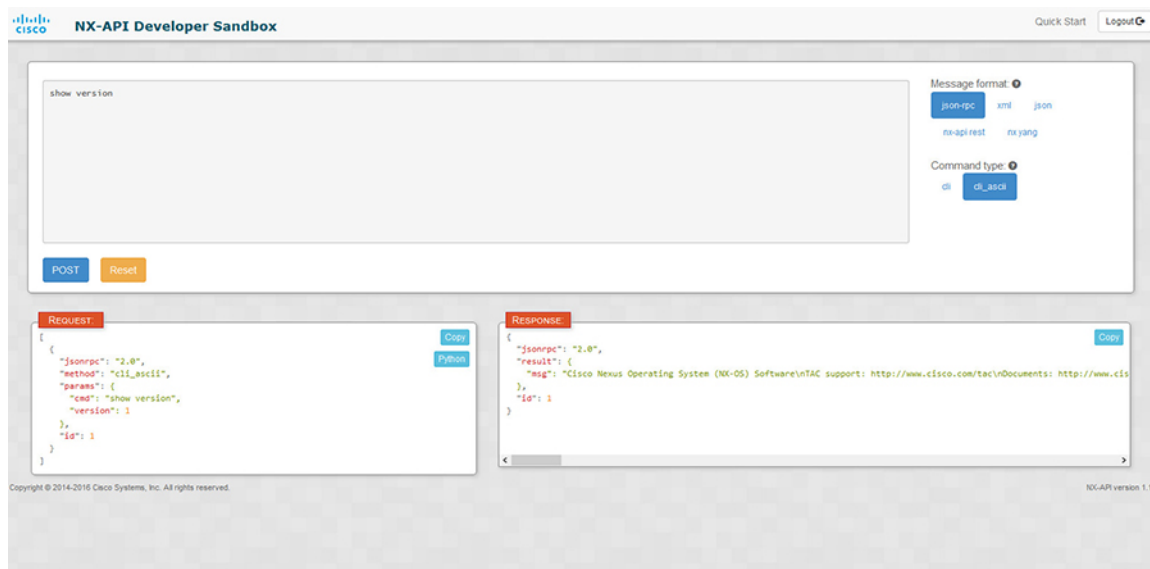
NX-API 開発者サンドボックス: 9.2 (2) より前の NX-OS リリース

About the NX-API デベロッパー サンドボックス

NX-API Developer Sandbox は、スイッチでホストされる Web フォームです。NX-OS CLI コマンドを同等の XML または JSON ペイロードに変換し、NX-API REST ペイロードを同等の CLI に変換します。

図に示すように、Web フォームは 3 つのペイン（コマンド（上部ペイン）、要求、および応答）を持つ 1 つの画面です。

図 1: リクエストと出力応答の例を含む NX-API デベロッパー サンドボックス



コマンドペインのコントロールを使用すると、サポートされている API のメッセージフォーマット（NX-API REST など）とコマンドタイプ（XML や JSON など）を選択できます。使用可能なコマンドタイプオプションは、選択したメッセージフォーマットによって異なります。

コマンドペインに 1 つ以上の CLI コマンドを入力するか貼り付けると、Web フォームはコマンドを API ペイロードに変換し、構成エラーをチェックし、結果のペイロードを要求ペインに表示します。次に、コマンドペインの POST ボタンを使用して、ペイロードをサンドボックスからスイッチに直接送信することを選択した場合、応答ペインに API 応答が表示されます。

逆に、コマンドペインに NX-API REST 指定名 (DN) とペイロードを入力し、**nx-api rest** メッセージフォーマットと **[モデル (model)]** コマンドタイプを選択すると、デベロッパー サンドボックスはペイロードの構成エラーをチェックし、応答ペインに同等の CLI が表示されます。

注意事項と制約事項

デベロッパー サンドボックスのガイドラインと制限は次のとおりです：

- サンドボックスで **POST** をクリックすると、コマンドがスイッチにコミットされ、構成または状態が変更される可能性があります。
- 一部の機能構成コマンドは、関連する機能が有効になるまで使用できません。

メッセージフォーマットとコマンドタイプの構成

[メッセージフォーマット (Message Format)] と [コマンドタイプ (Command Type)] は、コマンドペイン（上部ペイン）の右上隅で構成されます。[メッセージフォーマット (Message Format)] で、使用する API プロトコルのフォーマットを選択します。開発者サンドボックスは、次の API プロトコルをサポートしています。

表 14: NX-OS API プロトコル

プロトコル	説明
json-rpc	JSON ペイロードで NX-OS CLI コマンドを配信するために使用できる標準の軽量リモートプロシージャコール (RPC) プロトコル。JSON-RPC 2.0 仕様は、 jsonrpc.org によって概説されています。
xml	XML ペイロードで NX-OS CLI または bash コマンドを配信するための Cisco NX-API 独自のプロトコル。
json	JSON ペイロードで NX-OS CLI または bash コマンドを配信するための Cisco NX-API 独自のプロトコル。
nx-api rest	内部 NX-OS データ管理エンジン (DME) モデルで管理対象オブジェクト (MO) とそのプロパティを操作および読み取るための Cisco NX-API 独自のプロトコル。詳細については、 [Cisco Nexus NX-API リファレンス (Cisco Nexus NX-API References)] を参照してください。
nx yang	構成および状態データ用の YANG (「Yet Another Next Generation」) データモデリング言語。

[メッセージフォーマット (Message Format)] を選択すると、[コマンドタイプ (Command Type)] オプションのセットが [メッセージフォーマット (Message Format)] コントロールのすぐ下に表示されます。[コマンドタイプ (Command Type)] の設定は、入力 CLI を制限でき、[要求 (Request)] と [応答 (Response)] のフォーマットを決定できます。オプションは、選択した [メッセージフォーマット (Message Format)] によって異なります。各 [メッセージフォーマット (Message Format)] について、次の表で [コマンドタイプ (Command Type)] オプションについて説明します。

表 15: コマンドタイプ

メッセージ形式	コマンドタイプ
json-rpc	- cli — show または構成コマンド - cli_ascii — show または構成コマンド、フォーマットせずに出力

メッセージ形式	コマンドタイプ
xml	• <code>cli_show</code> — コマンドを表示します。コマンドが XML 出力をサポートしていない場合、エラーメッセージが返されます。 • <code>cli_show_ascii</code> — コマンドを表示、フォーマットせずに出力 • <code>cli_conf</code> — 構成 コマンド。対話型の構成コマンドはサポートされていません。 • <code>bash</code> — <code>bash</code> コマンド。ほとんどの非対話型 <code>bash</code> コマンドがサポートされています。 (注) スイッチで <code>bash</code> シェルを有効にする必要があります。
json	• <code>cli_show</code> — コマンドを表示します。コマンドが XML 出力をサポートしていない場合、エラーメッセージが返されます。 • <code>cli_show_ascii</code> — コマンドを表示、フォーマットせずに出力 • <code>cli_conf</code> — 構成 コマンド。対話型の構成コマンドはサポートされていません。 • <code>bash</code> — <code>bash</code> コマンド。ほとんどの非対話型 <code>bash</code> コマンドがサポートされています。 (注) スイッチで <code>bash</code> シェルを有効にする必要があります。
nx-api rest	• <code>cli</code> — 構成コマンド • モデル — DN および対応するペイロード。
nx yang	• <code>json</code> — ペイロードに JSON 構造が使用されます • <code>xml</code> — XML 構造がペイロードに使用されます

出力チャンク

大量の `show` コマンド出力を処理するために、一部の NX-API メッセージフォーマットでは、`show` コマンドの出力チャンクがサポートされています。この場合、**[チャンク モードを有効にする (Enable chunk mode)]** チェックボックスが、セッション ID (SID) 入力ボックスとともに **[コマンドタイプ (Command Type)]** コントロールの下に表示されます。

チャンクが有効な場合、応答は複数の「チャンク」で送信され、最初のチャンクが即時のコマンド応答で送信されます。応答メッセージの次のチャンクを取得するには、前の応答メッセージのセッション ID に設定された **SID** を使用して NX-API 要求を送信する必要があります。

デベロッパー サンドボックスを使用

デベロッパー サンドボックスを使用して CLI コマンドをペイロードに変換する



ヒント オンライン ヘルプは、サンドボックス ウィンドウの右上隅にある **[クイック スタート (Quick Start)]** をクリックすると利用できます。

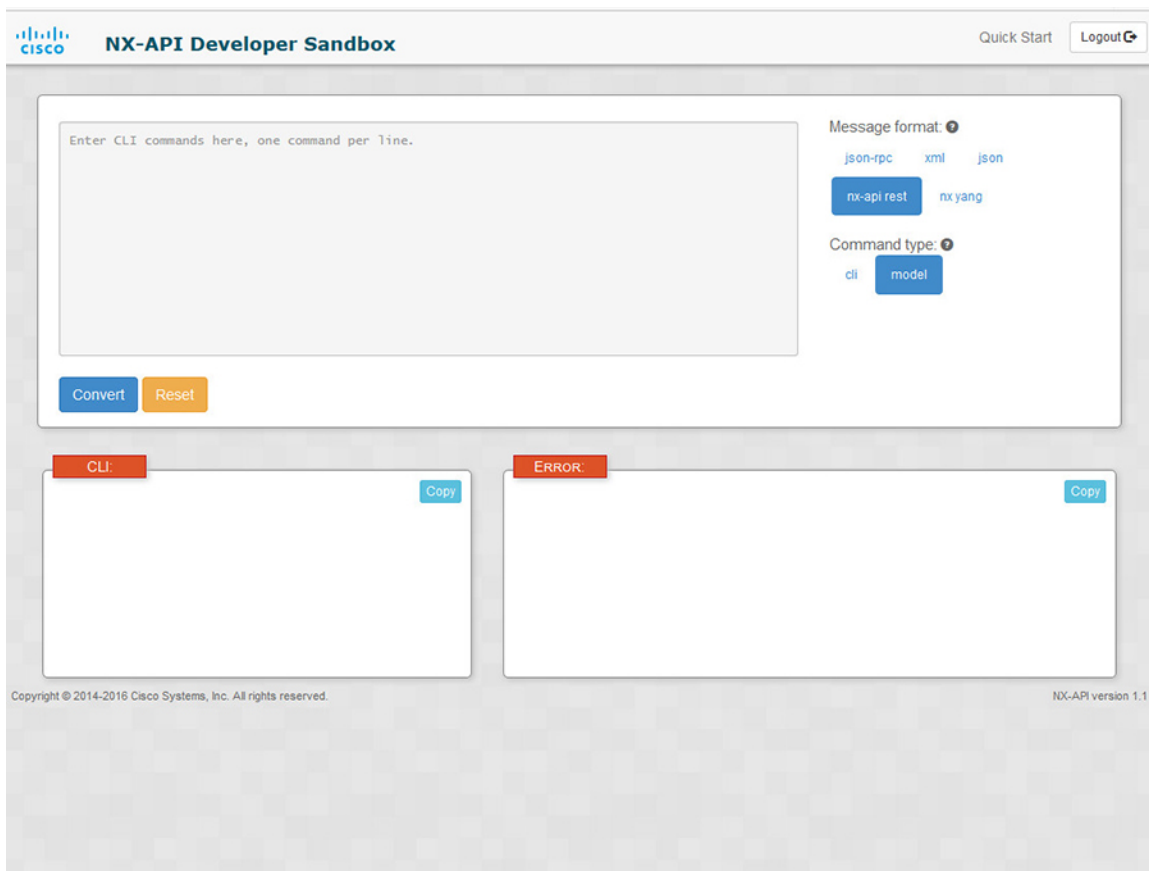
レスポンス コードやセキュリティ メソッドなどの詳細については、NX-API CLI の章を参照してください。

構成コマンドはサポートされていません。

手順

- Step 1** 使用する API プロトコルの**[メッセージ形式 (Message Format)]**と**[コマンド タイプ (Command Type)]**を構成します。
- 詳細な手順については、[メッセージフォーマットとコマンドタイプの構成 \(184ページ\)](#)を参照してください。
- Step 2** 上部ペインのテキスト エントリ ボックスに、NX-OS CLI 構成コマンドを 1 行に 1 つずつ入力するか貼り付けます。
- 上部ペインの下部にある**[リセット (Reset)]**をクリックすると、テキスト エントリ ボックス (および **[要求 (Request)]** ペインと **[応答 (Response)]** ペイン) の内容を消去できます。

デベロッパー サンドボックスを使用して CLI コマンドをペイロードに変換する



Step 3 トップ ペインの最下部にある **[変換 (Convert)]** をクリックします。

CLI コマンドに構成エラーが含まれていない場合、ペイロードは**[要求 (Request)]** ペインに表示されます。エラーが存在する場合は、説明のエラー メッセージが **[応答 (Response)]** ペインに表示されます。

The screenshot displays the NX-API Developer Sandbox web interface. At the top, the Cisco logo and 'NX-API Developer Sandbox' title are visible, along with 'Quick Start' and 'Logout' links. The main workspace contains a large text area with a JSON payload:

```
api/mo/sys.json
{
  "topSystem": {
    "attributes": {
      "name": "REST2CLI"
    }
  }
}
```

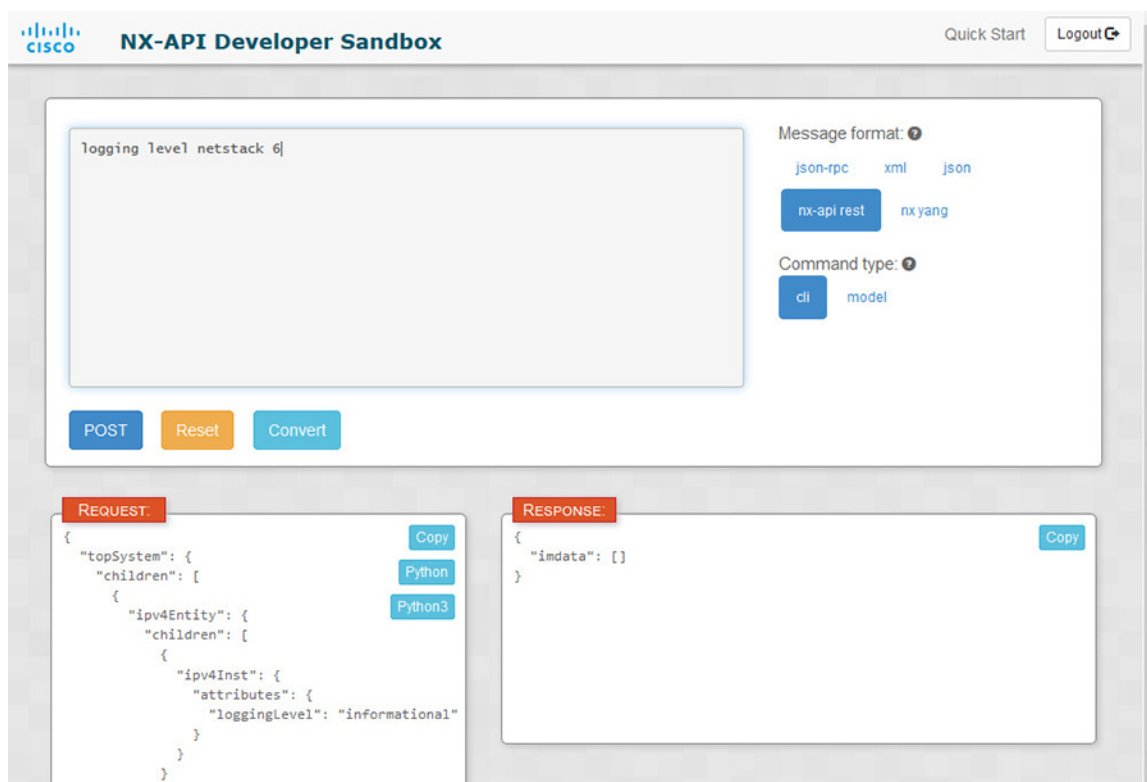
 To the right of the text area are two dropdown menus: 'Message format' with options 'json-rpc', 'xml', and 'json' (where 'json' is selected), and 'Command type' with options 'cli' and 'model' (where 'model' is selected). Below these are 'Convert' and 'Reset' buttons. Underneath the main workspace are two side-by-side panels. The left panel, titled 'CLI:', shows the output 'hostname REST2CLI' and a 'Copy' button. The right panel, titled 'ERROR:', is empty and also has a 'Copy' button. At the bottom of the interface, there is a copyright notice 'Copyright © 2014-2016 Cisco Systems, Inc. All rights reserved.' and 'NX-API version 1.1'. A status bar at the very bottom indicates 'Waiting for bam.nr-data.net...'.

Step 4 [リクエスト (Request)] ペインに有効なペイロードが表示されている場合は、**POST** をクリックして、ペイロードを API 呼び出しとしてスイッチに送信できます。

スイッチからのレスポンスは [Response (応答)] ペインに表示されます。

警告

POST をクリックすると、コマンドがスイッチにコミットされ、構成または状態が変更される可能性があります。



- Step 5** ペインで[コピー (Copy)]をクリックすると、[要求 (Request)] ペインまたは[応答 (Response)] ペインの格納ファイルをクリップボードにコピーできます。
- Step 6** [リクエスト (Request)] ペインで Python をクリックすると、クリップボード上のリクエストの Python 導入を取得できます。

NX-API 開発者サンドボックス: NX-OS リリース 9.2 (2) 以降

About the NX-API デベロッパー サンドボックス

Cisco NX-API Developer Sandbox は、スイッチでホストされる Web フォームです。NX-OS CLI コマンドを同等の XML または JSON ペイロードに変換し、NX-API REST ペイロードを同等の CLI に変換します。

Web フォームは、次の図に示すように、コマンド（上部のペイン）、要求（中央のペイン）、および応答（下部のペイン）の 3 つのペインを持つ 1 つの画面です。指定名 (DN) フィールドは、コマンドペインとリクエストペインの間にあります（下図の **POST** と送信オプションの間にあります）。

リクエストペインにも一連のタブがあります。各タブは、**Python**、**Python3**、**Java**、**JavaScript**、**Go-Lang** の異なる言語を表します。各タブでは、それぞれの言語でリクエストを表示できます。たとえば、CLI コマンドを XML または JSON ペイロードに変換した後、**[Python]** タブをクリックして、スクリプトの作成に使用できる Python でのリクエストを表示します。

図 2: リクエストと出力応答の例を含む NX-API デベロッパー サンドボックス

The screenshot displays the NX-API Sandbox interface. At the top, there's a header with the Cisco logo and 'NX-API Sandbox' title, along with links for 'Quick Start', 'Command Reference', and 'Logout'. The main area is divided into two sections: 'Request' and 'Response'. The 'Request' section shows a JSON payload for a REST API call to execute the 'show version' command. The 'Response' section shows the resulting JSON output from the device. The interface includes tabs for different languages (Python, Python3, Java, JavaScript, Go-Lang) and buttons for 'Send', 'Reset', and 'Output Schema'.

```
show version
```

Method: Message format: Input type:

POST /fms

Request Python Python3 Java JavaScript Go-Lang

```
{
  "jsonrpc": "2.0",
  "method": "cli_ascii",
  "params": {
    "cmd": "show version",
    "version": 1
  },
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": {
    "msg": "Cisco Nexus Operating System (NX-OS) Software\nTAC support: http://www.cisco.com/tac\nDocuments: http://www.cisco.com/en/US/products/ps9372/tsd_products_support_series_home.",
    "id": 1
  }
}
```

コマンドペインのコントロールを使用すると、NX-API REST などのサポートされている API、モデル（ペイロード）や CLI などの入力タイプ、および XML や JSON などのメッセージ形式を選択できます。使用可能なオプションは、選択した方法によって異なります。

NXAPI-REST（DME）メソッドを選択し、1 つ以上の CLI コマンドをコマンドペインに入力するか貼り付けて、**[変換]** をクリックすると、Web フォームはコマンドを REST API ペイロードに変換し、構成エラーをチェックし、要求ペインに結果のペイロードを表示します。次に、ペイロードをサンドボックスからスイッチに直接送信することを選択した場合（**POST** オプションを選択して **[SEND]** をクリック）、**[応答]** ペインに API 応答が表示されます。詳細については、[デベロッパー サンドボックスを使用して CLI コマンドを REST ペイロードに変換する（198 ページ）](#) を参照してください。

逆に、Cisco NX-API Developer Sandbox はペイロードの設定エラーをチェックし、対応する CLI を **[応答]** ペインに表示します。詳細については、「[デベロッパー サンドボックスを使用した REST ペイロードから CLI コマンドへの変換（201 ページ）](#)」を参照してください。

注意事項と制約事項

デベロッパー サンドボックスのガイドラインと制限は次のとおりです:

- サンドボックスで **[送信 (Send)]** をクリックすると、コマンドがスイッチにコミットされ、構成または状態が変更される可能性があります。
- 一部の機能構成コマンドは、関連する機能が有効になるまで使用できません。たとえば、BGP ルータを構成するには、最初に **[機能 bgp (feature bgp)]** コマンドを使用して BGP を有効にする必要があります。同様に、OSPF ルータを構成するには、最初に **[機能 ospf (feature ospf)]** コマンドを使用して OSPF を有効にする必要があります。これは、**[evpn マルチホーミング コア トラッキング (evpn multihoming core-tracking)]** などの依存コマンドを有効にする **[evpn esi マルチホーミング (evpn esi multihoming)]** にも適用されます。機能が機能依存コマンドにアクセスできるようにする方法の詳細については、[Cisco Nexus 9000 Configuration Guides](#) [Cisco Nexus 3000 Configuration Guides](#) を参照してください。
- サンドボックスを使用した DN での変換は、CLI 構成の DN を検索する場合にのみサポートされます。DME を使用して CLI 構成コマンドの DN を変換するなど、他のワークフローはサポートされていません。
- OSPFv2 インターフェイス コマンドのための、CLI からモデルまたは xml への変換は、**[no] ip router ospf <tag> area {<area-id-ip>|<area-id-int>} [secondaries none]** コマンドを使用してルータ インスタンスとエリアを構成し、インターフェイスで OSPF を明示的に有効にするまでは行われません。
- コマンド ペイン (上部のペイン) は、最大 10,000 行の入力をサポートします。
- CLI 入力のメッセージ タイプとして XML または JSON を使用する場合、セミコロンを使用して同じ行の複数のコマンドを区切ることができます。ただし、CLI 入力のメッセージ タイプとして JSON RPC を使用する場合、同じ行に複数のコマンドを入力し、セミコロン (;) で区切ることはできません。

たとえば、次のように JSON RPC を介して **show hostname** コマンドと **show clock** コマンドを送信したいとします。

Sandbox で、次のように CLI を入力します。

```
show hostname ; show clock
```

JSON RPC リクエストでは、入力は次のようにフォーマットされます。

```
[
  {
    "jsonrpc": "2.0",
    "method": "cli",
    "params": {
      "cmd": "show hostname ; show clock",
      "version": 1
    },
    "id": 1
  }
]
```

リクエストを送信すると、レスポンスで次のエラーが返されます。

```
{
  "jsonrpc": "2.0",
  "error": {
    "code": -32602,
    "message": "Invalid params",
    "data": {
      "msg": "Request contains invalid special characters"
    }
  },
  "id": 1
}
```

この状況は、Sandbox が JSONRPC リクエストの各コマンドを個別のアイテムとして解析し、それぞれに識別子を割り当てるために発生します。JSON RPC リクエストを使用する場合、同じ回線で複数のコマンドを区切るために内部句読点を使用することはできません。代わりに、各コマンドを個別の回線に入力すると、リクエストが正常に完了します。

同じ例を続けて、NX-API CLI で次のようにコマンドを入力します。

```
show hostname
show clock
```

リクエストでは、入力は次のようにフォーマットされます。

```
[
  {
    "jsonrpc": "2.0",
    "method": "cli",
    "params": {
      "cmd": "show hostname",
      "version": 1
    },
    "id": 1
  },
  {
    "jsonrpc": "2.0",
    "method": "cli",
    "params": {
      "cmd": "show clock",
      "version": 1
    },
    "id": 2
  }
]
```

応答は正常に完了します。

```
[
  {
    "jsonrpc": "2.0",
    "result": {
      "body": {
        "hostname": "switch-1"
      }
    },
    "id": 1
  },
  {
    "jsonrpc": "2.0",
    "result": {
      "body": {
        "simple_time": "12:31:02.686 UTC Wed Jul 10 2019\n",
        "time_source": "NTP"
      }
    }
  }
]
```

```

    },
    "id": 2
  }
]

```

メッセージフォーマットと入力タイプの構成

メソッド、メッセージ形式、および入力タイプは、コマンドペイン（上部のペイン）の右上隅で構成されます。**[メソッド]**で、使用する API プロトコルの形式を選択します。Cisco NX-API Developer Sandbox は、次の API プロトコルをサポートしています。

表 16: **NX-OS API** プロトコル

プロトコル	説明
NXAPI-CLI	XML または JSON ペイロードで NX-OS CLI または bash コマンドを配信するための Cisco NX-API 独自のプロトコル。
NXAPI-REST (DME)	内部 NX-OS データ管理エンジン（DME）モデルで管理対象オブジェクト（MO）とそのプロパティを操作および読み取るための Cisco NX-API 独自のプロトコル。NXAPI-REST（DME）プロトコルは、次の方法から選択できるドロップダウンリストを表示します。 • POST • GET • PUT • DELETE Cisco Nexus 3000 および 9000 シリーズ NX-API REST SDK の詳細については、https://developer.cisco.com/site/cisco-nexus-nx-api-references/ を参照してください。
RESTCONF (Yang)	構成および状態データ用の YANG（「Yet Another Next Generation」）データモデリング言語。 RESTCONF（Yang）プロトコルは、次の方法から選択できるドロップダウンリストを表示します。 • POST • GET • PUT • PATCH • DELETE

メソッドを選択すると、メッセージ形式または入力タイプのオプションのセットがドロップダウンリストに表示されます。メッセージ形式は、入力 CLI を制約し、要求と応答の形式を決定できます。オプションは、選択したメソッドによって異なります。

次の表では、各メッセージ形式の入力/コマンドタイプオプションについて説明します。

表 17: コマンドタイプ

方法	メッセージ形式	入力/コマンドタイプ
NXAPI-CLI	json-rpc	• <code>cli — show</code> または構成コマンド • <code>cli_ascii — show</code> または構成コマンド、フォーマットせずに出力 • <code>cli-array — show</code> コマンド。cli に似ていますが、<code>cli_array</code> を使用すると、データは 1 つの要素のリスト、または角括弧 [] で囲まれたアレイとして返されます。
NXAPI-CLI	xml	• <code>cli_show</code> — コマンドを表示します。コマンドが XML 出力をサポートしていない場合、エラーメッセージが返されます。 • <code>cli_show_ascii</code> — コマンドを表示、フォーマットせずに出力 • <code>cli_conf</code> — 構成 コマンド。対話型の構成コマンドはサポートされていません。 • <code>bash</code> — bash コマンド。ほとんどの非対話型 bash コマンドがサポートされています。 (注) スイッチで bash シェルを有効にする必要があります。

方法	メッセージ形式	入力/コマンドタイプ
NXAPI-CLI	json	• <code>cli_show</code> — コマンドを表示します。コマンドがXML出力をサポートしていない場合、エラーメッセージが返されます。 (注) Cisco NX-OS リリース 9.3(3) 以降では、<code>cli_show</code> コマンドよりも <code>cli_show_array</code> コマンドが推奨されます。 • <code>cli_show_array</code> — <code>show</code> コマンド <code>cli_show</code> に似ていますが、<code>cli_show_array</code> を使用すると、データは角括弧[]で囲まれた1つの要素のリストまたは配列として返されます。 • <code>cli_show_ascii</code> — コマンドを表示、フォーマットせずに出力 • <code>cli_conf</code> — 構成 コマンド。対話型の構成コマンドはサポートされていません。 • <code>bash</code> — <code>bash</code> コマンド。ほとんどの非対話型 <code>bash</code> コマンドがサポートされています。 (注) スイッチで <code>bash</code> シェルを有効にする必要があります。
NXAPI-REST (DME)		• <code>cli</code> — CLI からモデルへの変換 • <code>model</code> — モデルから CLI への変換。
RESTCONF (Yang)	• <code>json</code> — ペイロードにJSON構造が使用されます • <code>xml</code> — XML構造がペイロードに使用されます	

出力チャンク

JSON および XML NX-API メッセージ形式を使用すると、10 MB のチャンクで大きな `show` コマンド応答を受信できます。受信すると、チャンクが連結されて、有効な JSON オブジェクトまたは XML 構造が作成されます。出力チャンクを示すサンプルスクリプトを表示するには、次のリンクをクリックし、リリース 9.3x に対応するディレクトリを選択します: [Cisco NX-OS NXAPI](#)。



- (注) チャンク JSON モードの場合、ブラウザまたは Python スクリプト パーツは有効な JSON 出力を提供しません（終了タグはありません）。チャンク モードを使用して有効な JSON を取得するには、ディレクトリで提供されるスクリプトを使用します。

即時のコマンド応答で最初のチャンクを受け取ります。これには、セッション ID を含む **sid** フィールドも含まれます。次のチャンクを取得するには、前のチャンクのセッション ID を **[SID]** テキストボックスに入力します。**sid** フィールドの **eoc**（コンテンツの終わり）値で示される最後の応答に到達するまで、プロセスを繰り返します。

チャンク モードは、**JSON** または **XML** フォーマット タイプおよび **cli_show**、**cli_show_array**、または **cli_show_ascii** コマンド タイプで **NXAPI-CLI** メソッドを使用する場合に使用できます。チャンク モードの設定の詳細については、チャンク モードフィールドの表を参照してください。



- (注) NX-API は、最大 2 つのチャンク セッションをサポートします。

表 18: チャンク モード フィールド

フィールド名	説明
チャンク モードを有効にする	[チャンク モードを有効にする (Enable Chunk Mode)] チェックボックスをクリックしてチェックマークを付けると、チャンクが有効になります。チャンク モードを有効にすると、10 MB を超える応答は、最大 10 MB のサイズの複数のチャンクで送信されます。
SID	応答メッセージの次のチャンクを取得するには、 SID テキストボックスに前の応答のセッション ID を入力します。 (注) 使用できる文字は英数字と「_」のみです。無効な文字はエラーを受け取ります。

デベロッパー サンドボックスを使用

Cisco NX-API デベロッパー サンドボックス を使用して、次のような複数の変換を行うことができます。

デベロッパー サンドボックスを使用して CLI コマンドを REST ペイロードに変換する



ヒント

- Cisco NX-API デベロッパー サンドボックス ウィンドウの右上隅にあるフィールド名の横にあるヘルプ アイコン (?) をクリックすると、オンライン ヘルプを利用できます。
- 応答コードやセキュリティ メソッドなどの詳細については、*NX-API CLI* の章を参照してください。
- 構成コマンドはサポートされていません。

Cisco NX-API Developer Sandbox を使用すると、CLI コマンドを REST ペイロードに変換できます。

手順

-
- Step 1** [方法 (Method)] ドロップダウン リストをクリックし、**NXAPI-REST (DME)** を選択します。
[入力タイプ] ドロップダウン リストが表示されます。
- Step 2** [入力 (Input)] タイプドロップダウン リストをクリックし、**cli** を選択します。
- Step 3** 上部ペインのテキスト エントリ ボックスに、NX-OS CLI 構成コマンドを 1 行に 1 つずつ入力するか貼り付けます。
- 上部ペインの下部にある [リセット (Reset)] をクリックすると、テキスト エントリ ボックス (および [要求 (Request)] ペインと [応答 (Response)] ペイン) の内容を消去できます。

Enter DME payload here.

Method: NXAPI-REST (DME)

Input type: model

/api/mo/sys.json

Send Reset Convert

Request Python Python3 Java JavaScript Go-Lang

Copy

Response: Copy

Step 4 [変換（Convert）] をクリックします。

CLI コマンドに構成エラーが含まれていない場合、ペイロードは[要求（Request）]ペインに表示されます。エラーが存在する場合は、説明のエラーメッセージが[応答（Response）]ペインに表示されます。

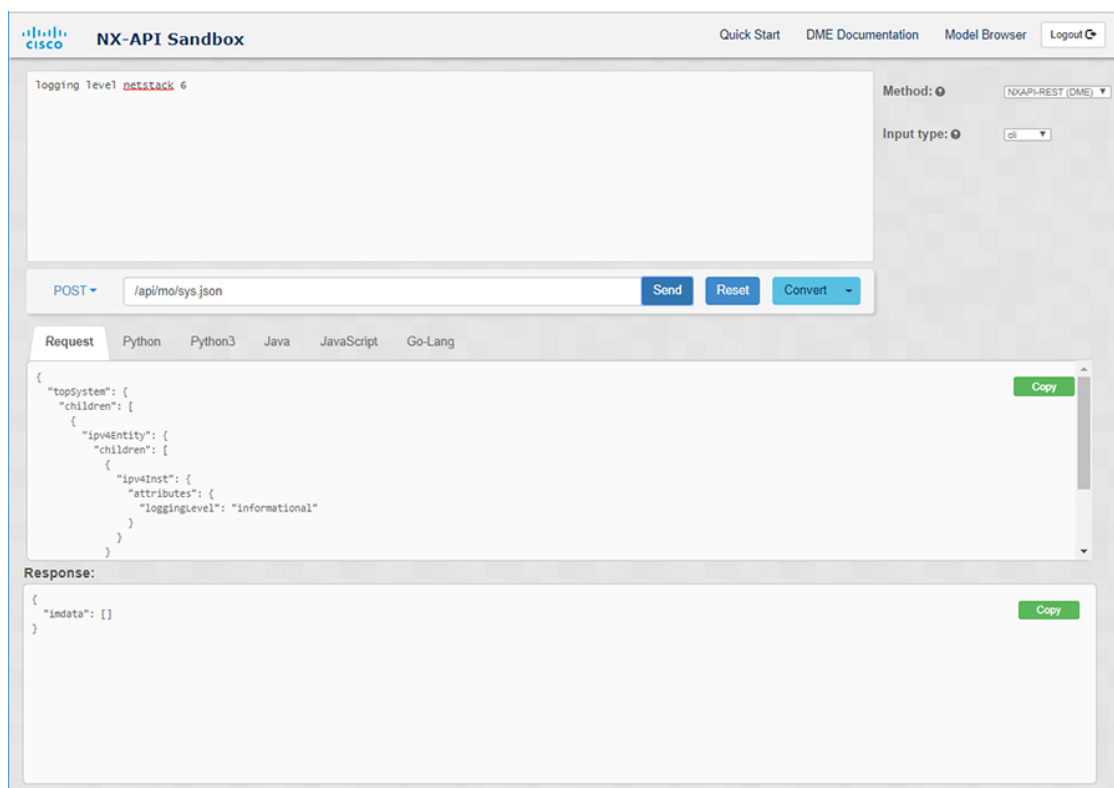
Step 5 (オプション) 有効なペイロードを API 呼び出しとしてスイッチに送信するには、[送信（Send）] をクリックします。

スイッチからのレスポンスは [Response（応答）] ペインに表示されます。

警告

[送信（Send）] をクリックすると、コマンドがスイッチにコミットされ、構成または状態が変更される可能性があります。

デベロッパー サンドボックスを使用して CLI コマンドを REST ペイロードに変換する



Step 6 (オプション) ペイロード内の MO の DN を取得するには:

1. [リクエスト (Request)] ペインから、**POST** を選択します。
2. [変換 (Convert)] ドロップダウン リストをクリックし、[変換 (DN を使用) (Convert (with DN))] を選択します。

ペイロードは、ペイロード内の各 MO に対応する DN を含む **dn** フィールドとともに表示されます。

Step 7 (オプション) 新しい構成で現在の構成を上書きする場合:

1. [変換 (Convert)] ドロップダウン リストをクリックし、[変換 (置換用) (Convert (for Replace))] を選択します。[リクエスト (Request)] ペインには、[ステータス (status)] フィールドが[置換 (replace)] ように設定されたペイロードが表示されます。
2. [リクエスト (Request)] ペインから、**POST** を選択します。
3. [送信 (Send)] をクリックします。

現在の構成は、投稿された構成に置き換えられます。たとえば、次の構成で開始するとします:

```
interface eth1/2
  description test
  mtu 1501
```

次に、[変換 (置換用) (Convert (for Replace))] を使用して、次の構成を POST します。

```
interface eth1/2
  description testForcr
```

mtu 構成が削除され、新しい説明 (testForcr) のみがインターフェイスの下に表示されます。この変更は、**show running-config** と入力すると確認されます。

Step 8 (オプション) [リクエスト (Request)] ペインや [応答 (Response)] ペインなどのペインの内容をコピーするには、[コピー (Copy)] をクリックします。それぞれのペインの内容がクリップボードにコピーされます。

Step 9 (オプション) リクエストを以下のいずれかのフォーマットに変換するには、[リクエスト (Request)] ペインの適切なタブをクリックします。

- Python
- python3
- Java
- JavaScript
- Go-Lang

デベロッパー サンドボックスを使用した REST ペイロードから CLI コマンドへの変換

Cisco NX-API Developer Sandbox を使用すると、REST ペイロードを対応する CLI コマンドに変換できます。このオプションは、NXAPI-REST (DME) メソッドでのみ使用できます。



ヒント

- Cisco NX-API Developer Sandbox のフィールド名の横にあるヘルプアイコン (?) をクリックすると、オンライン ヘルプを利用できます。ヘルプ アイコンをクリックして、それぞれのフィールドに関する情報を取得します。

応答コードやセキュリティ メソッドなどの詳細については、*NX-API CLI* の章を参照してください。
- Cisco NX-API Developer Sandbox の右上隅には、追加情報へのリンクが含まれています。表示されるリンクは、選択した[方法 (Method)]によって異なります。NXAPI-REST (DME) メソッドに表示されるリンク：
 - **[NX-API リファレンス (NX-API References)]**— 追加の NX-API ドキュメントにアクセスできます。
 - **[DME ドキュメント (DME Documentation)]**— NX-API DME モデル リファレンス ページにアクセスできます。
 - **[モデル ブラウザ (Model Browser)]**— モデル ブラウザである Visore にアクセスできます。Visore ページにアクセスするには、スイッチの IP アドレスを手動で入力する必要があります。ある場合があることに注意してください。

`https://management-ip-address/visore.html`。

手順

Step 1 [方法 (Method)] ドロップダウン リストをクリックし、**NXAPI-REST (DME)** を選択します。

例:

Step 2 [タイプを入力 (Input Type)] タイプドロップダウン リストをクリックし、[モデル (model)] を選択します。

Step 3 要求ペインの上にあるフィールドに、ペイロードに対応する指定名 (DN) を入力します。

Step 4 コマンド ペインにペイロードを入力します。

Step 5 [変換 (Convert)] をクリックします。

例:

この例では、DN は **/api/mo/sys.json** であり、NX-API REST ペイロードは次のとおりです。

```
{
  "topSystem": {
    "attributes": {
      "name": "REST2CLI"
    }
  }
}
```

デベロッパー サンドボックスを使用した REST ペイロードから CLI コマンドへの変換

The screenshot shows the NX-API Sandbox interface. At the top, there's a header with the Cisco logo and the title "NX-API Sandbox". Navigation links include "Quick Start", "DME Documentation", "Model Browser", and a "Logout" button. The main area is divided into two sections. The top section contains a JSON payload in a text area:

```
{  "topSystem": {    "attributes": {      "name": "RESTAPI"    }  }}
```

 To the right of the text area, there are two dropdown menus: "Method:" set to "NX-API-REST (DME)" and "Input type:" set to "model". Below the text area is a URL input field containing "/api/mo/sys.json", and three buttons: "Send" (blue), "Reset" (orange), and "Convert" (blue). Below the URL field is a tabbed interface with tabs for "Request", "Python", "Python3", "Java", "JavaScript", and "Go-Lang". The "Request" tab is active, showing a large empty text area for the request body, with a green "Copy" button in the top right corner. Below the request area is a "Response:" section, also with a large empty text area and a green "Copy" button in the top right corner.

[変換 (Convert)] ボタンをクリックすると、次の図に示すように、同等の CLI が CLI ペインに表示されます。

The screenshot shows the Cisco NX-API Sandbox web interface. At the top, there's a header with the Cisco logo and 'NX-API Sandbox' title, along with links for 'Quick Start', 'DME Documentation', 'Model Browser', and a 'Logout' button. The main area is divided into two sections. The top section contains a large text area with a JSON payload:

```
{  "aaaSystem": {    "attributes": {      "name": "REST2CLI"    }  }}
```

. To the right of this text area are two dropdown menus: 'Method:' set to 'NX-API REST (DME)' and 'Input type:' set to 'model'. Below the text area is a URL input field containing '/api/mo/sys.json', and three buttons: 'Send' (blue), 'Reset' (orange), and 'Convert' (blue). Below the URL field are tabs for 'Request', 'Python', 'Python3', 'Java', 'JavaScript', and 'Go-Lang'. The 'Request' tab is active, showing the text 'hostname REST2CLI' and a green 'Copy' button. Below this is a 'Response:' section, which is currently empty, with a green 'Copy' button on the right.

(注)

Cisco NX-API Developer Sandbox は、サンドボックスが CLI を NX-API REST ペイロードに変換した場合でも、すべてのペイロードを同等の CLI に変換できません。以下は、ペイロードが CLI コマンドに完全に変換するのを妨げる可能性のあるエラーの原因のリストです。

表 19: REST2CLI エラーの原因

ペイロードの問題	結果
ペイロードに、MO に存在しない属性が含まれています。 例: <pre>api/mo/sys.json { "topSystem": { "children": [{ "interfaceEntity": { "children": [{ "l1PhysIf": { "attributes": { "id": "eth1/1", "fakeattribute": "totallyFake" } } }] } }] } }</pre>	[エラー (Error)] ペインは、属性に関連するエラーを返します。 例: CLI 要素「l1PhysIf」の不明な属性「fakeattribute」の[エラー (Error)]
ペイロードには、変換がまだサポートされていない MO が含まれています。 例: <pre>api/mo/sys.json { "topSystem": { "children": [{ "dhcpEntity": { "children": [{ "dhcpInst": { "attributes": { "SnoopingEnabled": "yes" } } }] } }] } }</pre>	[エラー (Error)] ペインは、サポートされていない MO に関連するエラーを返します。 例: CLI [エラー (Error)] [「sys/dhcp」のサブツリー全体が変換されていません。(The entire subtree of "sys/dhcp" is not converted.)]

デベロッパー サンドボックスを使用して RESTCONF から json または XML に変換する



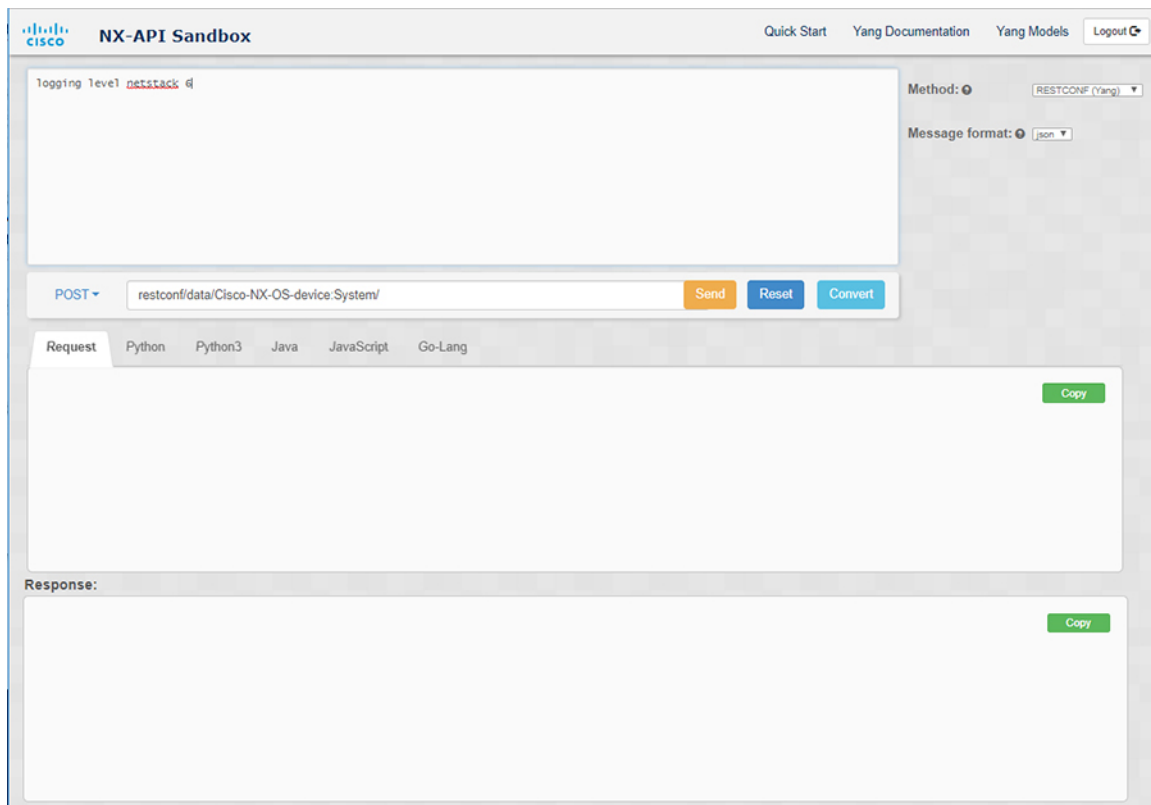
ヒント

- Cisco NX-API Developer Sandbox ウィンドウの右上隅にあるヘルプアイコン (?) をクリックすると、オンライン ヘルプを利用できます。
- [サンドボックス] ウィンドウの右上隅にある **Yang Documentation** リンクをクリックして、Model Driven Programmability with Yang ページに移動します。
- [サンドボックス] ウィンドウの右上隅にある **Yang Models** リンクをクリックして、YangModels GitHub サイトにアクセスします。

手順

Step 1 [メソッド] ドロップダウン リストをクリックし、[RESTCONF (Yang)] を選択します。

例:



Step 2 [メッセージ形式] をクリックし、json または xml を選択します。

Step 3 上部ペインのテキスト入力ボックスにコマンドを入力します。

Step 4 メッセージ形式を選択します。

デベロッパー サンドボックスを使用して RESTCONF から json または XML に変換する

Step 5 [変換（Convert）] をクリックします。

例:

この例では、コマンドはログ レベル **netstack 6** で、メッセージ形式は json です。

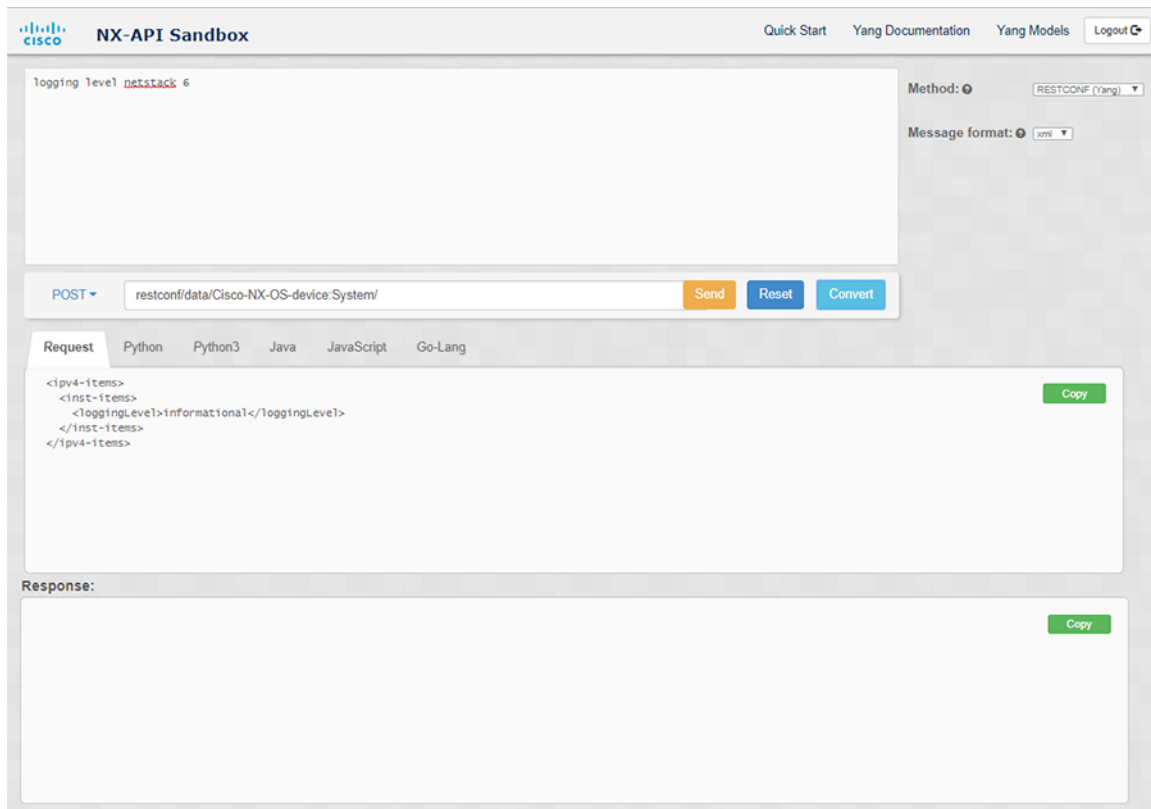
The screenshot shows the NX-API Sandbox interface. At the top, there's a header with the Cisco logo and 'NX-API Sandbox' title. Navigation links include 'Quick Start', 'Yang Documentation', 'Yang Models', and 'Logout'. The main area is divided into two sections: 'Request' and 'Response'. In the 'Request' section, the 'Method' is set to 'RESTCONF (Yang)' and the 'Message format' is set to 'json'. The 'Request' tab is active, showing a JSON payload:

```
{  "ipv4-items": {    "inst-items": {      "loggingLevel": "informational"    }  } }
```

. Below the request, there's a 'Response' section which is currently empty. The interface also includes a 'Send' button, a 'Reset' button, and a 'Convert' button. The 'Convert' button is highlighted in blue.

例:

この例では、コマンドはログ レベル **netstack 6** で、メッセージ形式は xml です。



(注)

XML または JSON メッセージ形式を使用して、否定された CLI を Yang ペイロードに変換すると、サンドボックスは警告をスローし、**[送信]** オプションを無効にします。表示される警告メッセージは、メッセージの形式によって異なります。

- XML メッセージ形式の場合 — 「これは Netconf ペイロードであり、DELETE 操作用に生成されているため、Restconf では SEND オプションが無効になっています!」
- JSON メッセージ形式の場合 - 「これは、DELETE 操作用に生成される gRPC ペイロードであるため、Restconf では SEND オプションが無効になっています!」

Step 6 **[リクエスト]** ペインの適切なタブをクリックして、リクエストを次の形式に変換することもできます。

- Python
- python3
- Java
- JavaScript
- Go-Lang

(注)

[リクエスト] タブの上の領域にあるドロップダウンメニューから [PATCH] オプションを選択した場合、Java で生成されたスクリプトは機能しません。これは Java の既知の制限であり、予期される動作です。



第 17 章

N3500 での ABM および LM の XML サポート

• [N3500 での ABM および LM の XML サポート](#) (211 ページ)

N3500 での ABM および LM の XML サポート

次のコマンドは、ABM および LM の XML 出力を表示します：

[ハードウェア プロファイル バッファ モニタ サンプリングを表示します (show hardware profile buffer monitor sampling)]

CLI :

```
MTC-8(config)# show hardware profile buffer monitor sampling
```

```
Sampling CLI issued at: 05/25/2016 04:18:56
```

```
Sampling interval: 200
```

XML :

```
MTC-8(config)# show hardware profile buffer monitor sampling | xml
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<nf:rpc-reply xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns="http://www.cisco.com/nxos:1.0:mtc_usd_cli">
```

```
<nf:data>
```

```
<show>
```

```
<hardware>
```

```
<profile>
```

```
<buffer>
```

```
<monitor>
```

```
<__XML__BLK_Cmd_show_hardware_profile_buffer_monitor_summary>
```

```
<__XML__OPT_Cmd_show_hardware_profile_buffer_monitor__readonly__>
```

```
<__readonly__>
```

```

<cmd_name>Sampling CLI</cmd_name>

<cmd_issue_time>05/25/2016 04:19:12</cmd_issue_time>

<TABLE_sampling>

  <ROW_sampling>

    <sampling_interval>200</sampling_interval>

  </ROW_sampling>

</TABLE_sampling>

</__readonly__>

</__XML__OPT_Cmd_show_hardware_profile_buffer_monitor__readonly__>

</__XML__BLK_Cmd_show_hardware_profile_buffer_monitor_summary>

</monitor>

</buffer>

</profile>

</hardware>

</show>

</nf:data>

</nf:rpc-reply>

]]>]]>

```

[ハードウェア プロファイル バッファ モニタの詳細を表示します | xml (show hardware profile buffer monitor detail | xml)]

XML :

```

<show>
  <hardware>
    <profile>
      <buffer>
        <monitor>
          <__XML__BLK_Cmd_show_hardware_profile_buffer_monitor_summary>
            <__XML__OPT_Cmd_show_hardware_profile_buffer_monitor__readonly__>
              <__readonly__>
                <cmd_name>Detail CLI</cmd_name>
                <cmd_issue_time>10/02/2001 10:58:58</cmd_issue_time>
                <TABLE_detail_entry>
                  <ROW_detail_entry>
                    <detail_util_name>Ethernet1/1</detail_util_name>
                    <detail_util_state>Active</detail_util_state>
                  </ROW_detail_entry>
                </TABLE_detail_entry>
                <time_stamp>10/02/2001 10:58:58</time_stamp>
                <__XML__DIGIT384k_util>0</__XML__DIGIT384k_util>
                <__XML__DIGIT768k_util>0</__XML__DIGIT768k_util>
                <__XML__DIGIT1152k_util>0</__XML__DIGIT1152k_util>
                <__XML__DIGIT1536k_util>0</__XML__DIGIT1536k_util>
                <__XML__DIGIT1920k_util>0</__XML__DIGIT1920k_util>
              </__readonly__>
            </__XML__OPT_Cmd_show_hardware_profile_buffer_monitor__readonly__>
          </__XML__BLK_Cmd_show_hardware_profile_buffer_monitor_summary>
        </monitor>
      </buffer>
    </profile>
  </hardware>
</show>

```


Cisco Nexus 3500 Series NX-OS Programmability Guide, Release 10.5(x)

```

<__XML__DIGIT4992k_util>0</__XML__DIGIT4992k_util>
<__XML__DIGIT5376k_util>0</__XML__DIGIT5376k_util>
<__XML__DIGIT5760k_util>0</__XML__DIGIT5760k_util>
<__XML__DIGIT6144k_util>0</__XML__DIGIT6144k_util>
</ROW_detail_entry>
<ROW_detail_entry>
  <time_stamp>10/02/2001 10:58:54</time_stamp>
  <__XML__DIGIT384k_util>0</__XML__DIGIT384k_util>
  <__XML__DIGIT768k_util>0</__XML__DIGIT768k_util>
  <__XML__DIGIT1152k_util>0</__XML__DIGIT1152k_util>
  <__XML__DIGIT1536k_util>0</__XML__DIGIT1536k_util>
  <__XML__DIGIT1920k_util>0</__XML__DIGIT1920k_util>
  <__XML__DIGIT2304k_util>0</__XML__DIGIT2304k_util>
  <__XML__DIGIT2688k_util>0</__XML__DIGIT2688k_util>
  <__XML__DIGIT3072k_util>0</__XML__DIGIT3072k_util>
  <__XML__DIGIT3456k_util>0</__XML__DIGIT3456k_util>
  <__XML__DIGIT3840k_util>0</__XML__DIGIT3840k_util>
  <__XML__DIGIT4224k_util>0</__XML__DIGIT4224k_util>
  <__XML__DIGIT4608k_util>0</__XML__DIGIT4608k_util>
  <__XML__DIGIT4992k_util>0</__XML__DIGIT4992k_util>
  <__XML__DIGIT5376k_util>0</__XML__DIGIT5376k_util>
  <__XML__DIGIT5760k_util>0</__XML__DIGIT5760k_util>
  <__XML__DIGIT6144k_util>0</__XML__DIGIT6144k_util>
</ROW_detail_entry>

```

[ハードウェア プロファイル バッファ モニタの概要を表示します (show hardware profile buffer monitor brief)]

XML :

```

show hardware profile buffer monitor brief | xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<nf:rpc-reply xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns="http://www.cisco.com/nxos:1.0:mtc_usd_cli">
  <nf:data>
    <show>
      <hardware>
        <profile>
          <buffer>
            <monitor>
              <__XML__BLK_Cmd_show_hardware_profile_buffer_monitor_summary>
                <__XML__OPT_Cmd_show_hardware_profile_buffer_monitor__readonly__>
                  <__readonly__>
                    <cmd_name>Brief CLI</cmd_name>
                    <cmd_issue_time>03/21/2016 09:06:38</cmd_issue_time>
                    <TABLE_ucst_hdr>
                      <ROW_ucst_hdr>
                        <ucst_hdr_util_name>Buffer Block 1</ucst_hdr_util_name>
                        <ucst_hdr_1sec_util>0KB</ucst_hdr_1sec_util>
                        <ucst_hdr_5sec_util>0KB</ucst_hdr_5sec_util>
                        <ucst_hdr_60sec_util>N/A</ucst_hdr_60sec_util>
                        <ucst_hdr_5min_util>N/A</ucst_hdr_5min_util>
                        <ucst_hdr_1hr_util>N/A</ucst_hdr_1hr_util>
                        <ucst_hdr_total_buffer>Total Shared Buffer Available = 5397 Kbytes
                      </ucst_hdr_total_buffer>
                        <ucst_hdr_class_threshold>Class Threshold Limit = 5130 Kbytes
                      </ucst_hdr_class_threshold>
                    </ROW_ucst_hdr>
                  </TABLE_ucst_hdr>
                  <TABLE_brief_entry>
                    <ROW_brief_entry>
                      <brief_util_name>Ethernet1/45</brief_util_name>
                      <brief_1sec_util>0KB</brief_1sec_util>
                      <brief_5sec_util>0KB</brief_5sec_util>

```

Cisco Nexus 3500 Series NX-OS Programmability Guide, Release 10.5(x)

```

<brief_1hr_util>N/A</brief_1hr_util>
<brief_util_name>Ethernet1/12</brief_util_name>
<brief_1sec_util>0KB</brief_1sec_util>
<brief_5sec_util>0KB</brief_5sec_util>
<brief_60sec_util>N/A</brief_60sec_util>
<brief_5min_util>N/A</brief_5min_util>
<brief_1hr_util>N/A</brief_1hr_util>

```

[ハードウェア プロファイルの遅延モニタ サンプリングを表示します (show hardware profile latency monitor sampling)]

CLI

```
MTC-8(config)# show hardware profile latency monitor sampling
```

```
Sampling CLI issued at: 05/25/2016 04:19:54
```

```
Sampling interval: 20
```

XML

```
MTC-8(config)# show hardware profile latency monitor sampling | xml
```

```

<?xml version="1.0" encoding="ISO-8859-1"?>

<nf:rpc-reply xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns="http://www.cisco.com/nxos:1.0:mtc_usd_cli">

  <nf:data>

    <show>

      <hardware>

        <profile>

          <latency>

            <monitor>

              <__XML__BLK_Cmd_show_hardware_profile_latency_monitor_summary>

                <__XML__OPT_Cmd_show_hardware_profile_latency_monitor__readonly__>

                  <__readonly__>

                    <cmd_issue_time>05/25/2016 04:20:06</cmd_issue_time>

                    <device_instance>0</device_instance>

                    <TABLE_sampling>

                      <ROW_sampling>

                        <sampling_interval>20</sampling_interval>

                      </ROW_sampling>

                    </TABLE_sampling>

                  </__readonly__>

                </__XML__OPT_Cmd_show_hardware_profile_latency_monitor__readonly__>

```

```

        </__XML__BLK_Cmd_show_hardware_profile_latency_monitor_summary>
    </monitor>
</latency>
</profile>
</hardware>
</show>
</nf:data>
</nf:rpc-reply>
]]>]]>

```

[ハードウェア プロファイルの遅延モニタのしきい値を表示します (how hardware profile latency monitor threshold)]

CLI

```

MTC-8(config)# show hardware profile latency monitor threshold

Sampling CLI issued at: 05/25/2016 04:20:53

Threshold Avg: 3000

Threshold Max: 300000

```

XML

```

MTC-8(config)# show hardware profile latency monitor threshold | xml

<?xml version="1.0" encoding="ISO-8859-1"?>

<nf:rpc-reply xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns="http://www.cisco.com/nxos:1.0:mtc_usd_cli">

  <nf:data>

    <show>

      <hardware>

        <profile>

          <latency>

            <monitor>

              <__XML__BLK_Cmd_show_hardware_profile_latency_monitor_summary>

                <__XML__OPT_Cmd_show_hardware_profile_latency_monitor__readonly__>

                  <__readonly__>

                    <cmd_issue_time>05/25/2016 04:21:04</cmd_issue_time>

                    <device_instance>0</device_instance>

                    <TABLE_threshold>

```

```
<ROW_threshold>

  <threshold_avg>3000</threshold_avg>

  <threshold_max>300000</threshold_max>

</ROW_threshold>

</TABLE_threshold>

</__readonly__>

</__XML__OPT_Cmd_show_hardware_profile_latency_monitor__readonly__>

</__XML__BLK_Cmd_show_hardware_profile_latency_monitor_summary>

</monitor>

</latency>

</profile>

</hardware>

</show>

</nf:data>

</nf:rpc-reply>

]]>]]>
```



第 18 章

CLI コマンドのネットワーク構成フォーマットへの変換

- [XMLIN に関する情報（219 ページ）](#)
- [XMLIN のライセンス要件（219 ページ）](#)
- [XMLIN ツールのインストールおよび使用（220 ページ）](#)
- [show コマンド出力の XML への変換（221 ページ）](#)
- [XMLIN の構成例（221 ページ）](#)

XMLIN に関する情報

XMLIN ツールは、CLI コマンドをネットワーク構成（NETCONF）プロトコル形式に変換します。NETCONF は、ネットワーク デバイスの構成をインストール、処理、削除する機能を提供するネットワーク管理プロトコルです。これは、構成データとプロトコル メッセージに XML ベースのエンコーディングを使用します。NETCONF プロトコルの NX-OS 実装は、<get>、<edit-config>、<close-session>、<kill-session>、および <exec-command> のプロトコル操作をサポートします。

XMLIN ツールは、show、EXEC、および構成コマンドを対応する NETCONF <get>、<exec-command>、および <edit-config> リクエストに変換します。複数の構成コマンドを単一の NETCONF <edit-config> インスタンスにまとめることができます。

XMLIN ツールはまた、show コマンドの出力を XML 形式に変換します。

XMLIN のライセンス要件

表 20: XMLIN ライセンス要件

製品	ライセンス要件

Cisco NX-OS	XMLIN にはライセンスは必要ありません。ライセンス パッケージに含まれていない機能はすべて Cisco NX-OS システム イメージにバンドルされており、追加費用は一切発生しません。NX-OS ライセンス方式の詳細については、『 <i>Cisco NX-OS Licensing Guide</i> 』を参照してください。

XMLIN ツールのインストールおよび使用

XMLIN ツールをインストールして、構成 コマンドを NETCONF フォーマットに変換するために使用できます。

始める前に

XMLIN ツールは、対応する機能セットまたは必要なハードウェア機能がデバイスで利用できない場合でも、コマンドの NETCONF インスタンスを生成できます。ただし、**xmlin** コマンドを入力する前に、いくつかの機能セットをインストールする必要がある場合があります。

手順の概要

1. switch# **xmlin**
2. switch(xmlin)# **configure terminal**
3. コンフィギュレーション コマンド
4. （任意） switch(config)(xmlin)# **end**
5. （任意） switch(config-if-verify)(xmlin)# **show commands**
6. （任意） switch(config-if-verify)(xmlin)# **exit**

手順の詳細

手順

	コマンドまたはアクション	目的
Step 1	switch# xmlin	
Step 2	switch(xmlin)# 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
Step 3	コンフィギュレーション コマンド	構成 コマンドを NETCONF フォーマットに変換します。
Step 4	（任意） switch(config)(xmlin)# end	対応する <edit-config> 要求を生成します。 （注） show コマンドに対して XML インスタンスを生成する前に、 end コマンドを入力して現在の XML 構成を終了する必要があります。

	コマンドまたはアクション	目的
Step 5	(任意) switch(config-if-verify)(xmlin)# show commands	show コマンドを NETCONF フォーマットに変換します。
Step 6	(任意) switch(config-if-verify)(xmlin)# exit	EXEC モードに戻ります。

show コマンド出力の XML への変換

show コマンドの出力を XML に変換できます。

始める前に

変換するコマンドのすべての機能がインストールされ、デバイス上で有効になっていることを確認します。そうしない場合、コマンドは機能不全になります。

terminal verify-only コマンドを使用すると、デバイスに入力しなくても機能が有効になっていることを確認できます。

コマンドに対するすべての必須ハードウェアがデバイス上に存在することを確認します。そうしない場合、コマンドは機能不全になります。

XMLIN ツールがインストールされていることを確認します。

手順の概要

1. switch# *show-command* | **xmlin**

手順の詳細

手順

	コマンドまたはアクション	目的
Step 1	switch# <i>show-command</i> xmlin	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 (注) 構成コマンドと一緒にこのコマンドを使用することはできません。

XMLIN の構成例

次の例は、XMLIN ツールがデバイス上にどのようにインストールされ、一連の構成コマンドを <edit-config> インスタンスに変換するためにどのように使用されるかを示しています。

```

switch# xmlin
*****
Loading the xmlin tool. Please be patient.
*****
Cisco Nexus Operating System (NX-OS) Software
TAC support: http://www.cisco.com/tac
Copyright ©) 2002-2013, Cisco Systems, Inc. All rights reserved.
The copyrights to certain works contained in this software are
owned by other third parties and used and distributed under
license. Certain components of this software are licensed under
the GNU General Public License (GPL) version 2.0 or the GNU
Lesser General Public License (LGPL) Version 2.1. A copy of each
such license is available at
http://www.opensource.org/licenses/gpl-2.0.php and
http://www.opensource.org/licenses/lgpl-2.1.php

switch(xmlin)# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
switch(config)(xmlin)# interface ethernet 2/1
% Success
switch(config-if-verify)(xmlin)# cdp enable
% Success
switch(config-if-verify)(xmlin)# end
<?xml version="1.0"?>
<nf:rpc xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns="http://www.cisco.com/nxos:6.2.2.:configure_"
xmlns:m="http://www.cisco.com/nxos:6.2.2.:_exec"
xmlns:m1="http://www.cisco.com/nxos:6.2.2.:configure__if-eth-base" message-id="1">
  <nf:edit-config>
    <nf:target>
      <nf:running/>
    </nf:target>
    <nf:config>
      <m:configure>
        <m:terminal>
          <interface>
            <__XML_PARAM_interface>
              <__XML_value>Ethernet2/1</__XML_value>
              <m1:cdp>
                <m1:enable/>
              </m1:cdp>
            </__XML_PARAM_interface>
          </interface>
        </m:terminal>
      </m:configure>
    </nf:config>
  </nf:edit-config>
</nf:rpc>
]]>]]>

```

次の例は、**show** コマンドに対して XML インスタンスを生成する前に現在の XML 構成を終了するために **end** コマンドを入力する方法を表示しています。

```

switch(xmlin)# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
switch(config)(xmlin)# interface ethernet 2/1
switch(config-if-verify)(xmlin)# show interface ethernet 2/1
*****
Please type "end" to finish and output the current XML document before building a new one.
*****
% Command not successful

```

```

switch(config-if-verify) (xmlin) # end
<?xml version="1.0"?>
<nf:rpc xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns="http://www.cisco.com/nxos:6.2.2.:configure_"
xmlns:m="http://www.cisco.com/nxos:6.2.2.:_exec" message-id="1">
  <nf:edit-config>
    <nf:target>
      <nf:running/>
    </nf:target>
    <nf:config>
      <m:configure>
        <m:terminal>
          <interface>
            <__XML__PARAM__interface>
              <__XML__value>Ethernet2/1</__XML__value>
            </__XML__PARAM__interface>
          </interface>
        </m:terminal>
      </m:configure>
    </nf:config>
  </nf:edit-config>
</nf:rpc>
]]>]]>

switch(xmlin) # show interface ethernet 2/1
<?xml version="1.0"?>
<nf:rpc xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns="http://www.cisco.com/nxos:6.2.2.:if_manager" message-id="1">
  <nf:get>
    <nf:filter type="subtree">
      <show>
        <interface>
          <__XML__PARAM__ifeth>
            <__XML__value>Ethernet2/1</__XML__value>
          </__XML__PARAM__ifeth>
        </interface>
      </show>
    </nf:filter>
  </nf:get>
</nf:rpc>
]]>]]>
switch(xmlin) # exit
switch#

```

次の例は、**show interface brief** コマンドの出力を XML に変換する方法を示しています。

```

switch# show interface brief | xmlin
<?xml version="1.0"?>
<nf:rpc xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns="http://www.cisco.com/nxos:6.2.2.:if_manager"

message-id="1">
  <nf:get>
    <nf:filter type="subtree">
      <show>
        <interface>
          <brief/>
        </interface>
      </show>
    </nf:filter>
  </nf:get>
</nf:rpc>
]]>]]>

```




CHAPTER 19

OpenConfig YANG

This section contains the following topics:

- [About OpenConfig YANG, on page 225](#)
- [Guidelines and Limitations for OpenConfig YANG, on page 225](#)
- [Understanding Deletion of BGP Routing Instance, on page 233](#)
- [Verifying YANG, on page 235](#)
- [Enabling OpenConfig Support, on page 235](#)

About OpenConfig YANG

OpenConfig YANG supports modern networking principles, such as declarative configuration and model-driven management and operations. OpenConfig provides vendor-neutral data models for configuration and monitoring of the network. And, helping with moving from a pull model to a push model, with subscriptions and event update streaming.

Beginning with Cisco NX-OS Release 9.2(1), support is added across a broad range of functional areas. Those include BGP, OSPF, Interface L2 and L3, VRFs, VLANs, and TACACs.

CPU Utilization is the sum of user and kernel utilization values. This value is approximately equal to the sum of user and kernel percentages that are displayed in “show system resources” output.

The min, max, average, min time and max time values are calculated over a 30 second moving window. In this 30-second window, CPU utilization is calculated every 5 seconds. Min, max and average values are calculated over these six samples.

The time instant at which the minimum and maximum values occur within the last 30-second window, are displayed in epoch time format.

For additional information about OpenConfig YANG, see [About OpenConfig YANG](#).

For the OpenConfig models for Cisco NX-OS 9.2(1), see [YANG Models 9.2\(1\)](#). OpenConfig YANG models are grouped by Cisco NX-OS release, so when the Cisco NX-OS release number changes, the last digits in the URL change.

Guidelines and Limitations for OpenConfig YANG

OpenConfig YANG has the following guidelines and limitations:

- For IPv4 and IPv6 addresses, you must provide the same operation for remove and delete for the IP address field (**oc-ip:ip** and **oc-ip:prefix_length**).

For example:

```
oc-ip:ip: remove
oc-ip:prefix_length: remove
```

- Configuring BGP actions with **set med** and OSPF actions with metric in the same route-map via OpenConfig NETCONF is not recommended as the OSPF actions metric takes precedence over BGP **set med** property.

Use two different route-maps to set metrics under OSPF actions. Use **set-med** under BGP actions using separate route-maps.

We recommended that you do not change the metric of BGP actions to OSPF actions or OSPF actions to BGP actions of a route-map in a single payload.

- In order to have a valid BGP instance, an autonomous system (AS) number must be provided. Since there cannot be a default value for an AS number, any attempt to delete in NETCONF/OPENCONFIG <asn> without removing the BGP instance, results in the following highlighted error message:

```
764
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
message-id="urn:uuid:1ea09de2-605e-46aa-984b-9dfdada03354d">
  <nc:edit-config>
    <nc:target>
      <nc:running/>
    </nc:target>
    <nc:config>
      <network-instances xmlns="http://openconfig.net/yang/network-instance">
        <network-instance>
          <name>default</name>
          <protocols>
            <protocol>
              <identifier>BGP</identifier>
              <name>bgp</name>
              <bgp>
                <global>
                  <config nc:operation="delete">
                    <as>100</as>
                  </config>
                </global>
                <neighbors>
                  <neighbor>
                    <neighbor-address>1.1.1.1</neighbor-address>
                    <enable-bfd xmlns="http://openconfig.net/yang/bfd">
                      <config>
                        <enabled>true</enabled>
                      </config>
                    </enable-bfd>
                  </neighbor>
                </neighbors>
              </bgp>
            </protocol>
          </protocols>
        </network-instance>
      </network-instances>
    </nc:config>
  </nc:edit-config>
</nc:rpc>

##
```

```

Received:
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
message-id="urn:uuid:1ea09de2-605e-46aa-984b-9dfdad03354d">
  <rpc-error>
    <error-type>protocol</error-type>
    <error-tag>operation-failed</error-tag>
    <error-severity>error</error-severity>
    <error-message xml:lang="en">invalid property value , for property asn, class
bgpInst</error-message>
    <error-path>/config/network-instances</error-path>
  </rpc-error>
  <rpc-error>
    <error-type>protocol</error-type>
    <error-tag>operation-failed</error-tag>
    <error-severity>error</error-severity>
    <error-message xml:lang="en">invalid property value , for property asn, class
bgpInst Commit Failed</error-message>
    <error-path>/config/network-instances</error-path>
  </rpc-error>
</rpc-reply>

```

- The following OpenConfig YANG limitations exist for OC-BGP-POLICY:
 - Action type is always permit for community-set and as-path-set, which applies to the following containers:
 - /bgp-defined-sets/community-sets/community-set/
 - /bgp-defined-sets/as-path-sets/as-path-set/

In OpenConfig YANG, there is no action type concept as there is in the CLI for community-set and as-path-set. Therefore, the action type is always permit for community-set and as-path-set.

- The following OpenConfig YANG limitation applies to this container:
 - /bgp-defined-sets/community-sets/community-set/

In the CLI, community-list can have two different types: standard and expanded. However, in the OpenConfig YANG model, community-set-name has no such differentiation.

When you create the community-set-name through OpenConfig YANG, the following things happen internally:

- The `_std` suffix will be appended after community-set-name if community-member is in the standard form (AS:NN).
- The `_exp` suffix will be appended after community-set-name if community-member is in the expanded form (regex):

```

<community-set>
  <community-set-name>oc_commset1d</community-set-name>
  <config>
    <community-set-name>oc_commset1d</community-set-name>
    <community-member>0:1</community-member>
    <community-member>_1_</community-member>
  </config>
</community-set>

```

The preceding OpenConfig YANG configuration is mapped to the following CLI:

```
ip community-list expanded oc_commsetld_exp seq 5 permit "_1_"
ip community-list standard oc_commsetld_std seq 5 permit 0:1
```

- The following OpenConfig YANG limitation applies to this container:
/bgp-conditions/match-community-set/config/community-set/

OpenConfig YANG can only map to one community-set, while the CLI can match to multiple instances of the community-set:

- In the CLI:

```
ip community-list standard 1-1 seq 1 permit 1:1
ip community-list standard 1-2 seq 1 permit 1:2
ip community-list standard 1-3 seq 1 permit 1:3
route-map To_LC permit 10
match community 1-1 1-2 1-3
```

- The corresponding OpenConfig YANG payload follows:

```
<config>
  <routing-policy xmlns="http://openconfig.net/yang/routing-policy">
    <defined-sets>
      <bgp-defined-sets xmlns="http://openconfig.net/yang/bgp-policy">
        <community-sets>
          <community-set>
            <community-set-name>cs</community-set-name>
            <config>
              <community-set-name>cs</community-set-name>
              <community-member>1:1</community-member>
              <community-member>1:2</community-member>
              <community-member>1:3</community-member>
            </config>
          </community-set>
        </community-sets>
      </bgp-defined-sets>
    </defined-sets>
    <policy-definitions>
      <policy-definition>
        <name>To_LC</name>
        <statements>
          <statement>
            <name>10</name>
            <conditions>
              <bgp-conditions xmlns="http://openconfig.net/yang/bgp-policy">
                <match-community-set>
                  <config>
                    <community-set>cs</community-set>
                  </config>
                </match-community-set>
              </bgp-conditions>
            </conditions>
          </statement>
        </statements>
      </policy-definition>
    </policy-definitions>
  </routing-policy>
</config>
```

As a workaround, create one community with multiple statements through OpenConfig YANG:


```

ip community-list standard cs_std seq 5 permit 1:1
ip community-list standard cs_std seq 10 permit 1:2
ip community-list standard cs_std seq 15 permit 1:3
route-map To_LC permit 10
match community cs_std

```

- The following OpenConfig YANG limitation applies to this container:

/bgp-conditions/state/next-hop-in

In OpenConfig YANG, the `next-hop-in` type is an IP address, but in the CLI, it is an IP prefix.

While creating the `next-hop-in` through OpenConfig YANG, the IP address is converted to a "/32" mask prefix in the CLI configuration. For example:

- Following is an example of `next-hop-in` in the OpenConfig YANG payload:

```

<policy-definition>
  <name>sc0</name>
  <statements>
    <statement>
      <name>5</name>
      <conditions>
        <bgp-conditions xmlns="http://openconfig.net/yang/bgp-policy">
          <config>
            <next-hop-in>2.3.4.5</next-hop-in>
          </config>
        </bgp-conditions>
      </conditions>
    </statement>
  </statements>
</policy-definition>

```

- Following is an example of the same information in the CLI:

```

ip prefix-list IPV4_PFX_LIST_OPENCONFIG_sc0_5 seq 5 permit 2.3.4.5/32
route-map sc0 permit 5
match ip next-hop prefix-list IPV4_PFX_LIST_OPENCONFIG_sc0_5

```

- The following NX-OS limitations exist for OC-BGP-POLICY:

- /bgp-actions/set-community/config/method enum "REFERENCE" is not supported.
- enum "SELF", which is supported in the OpenConfig YANG model for /bgp-actions/config/set-next-hop, is not supported.

- For OC-BGP-POLICY,

/bgp-conditions/match-community-set/config/community-set get mapped only to `match community <community-set>_std`, so only standard community is supported. Match to expanded community set is not supported.

- There is a limitation in replacing `match-tag-set` because defined sets for tag-sets are not currently implemented.

Currently, replacing `match-tag-set` appends the values. To replace `match-tag-set`, delete it, then create it again.

- The following guidelines and limitations apply to OSPF OpenConfig YANG:

- If you configure and remove an area configuration in OSPF, the deleted areas (stale entries) are still shown in DME. Those stale area entries are shown in the GETCONFIG/GET output in OpenConfig YANG.
- Only one area is supported in OpenConfig YANG in the OSPF policy match `ospf-area` configuration. In the CLI, you can configure to match multiple areas, such as `match ospf-area 100 101`. However, in OpenConfig YANG, you can configure only one area (for example, `match ospf-area 100`).
- The area virtual-link and area interface configurations payload cannot go under the same area list. Split the area container payload as a Virtual link area and interface area in the same payload.
- The MD5 authentication string cannot be configured in OSPF OpenConfig YANG.

In the OSPF model, `Authentication-type` is defined for the Authentication:

```
leaf authentication-type {
  type string;
  description
    "The type of authentication that should be used on this
    interface";
}
```

OSPF OpenConfig YANG does not support an option for authentication password.

- The OSPF area authentication configuration is not supported. For example, `area 0.0.0.200 authentication message-digest` cannot be configured from OpenConfig YANG.
- The OSPF/BGP instance configuration that falls under default VRF (for example, **router ospf 1/router bgp 1**) is not deleted when you delete the Protocols container with the default network instance.
- The following are guidelines and limitations for VLAN configuration between the OpenConfig payload and the Cisco Nexus 9000 interfaces:

- When you attempt to simultaneously configure a trunk-mode interface and trunk VLANs in the same OpenConfig payload, the configuration does not complete successfully. However, when you split the payload so that the trunk-mode interface is sent first, then the trunk VLANs are sent, the configuration completes successfully.

On Cisco NX-OS interfaces, the default interface mode is **access**. To implement any trunk-related configurations, you must first change the interface mode to **trunk**, then configure the trunk VLAN ranges. Do these configurations in separate payloads.

The following examples show the separate payloads for the configuring trunk mode and VLAN ranges.

Example 1, payload configuring the interface to trunk mode.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <interfaces xmlns="http://openconfig.net/yang/interfaces">
        <interface>
          <name>eth1/47</name>
          <subinterfaces>
            <subinterface>
              <index>0</index>
```

```

        <config>
          <index>0</index>
        </config>
      </subinterface>
    </subinterfaces>
    <ethernet xmlns="http://openconfig.net/yang/interfaces/ethernet">
      <switched-vlan xmlns="http://openconfig.net/yang/vlan">
        <config>
          <interface-mode>TRUNK</interface-mode>
        </config>
      </switched-vlan>
    </ethernet>
  </interface>
</interfaces>
</config>
</edit-config>
</rpc>

```

Example 2, payload configuring the VLAN ranges.

```

<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <interfaces xmlns="http://openconfig.net/yang/interfaces">
        <interface>
          <name>eth1/47</name>
          <subinterfaces>
            <subinterface>
              <index>0</index>
              <config>
                <index>0</index>
              </config>
            </subinterface>
          </subinterfaces>
          <ethernet xmlns="http://openconfig.net/yang/interfaces/ethernet">
            <switched-vlan xmlns="http://openconfig.net/yang/vlan">
              <config>
                <native-vlan>999</native-vlan>
                <trunk-vlans nc="urn:ietf:params:xml:ns:netconf:base:1.0"
nc:operation="delete">1..4094</trunk-vlans>
                <trunk-vlans>401</trunk-vlans>
                <trunk-vlans>999</trunk-vlans>
              </config>
            </switched-vlan>
          </ethernet>
        </interface>
      </interfaces>
    </config>
  </edit-config>
</rpc>

```

- Because of the design of OpenConfig YANG, when you configure VLANs, there must be no overlap between the VLANs in the payload and the VLANs already configured on an interface. If an overlap exists, the configuration through OpenConfig is not successful. Make sure that the VLANs configured on an interface are different from the VLANs in the OpenConfig payload. Pay particular attention to the starting and ending VLANs in a range.
- The following guidelines and limitations apply to OC-LACP:
 - Port-channel mode:

- OC-LACP enables configuring the port-channel mode on the port-channel interface. However, through the NXOS-CLI, the port-channel mode is configured on the member interface using channel-group mode active or passive.
- Although OC-LACP explicitly configures the port-channel mode on a port-channel interface, issuing the NX-OS **show running-config** command on a port-channel interface does not show the port-channel mode configuration for either empty or non-empty port-channels.
- Once a member is added to the port-channel, **show running interface ethernet <>** shows the port-channel mode configuration as a channel-group mode active or passive.

**Note**

All port-channels that are created through OpenConfig should continue to be managed by OpenConfig.

- Port-channel interval rate:
 - Port channel interval can only be changed when members are in `shut` state.
 - The OC-LACP interval is per port-channel. The NX-OS LACP interval is per port-channel member. Because of this difference, the following behavior can be expected:
 - If you configure the port-channel interval through OpenConfig, all members in the port-channel get the same configuration applied to them.
 - If you configure the port-channel interval through OpenConfig and later a member is added to the port-channel, you must configure the interval again through OpenConfig for the configuration to be applied to the new member.
- System MAC ID:
 - In this release, Cisco NX-OS does not support `system-id-mac` per port-channel.
- Member-state data for the following is present only when a port is in `admin up` state:
 - LACP
 - Interface
 - Interfaces
 - Member
 - State
- OSPFv2 can send an error response when you attempt to add an interface through OpenConfig YANG. When the problem occurs, the interface is not added, and the RPC reply contains a "list merge failed" error as follows:

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
message-id="urn:uuid:39507023-8569-4cf8-869c-e19aaf76a260">
  <rpc-error>
    <error-type>protocol</error-type>
    <error-tag>operation-failed</error-tag>
    <error-severity>error</error-severity>
```

```
<error-message xml:lang="en">List Merge Failed: operation-failed</error-message>
```

```
<error-path>/network-instances/network-instance/protocols/protocol/ospfv2/areas/area/interfaces/interface/idx</error-path>
```

```
</rpc-error>
```

```
</rpc-reply>
```

- Queueing stats for Hlg (ii) ports is not supported.
- You do not see the tx-packets, or bytes, and drop-packets per unicast, multicast, or broadcast queue. The stats that display in the OC response are a sum of the ucast, mcast, and bcast queues per qos-group.
- OpenConfig YANG does not support stats for a QoS policy that is applied at the VLAN level.
- The ingress queue drop count that can be retrieved through OC can be displayed at the slice/port/queue level depending on the platform.
- The following is the guideline and limitation for OpenConfig configurations for switchport, shut/no shut, MTU, and mac-address:
 - An ascii reload is required when configuring switchport, shut/no shut, MTU, and mac-address. Using a binary reload results in the configuration being lost.
- The following state containers are implemented for the OpenConfig ACL at interface-ref level:
 - /acl/interfaces/interface/interface-ref/state for acl/interfaces/state container.
 - acl/interfaces/interface/interface-ref/state/interface for read-only oc-if:interface leaf.
 - acl/interfaces/interface/interface-ref/state/subinterface for read-only oc-if:subinterface leaf.
- The following system config containers are implemented for domain-name, login-banner, and motd-banner models:
 - /system/config/domain-name for /top:System/top:dns-items/top:prof-items/top:Prof-list/top:dom-items/top:name container
 - system/config/login-banner for /top:System/top:userext-items/top:postloginbanner-items/top:message container
 - /system/config/motd-banner for /top:System/top:userext-items/top:preloginbanner-items/top:message container

Understanding Deletion of BGP Routing Instance

With OpenConfig YANG network-instance (OCNI), when attempting to delete only the BGP configuration of the default VRF instead of deleting the entire BGP routing instance, BGP information might not be deleted

at the protocols/BGP level. In this situation, when the delete is at the protocols or BGP level with the autonomous system number in the payload, only the configuration of the default VRF is deleted instead of removing the entire BGP routing instance.

Following is an example payload that would be used to delete the configuration under the default VRF in BGP.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <network-instances xmlns="http://openconfig.net/yang/network-instance">
        <network-instance>
          <name>default</name>
          <protocols>
            <protocol>
              <identifier>BGP</identifier>
              <name>bgp</name>
              <bgp xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" nc:operation="delete">

                <global>
                  <config>
                    <as>100</as>
                  </config>
                </global>
              </bgp>
            </protocol>
          </protocols>
        </network-instance>
      </network-instances>
    </config>
  </edit-config>
</rpc>
```

Expected Behavior: The BGP routing instance itself should be deleted, which is the equivalent to **no router bgp 100**.

Actual Behavior: Only the BGP configuration under the default VRF is deleted, and there is no equivalent single CLI configuration.

Following is the running configuration before the delete operation:

```
router bgp 100
  router-id 1.2.3.4
  address-family ipv4 unicast
  vrf abc
    address-family ipv4 unicast
    maximum-paths 2
```

And following is the running configuration after the delete operation:

```
router bgp 100
  vrf abc
    address-family ipv4 unicast
    maximum-paths 2
```

Verifying YANG

Use the following commands to verify YANG settings: :

Table 21: YANG Verification

Command	Description
<code>show telemetry yang direct-path cisco-nxos-device</code>	Displays the paths which are supported.

Enabling OpenConfig Support

To enable or disable OpenConfig support on the programmability agents (NETCONF, RESTCONF and gRPC), configure "[no] feature openconfig". For example:

```
switch(config)# feature netconf
switch(config)# feature restconf
switch(config)# feature grpc
switch(config)# feature openconfig
```

**Note**

In previous releases, mtz-openconfig-all RPM was downloaded separately and installed. This method is deprecated in 10.2(2) release.



第 20 章

XML 管理インターフェイス

ここでは、次の内容について説明します。

- [XML 管理インターフェイスについて \(237 ページ\)](#)
- [XML 管理インターフェイスのライセンス要件, on page 239](#)
- [XML 管理インターフェイスを使用するための前提条件, on page 239](#)
- [XML 管理インターフェイスを使用, on page 239](#)
- [サンプル XML インスタンスに関する情報 \(252 ページ\)](#)
- [その他の参考資料, on page 259](#)

XML 管理インターフェイスについて

XML 管理インターフェイスについて

XML 管理インターフェイスを使用してデバイスを構成できます。インターフェイスは XML ベースのネットワーク構成プロトコル (NETCONF) を使用します。これにより、デバイスを管理し、インターフェイスを介して XML 管理ツールまたはプログラムと通信できます。NETCONF の Cisco NX-OS 導入では、デバイスとの通信に Secure Shell (SSH) セッションを使用する必要があります。

NETCONF は、リモートプロシージャコール (RPC) メッセージ内にデバイス構成要素を含めることができる XML Schema (XSD) を使用して導入されます。RPC メッセージ内から、デバイスに実行させたいコマンドのタイプに一致する NETCONF 操作の 1 つを選択します。NETCONF を使用して、デバイスで CLI コマンドのセット全体を設定できます。NETCONF の使用については、[NETCONF XML インスタンスの作成, on page 242](#) と [RFC 4741](#) を参照してください。

SSH を介した NETCONF の使用の詳細については、[RFC 4742](#) を参照してください。

このセクションは、次のトピックで構成されています。

- [NETCONF レイヤ, on page 238](#)
- [SSH xmlagent, on page 238](#)

NETCONF レイヤ

NETCONF レイヤは次のとおりです：

Table 22: NETCONF レイヤ

レイヤ	例
トランスポート プロトコル	SSHv2
RPC	<rpc>、<rpc-reply>
運用者	<get-config>、<edit-config>
コンテンツ	show または 構成 コマンド

以下は、4 つの NETCONF レイヤの説明です。

- SSH トランスポート プロトコル — クライアントとサーバ間の安全な暗号化接続を提供します。
- RPC タグ：リクエストからの構成コマンドと、それに対応する XML サーバからの応答を導入します。
- NETCONF 操作タグ：構成コマンドのタイプを示します。
- 格納ファイル — 構成する機能の XML 表現を示します。

SSH xmlagent

デバイス ソフトウェアは、SSH バージョン 2 を介した NETCONF をサポートする xmlagent と呼ばれる SSH サービスを提供します。



Note xmlagent サービスは、Cisco NX-OS ソフトウェアでは XML サーバと呼ばれます。

NETCONF over SSH は、クライアントと XML サーバ間の hello メッセージの交換から始まります。最初の交換の後、クライアントは XML 要求を送信し、サーバは XML 応答で応答します。クライアントとサーバは、文字シーケンス > で要求と応答を終了します。この文字シーケンスは XML では有効ではないため、クライアントとサーバはメッセージがいつ終了するかを解釈でき、通信の同期が維持されます。

この [NETCONF XML インスタンスの作成](#), on page 242 セクションでは、使用できる XML 構成インスタンスを定義する XML スキーマについて説明します。

XML 管理インターフェイスのライセンス要件

製品	製品
Cisco NX-OS	XML 管理インターフェイスにはライセンスは必要ありません。ライセンスパッケージに含まれていない機能は Cisco NX-OS イメージにバンドルされており、無料で提供されます。NX-OS ライセンス方式の詳細については、『Cisco NX-OS Licensing Guide』を参照してください。

XML 管理インターフェイスを使用するための前提条件

XML 管理インターフェイスには、次の前提条件があります。

- クライアント PC に SSHv2 をインストールする必要があります。
- クライアント PC に NETCONF over SSH をサポートする XML 管理ツールをインストールする必要があります。
- デバイスの XML サーバに適切なオプションを設定する必要があります。

XML 管理インターフェイスを使用

このセクションでは、XML 管理インターフェイスを手動で構成して使用方法について説明します。デバイスのデフォルト設定で XML 管理インターフェイスを使用します。

SSH および XML サーバオプションの構成

デバイス上デフォルトで SSH サーバがイネーブル化されています。SSH を無効にする場合は、クライアント PC で SSH セッションを開始する前に有効にする必要があります。

XML サーバオプションを構成して、同時セッションの数とアクティブセッションのタイムアウトを制御できます。XML ドキュメントの検証を有効にして、XML セッションを終了することもできます。



Note XML サーバ タイムアウトはアクティブ セッションだけに適用できます。

SSH の構成の詳細については、ご使用のプラットフォームの Cisco NX-OS セキュリティ構成ガイドを参照してください。

XML コマンドの詳細については、ご使用のプラットフォームの Cisco NX-OS システム マネジメント 構成ガイドを参照してください。

SSH セッションを開始

次のようなコマンドを使用して、クライアント PC で SSHv2 セッションを開始できます。

```
ssh2 username@ip-address -s xmlagent
```

ログインユーザー名、デバイスの IP アドレス、接続するサービスを入力します。xmlagent サービスは、デバイス ソフトウェアでは XML サーバと呼ばれます。



Note SSH コマンドの構文は、クライアント PC の SSH ソフトウェアによって異なることがあります。

XML サーバから hello メッセージを受信しない場合は、次の条件を確認してください。

- デバイスで SSH サーバがイネーブルになっています。
- XML サーバの max-sessions オプションは、デバイスへの SSH 接続の数をサポートするのに十分です。
- デバイス上の現用系 XML サーバセッションの一部が使用されていません。

Hello メッセージを送信

XML サーバへの SSH セッションを開始すると、サーバはすぐに hello メッセージで応答し、サーバの機能をクライアントに通知します。サーバが他の要求を処理する前に、hello メッセージを使用してサーバに機能をアダプタイズする必要があります。XML サーバは基本機能のみをサポートし、クライアントからの基本機能のみのサポートを想定しています。

以下は、サーバとクライアントからのサンプルの hello メッセージです。



Note すべての XML ドキュメントは、]]>]]> で終了して、SSH 経由の NETCONF で同期がサポートされるようにする必要があります。

サーバからの hello メッセージ

```
<?xml version="1.0"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>urn:ietf:params:xml:ns:netconf:base:1.0</capability>
  </capabilities>
  <session-id>25241</session-id>
</hello>]]>]]>
```

クライアントからの hello メッセージ

```
<?xml version="1.0"?>
<nc:hello xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
```

```
<nc:capabilities>
  <nc:capability>urn:ietf:params:xml:ns:netconf:base:1.0</nc:capability>
</nc:capabilities>
</nc:hello>]]>]]>
```

XSD ファイルの取得

Procedure

-
- Step 1** ブラウザから、次の URL にあるシスコ ソフトウェア ダウンロード サイトに移動します。
<http://www.cisco.com/cisco/web/support/JP/loc/download/index.html>
ソフトウェア ダウンロード ページが開きます。
- Step 2** [製品の選択] リストで、**Switches > Data Center Switches > [プラットフォーム (platform)] > [モデル (model)]** を選択します。
- Step 3** 登録済みの Cisco ユーザーとしてまだログインしていない場合は、ここでログインするようにプロンプトします。
- Step 4** ソフトウェア タイプの選択リストから、**NX-OS XML Schema Definition.** を選択します。
- Step 5** 目的のリリースを見つけて **Download.** をクリックします
- Step 6** リクエストをされたら、強力な暗号化ソフトウェアイメージをダウンロードするための資格を適用するには、次の手順を実行します。
Cisco エンド ユーザー使用許諾契約書が開きます。
- Step 7** 次の手順に従って、**Agree** をクリックしてファイルを PC にダウンロードします。
-

XML ドキュメントを XML サーバに送信する

コマンドシェルで開いた SSH セッションを介して XML ドキュメントを XML サーバに送信するには、エディターから XML テキストをコピーして、SSH セッションに貼り付けます。通常、自動化されたメソッドを使用して XML ドキュメントを XML サーバに送信しますが、この方法で XML サーバへの SSH 接続を確認できます。

このメソッドの注意事項に従ってください：

- コマンドシェル出力で **hello** メッセージテキストを検索して、SSH セッションを開始した直後に XML サーバが **hello** メッセージを送信したことを確認します。
- XML 要求を送信する前に、クライアントの **hello** メッセージを送信します。XML サーバは **hello** 応答をすぐに送信するため、クライアント **hello** メッセージを送信した後、追加の応答は送信されません。
- XML ドキュメントは常に文字シーケンス `]]>]]>` で終了します。

NETCONF XML インスタンスの作成

RPC タグおよび NETCONF 操作タグ内に XML デバイス要素を囲むことにより、NETCONF XML インスタンスを作成できます。XML デバイス要素は、使用可能な CLI コマンドを XML フォーマットで囲む機能ベースの XML スキーマ定義（XSD）ファイルで定義されます。

以下は、フレームワーク コンテキストの NETCONF XML リクエストで使用されるタグです。タグラインは次のレター コードでマークキングされています：

- X — XML 宣言
- R — RPC リクエスト タグ
- N — NETCONF 操作タグ
- D — デバイス タグ

NETCONF XML フレームワークのコンテキスト

```
X <?xml version="1.0"?>
R <nc:rpc message-id="1" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" R
  xmlns="http://www.cisco.com/nxos:1.0:nfcli"> N <nc:get> N <nc:filter type="subtree">
D <show>
D <xml>
D <server>
D <status/>
D </server>
D </xml>
D </show>
N </nc:filter>
N </nc:get>
R </nc:rpc>]]>]]>
```



Note 任意の XML エディタまたは XML 管理インターフェイス ツールを使用して、XML インスタンスを作成する必要があります。

RPC リクエスト タグ `rpc`

すべての NETCONF XML インスタンスは、RPC リクエスト タグ `<rpc>` で開始する必要があります。RPC リクエスト タグ `<rpc>` の例は、`<rpc>` 要素を必須の **message-id** 属性と共に示しています。message-id 属性は、`<rpc-reply>` リクエストと返信を関連付けるために使用できます。`<rpc>` ノードもまた、次の XML 名前空間宣言も含まれています。

- NETCONF 名前空間宣言：「urn:ietf:params:xml:ns:netconf:base:1.0」名前空間で定義されている `<rpc>` と NETCONF タグは、`netconf.xsd` スキーマ ファイルに存在します。
- デバイスの名前空間宣言：`<rpc>` と NETCONF タグによってカプセル化されたデバイス タグは、他の名前空間で定義されています。デバイスの名前空間は機能指向です。Cisco NX-OS 機能タグは、さまざまな名前空間で定義されています。RPC リクエスト タグ `<rpc>` は、`nfcli` 機能を使用する例です。デバイスの名前空間が「xmlns=http://www.cisco.com/nxos:1.0:nfcli」であることを宣言しています。nfcli.xsd には、この名前空間の定義が含まれています。詳細については、「XSD ファイルの取得」に関するセクションを参照してください。

RPC タグ リクエスト

```
<nc:rpc message-id="315" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns=http://www.cisco.com/nxos:1.0:nfcli">
...
</nc:rpc>]]>]]>
```

構成リクエスト

以下は、構成リクエストの例です。

```
<?xml version="1.0"?>
<nc:rpc message-id="16" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns="http://www.cisco.com/nxos:1.0:if_manager">
  <nc:edit-config>
    <nc:target>
      <nc:running/>
    </nc:target>
    <nc:config>
      <configure>
        <__XML__MODE__exec_configure>
          <interface>
            <ethernet>
              <interface>2/30</interface>
              <__XML__MODE_if-ethernet>
                <__XML__MODE_if-eth-base>
                  <description>
                    <desc_line>Marketing Network</desc_line>
                  </description>
                </__XML__MODE_if-eth-base>
              </__XML__MODE_if-ethernet>
            </ethernet>
          </interface>
        </__XML__MODE__exec_configure>
      </configure>
    </nc:config>
  </nc:edit-config>
</nc:rpc>]]>]]>
```

__XML__MODE タグは、NETCONF エージェントによって内部的に使用されます。一部のタグは、特定の __XML__MODE の子としてのみ存在します。スキーマファイルを調べると、XML で CLI コマンドを表すタグにつながる正しいモード タグを見つけることができます。

NETCONF 動作タグ

NETCONF は、次の構成動作を提供します。

Table 23: Cisco NX-OS の NETCONF 動作

NETCONF 動作	説明	例
close-session	XML サーバー セッションを閉じます。	NETCONF クローズセッションインスタンス, on page 253
commit	候補構成の現在のコンテンツに、実行構成を設定します。	NETCONF コミットインスタンス - 候補構成機能, on page 257

NETCONF 動作	説明	例
confirmed-commit	指定された時間に構成をコミットするためのパラメータを提供します。確認タイムアウト期間内にこの操作の後にコミット操作が行われない場合、構成は確認済みコミット操作の前の状態に戻ります。	NETCONF Confirmed-commit インスタンス, on page 257
copy-config	送信元構成データストアのコンテンツをターゲットデータストアにコピーします。	NETCONF copy-config インスタンス, on page 253
delete-config	操作がサポートされていません。	—
edit-config	デバイスの実行構成の機能を構成します。この動作は、構成コマンドに使用します。	NETCONF edit-config インスタンス, on page 254 NETCONF rollback-on-error インスタンス, on page 258
get	デバイスから構成情報を受信します。この操作は show コマンドに使用します。データのソースは実行コンフィギュレーションです。	NETCONF XML インスタンスの作成, on page 242
get-config	構成の全体または一部を取得する	NETCONF get-config インスタンス, on page 255
kill-session	指定した XML サーバー セッションを閉じます。自分のセッションを閉じることはできません。close-session NETCONF 動作を参照してください。	NETCONF キルセッションインスタンス, on page 253
ロック	クライアントがデバイスの構成システムをロックすることができます	NETCONF ロック インスタンス, on page 256
unlock	セッションが発行した構成ロックを解放します。	NETCONF ロック解除インスタンス, on page 256

NETCONF 動作	説明	例
検証	構成をデバイスに適用する前に、構成候補のシンタックスエラーおよびセマンティクスエラーをチェックします。	NETCONF 検証機能インスタンス, on page 258

デバイスタグ

XML デバイス要素は、使用可能な CLI コマンドを XML フォーマットで表します。機能固有のスキーマファイルには、その特定の機能の CLI コマンドの XML タグが含まれています。「[XSD ファイルの取得, on page 241](#)」の項を参照してください。

このスキーマを使用して、XML インスタンスを構築することができます。次の例では、ビルド [NETCONF XML インスタンスの作成, on page 242](#) に使用された `nfcli.xsd` スキーマファイルの関連部分が示されています。

次の例は、XML デバイス タグを示しています。

xml デバイス タグを表示します。

```
<xs:element name="show" type="show_type_Cmd_show_xml"/>
<xs:complexType name="show_type_Cmd_show_xml">
  <xs:annotation>
    <xs:documentation>to display xml agent information</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:choice maxOccurs="1">
      <xs:element name="xml" minOccurs="1" type="xml_type_Cmd_show_xml"/>
      <xs:element name="debug" minOccurs="1" type="debug_type_Cmd_show_debug"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="xpath-filter" type="xs:string"/>
  <xs:attribute name="uses-namespace" type="nxos:bool_true"/>
</xs:complexType>
```

次の例は、サーバステータス デバイス タグを示しています。

サーバステータス デバイス タグ

```
<xs:complexType name="xml_type_Cmd_show_xml">
  <xs:annotation>
    <xs:documentation>xml agent</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="server" minOccurs="1" type="server_type_Cmd_show_xml"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="server_type_Cmd_show_xml">
  <xs:annotation>
    <xs:documentation>xml agent server</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:choice maxOccurs="1">
      <xs:element name="status" minOccurs="1" type="status_type_Cmd_show_xml"/>
      <xs:element name="logging" minOccurs="1" type="logging_type_Cmd_show_logging_facility"/>
    </xs:choice>
  </xs:sequence>
</xs:complexType>
```

```
</xs:sequence>
</xs:complexType>
```

次の例は、デバスタグの応答を示しています。

デバスタグの応答

```
<xs:complexType name="status_type_Cmd_show_xml">
  <xs:annotation>
    <xs:documentation>display xml agent information</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="__XML__OPT_Cmd_show_xml__readonly__" minOccurs="0">
      <xs:complexType>
        <xs:sequence>
          <xs:group ref="og_Cmd_show_xml__readonly__" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
<xs:group name="og_Cmd_show_xml__readonly__">
  <xs:sequence>
    <xs:element name="__readonly__" minOccurs="1" type="__readonly__type_Cmd_show_xml"/>
  </xs:sequence>
</xs:group>
<xs:complexType name="__readonly__type_Cmd_show_xml">
  <xs:sequence>
    <xs:group ref="bg_Cmd_show_xml_operational_status" maxOccurs="1"/>
    <xs:group ref="bg_Cmd_show_xml_maximum_sessions_configured" maxOccurs="1"/>
    <xs:group ref="og_Cmd_show_xml_TABLE_sessions" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
</xs:complexType>
```



Note 「__XML__OPT_Cmd_show_xml__readonly__」はオプションです。このタグは応答を表します。応答の詳細については、[RPC 応答タグ, on page 251](#) のセクションを参照してください。

<get> を実行するために使用できるタグを見つけるための | XML オプションを使用できます。以下は | XML オプションの例です。

XML の例

```
Switch#> show xml server status | xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<nf:rpc-reply xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns="http://www.cisco.com/nxos:1.0:nfcli">
  <nf:data>
    <show>
      <xml>
        <server>
          <status>
            <__XML__OPT_Cmd_show_xml__readonly__>
              <__readonly__>
                <operational_status>
                  <o_status>enabled</o_status>
                </operational_status>
                <maximum_sessions_configured>
                  <max_session>8</max_session>
                </maximum_sessions_configured>
```

```

</__readonly__>
</__XML_OPT_Cmd_show_xml__readonly__>
</status>
</server>
</xml>
</show>
</nf:data>
</nf:rpc-reply>
]]>]]>

```

この応答から、このコンポーネントで操作を実行するタグを定義する名前空間は `http://www.cisco.com/nxos:1.0:nfcli` であり、`nfcli.xsd` ファイルを使用してこの機能の要求を作成できることがわかります。

NETCONF 操作タグとデバイスタグを RPC タグで囲むことができます。<rpc>end-tag の後に XML 終了文字シーケンスが続きます。

拡張された NETCONF の操作

Cisco NX-OS は、<exec-command> という名前の<rpc> 操作をサポートします。この操作により、クライアントアプリケーションは CLI 構成と表示コマンドを送信し、それらのコマンドへの応答を XML タグとして受信できます。

以下は、インターフェイスの構成に使用されるタグの例です。タグ回線は、次の文字コードでマークされます。

- X — XML 宣言
- R — RPC リクエスト タグ
- EO — 拡張操作

<exec-command> を通して送信される構成 CLI コマンド

```

X <?xml version="1.0"?>
R <nf:rpc xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0"
  xmlns:nxos="http://www.cisco.com/nxos:1.0" message-id="3">
EO <nxos:exec-command>
EO <nxos:cmd>conf t ; interface ethernet 2/1 </nxos:cmd>
EO <nxos:cmd>channel-group 2000 ; no shut; </nxos:cmd>
EO </nxos:exec-command>
R </nf:rpc>]]>]]>

```

操作に対する応答は次のとおりです。

<exec-command> を通して送信された CLI コマンドへの応答

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<nf:rpc-reply xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0"
  xmlns:nxos="http://www.cisco.com/nxos:1.0" message-id="3">
<nf:ok/>
</nf:rpc-reply>
]]>]]>

```

次の例は、<exec-command> データの取得に使用できます。

<exec-command> を通して送信される表示 CLI コマンド

```
<?xml version="1.0"?>
<nf:rpc xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nxos="http://www.cisco.com/nxos:1.0" message-id="110">
<nxos:exec-command>
<nxos:cmd>show interface brief</nxos:cmd>
</nxos:exec-command>
</nf:rpc>]]>]]>
```

以下は操作に対する反応です。

<exec-command> を通して送信された **show CLI コマンド** への応答

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<nf:rpc-reply xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nxos="http://www.cisco.com/nxos:1.0"
xmlns:mod="http://www.cisco.com/nxos:1.0:if_manager" message-id="110">
<nf:data>
<mod:show>
<mod:interface>
<mod:__XML__OPT_Cmd_show_interface_brief__readonly__>
<mod:__readonly__>
<mod:TABLE_interface>
<mod:ROW_interface>
<mod:interface>mgmt0</mod:interface>
<mod:state>up</mod:state>
<mod:ip_addr>172.23.152.20</mod:ip_addr>
<mod:speed>1000</mod:speed>
<mod:mtu>1500</mod:mtu>
</mod:ROW_interface>
<mod:ROW_interface>
<mod:interface>Ethernet2/1</mod:interface>
<mod:vlan>--</mod:vlan>
<mod:type>eth</mod:type>
<mod:portmode>routed</mod:portmode>
<mod:state>down</mod:state>
<mod:state_rsn_desc>Administratively down</mod:state_rsn_desc>
<mod:speed>auto</mod:speed>
<mod:ratemode>D</mod:ratemode>
</mod:ROW_interface>
</mod:TABLE_interface>
</mod:__readonly__>
</mod:__XML__OPT_Cmd_show_interface_brief__readonly__>
</mod:interface>
</mod:show>
</nf:data>
</nf:rpc-reply>
]]>]]>
```

次の表に、操作タグの詳細な説明を示します。

Table 24: タグ

タグ	説明
<exec-command>	CLI コマンドの実行

タグ	説明
<cmd>	CLI コマンドが含まれています。コマンドは、show または構成コマンドです。複数の構成コマンドは、セミコロン「;」を使用して区切ります。複数の show コマンドはサポートされていません。複数の構成コマンドを異なる <cmd> タグを同じリクエストの一部として送信できます。詳細については、[<exec-command>を通して送信される構成 CLI コマンド（Configuration CLI Commands Sent Through <exec-command>）] の例を参照してください。

を介して送信される構成コマンドへの応答 <cmd> タグは次のとおりです。

- <nf:ok>: すべての構成コマンドが正常に実行されました。
- <nf:rpc-error>: 一部のコマンドが機能不全になりました。操作は最初のエラーで停止し、<nf:rpc-error> サブツリーは、機能不全になった構成に関する詳細情報を提供します。機能不全になったコマンドの前に実行された構成は、実行中の構成に適用されていることに注意してください。

次の例は、機能不全になった構成を示しています:

機能不全の構成

```
<?xml version="1.0"?>
<nf:rpc xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nxos="http://www.cisco.com/nxos:1.0" message-id="3">
  <nxos:exec-command>
    <nxos:cmd>configure terminal ; interface ethernet2/1 </nxos:cmd>
    <nxos:cmd>ip address 1.1.1.2/24 </nxos:cmd>
    <nxos:cmd>no channel-group 2000 ; no shut; </nxos:cmd>
  </nxos:exec-command>
</nf:rpc>]]>]]>
<?xml version="1.0" encoding="ISO-8859-1"?>
<nf:rpc-reply xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nxos="http://www.cisco.com/nxos:1.0" message-id="3">
  <nf:rpc-error>
    <nf:error-type>application</nf:error-type>
    <nf:error-tag>invalid-value</nf:error-tag>
    <nf:error-severity>error</nf:error-severity>
    <nf:error-message>Ethernet2/1: not part of port-channel 2000
  </nf:error-message>
    <nf:error-info>
      <nf:bad-element>cmd</nf:bad-element>
    </nf:error-info>
  </nf:rpc-error>
</nf:rpc-reply>
]]>]]>
```

コマンドの実行により、インターフェイスの IP アドレスは設定されますが、管理状態は変更されません（no shut コマンドは実行されません）。管理状態が変更されない理由は、no port-channel 2000 コマンドがエラーになるためです。

は <rpc-reply> を介して送信される show コマンドの結果 <cmd> show コマンドの XML 出力を含むタグ。

構成コマンドと表示コマンドを同じに組み合わせることはできません <exec-command> インスタンス。次の例は、同じインスタンスで組み合わせられた構成と **show** コマンドを示しています。

構成コマンドと show コマンドの組み合わせ

```
<?xml version="1.0"?>
<nf:rpc xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nxos="http://www.cisco.com/nxos:1.0" message-id="110">
<nxos:exec-command>
<nxos:cmd>conf t ; interface ethernet 2/1 ; ip address 1.1.1.4/24 ; show xml
server status </nxos:cmd>
</nxos:exec-command>
</nf:rpc>]]>]]>
<?xml version="1.0" encoding="ISO-8859-1"?>
<nf:rpc-reply xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nxos="http://www.cisco.com/nxos:1.0" message-id="110">
<nf:rpc-error>
<nf:error-type>application</nf:error-type>
<nf:error-tag>invalid-value</nf:error-tag>
<nf:error-severity>error</nf:error-severity>
<nf:error-message>Error: cannot mix config and show in exec-command. Config cmds
before the show were executed.
Cmd:show xml server status</nf:error-message>
<nf:error-info>
<nf:bad-element>cmd</nf:bad-element>
</nf:error-info>
</nf:rpc-error>
</nf:rpc-reply>
]]>]]>
```

show コマンドは、それ自体で送信する必要があります <exec-command> 次の例に示すようなインスタンス。

送信された CLI コマンドを表示 <exec-command>

```
<?xml version="1.0"?>
<nf:rpc xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nxos="http://www.cisco.com/nxos:1.0" message-id="110">
<nxos:exec-command>
<nxos:cmd>show xml server status ; show xml server status </nxos:cmd>
</nxos:exec-command>
</nf:rpc>]]>]]>
<?xml version="1.0" encoding="ISO-8859-1"?>
<nf:rpc-reply xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nxos="http://www.cisco.com/nxos:1.0" message-id="110">
<nf:rpc-error>
<nf:error-type>application</nf:error-type>
<nf:error-tag>invalid-value</nf:error-tag>
<nf:error-severity>error</nf:error-severity>
<nf:error-message>Error: show cmds in exec-command shouldn't be followed by anything
</nf:error-message>
<nf:error-info>
<nf:bad-element><cmd></nf:bad-element>
</nf:error-info>
</nf:rpc-error>
```

```
</nf:rpc-reply>
]]>]]>
```

NETCONF 応答

クライアントによって送信されるすべての XML 要求に対して、XML サーバーは RPC 応答タグ `<rpc-reply>` で囲まれた XML 応答を送信します。

ここでは、次の内容について説明します。

- [RPC 応答タグ, on page 251](#)
- [データタグにカプセル化されたタグの解釈, on page 251](#)

RPC 応答タグ

次の例は、RPC 応答タグ `<rpc-reply>` を表示しています。

RPC 応答エレメント

```
<nc:rpc-reply message-id="315" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns=http://www.cisco.com/nxos:1.0:nfcli">
  <ok/>
</nc:rpc-reply>]]>]]>
```

`<ok>`、`<data>`、そして `<rpc-error>` の要素は RPC 応答に表示される可能性があります。次の表は、`<rpc-reply>` タグ。

Table 25: RPC 応答エレメント

要素	説明
<code><ok></code>	RPC 要求は正常に完了しました。この要素は、応答でデータが返されない場合に使用されます。
<code><data></code>	RPC 要求は正常に完了しました。RPC リクエストに関連するデータは、 <code><data></code> 要素。
<code><rpc-error></code>	RPC 要求が失敗しました。エラー情報は、 <code><rpc-error></code> 要素。

データタグにカプセル化されたタグの解釈

によってカプセル化されたデバイスタグ `<data>` タグには、リクエストとそれに続くレスポンスが含まれます。クライアントアプリケーションは、`<readonly>` タグ。次に、例を示します。

RPC 応答データ

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<nf:rpc-reply xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns="http://www.cisco.com/nxos:1.0:if_manager">
  <nf:data>
```

```

<show>
<interface>
<__XML__OPT_Cmd_show_interface_brief__readonly__>
<__readonly__>
<TABLE_interface>
<ROW_interface>
<interface>mgmt0</interface>
<state>up</state>
<ip_addr>xx.xx.xx.xx</ip_addr>
<speed>1000</speed>
<mtu>1500</mtu>
</ROW_interface>
<ROW_interface>
<interface>Ethernet2/1</interface>
<vlan>--</vlan>
<type>eth</type>
<portmode>routed</portmode>
<state>down</state>
<state_rsn_desc>Administratively down</state_rsn_desc>
<speed>auto</speed>
<ratemode>D</ratemode>
</ROW_interface>
</TABLE_interface>
</__readonly__>
</__XML__OPT_Cmd_show_interface_brief__readonly__>
</interface>
</show>
</nf:data>
</nf:rpc-reply>
]]>]]>

```

<__XML__OPT.*> と <__XML__BLK.*> はレスポンスに表示され、リクエストで使用されることもあります。これらのタグは NETCONF エージェントによって使用され、<__readonly__> タグの後の応答に存在します。これらは要求が必要であり、CLI コマンドを表す XML タグに到達するためにスキーマ ファイルに従って追加する必要があります。

サンプル XML インスタンスに関する情報

XML インスタンスの例

このセクションでは、次の XML インスタンスの例を示します：

- [NETCONF クローズ セッション インスタンス, on page 253](#)
- [NETCONF キルセッション インスタンス, on page 253](#)
- [NETCONF copy-config インスタンス, on page 253](#)
- [NETCONF edit-config インスタンス, on page 254](#)
- [NETCONF get-config インスタンス, on page 255](#)
- [NETCONF ロック インスタンス, on page 256](#)
- [NETCONF ロック解除インスタンス, on page 256](#)
- [NETCONF コミット インスタンス - 候補構成機能, on page 257](#)
- [NETCONF Confirmed-commit インスタンス, on page 257](#)
- [NETCONF rollback-on-error インスタンス, on page 258](#)

- [NETCONF 検証機能インスタンス](#) , on page 258

NETCONF クローズ セッション インスタンス

次の例は、セッション終了要求とそれに続くセッション終了応答を表示しています。

クローズ セッション リクエスト

```
<?xml version="1.0"?>
<nc:rpc message-id="101" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns="http://www.cisco.com/nxos:1.0">
<nc:close-session/>
</nc:rpc>]]>]]>
```

クローズ セッションの応答

```
<nc:rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns="http://www.cisco.com/nxos:1.0" message-id="101">
<nc:ok/>
</nc:rpc-reply>]]>]]>
```

NETCONF キルセッション インスタンス

次の例は、キルセッション要求とそれに続く kill-session レスポンスを示しています。

キルセッション要求

```
<nc:rpc message-id="101" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns="http://www.cisco.com/nxos:1.0">
<nc:kill-session>
<nc:session-id>25241</nc:session-id>
</nc:kill-session>
</nc:rpc>]]>]]>
```

キルセッション要求

```
<nc:rpc message-id="101" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns="http://www.cisco.com/nxos:1.0">
<nc:kill-session>
<nc:session-id>25241</nc:session-id>
</nc:kill-session>
</nc:rpc>]]>]]>
```

NETCONF copy-config インスタンス

次の例は、copy-config 要求とそれに続く copy-config 応答を示しています。

copy-config リクエスト

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<copy-config>
<target>
<running/>
</target>
<source>
<url>https://user@example.com:passphrase/cfg/new.txt</url>
```

```

</source>
</copy-config>
</rpc>

```

copy-config の応答

```

xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<ok/>
</rpc-reply>

```

NETCONF edit-config インスタンス

次の例は、NETCONF edit-config の使用を示しています。

edit-config リクエスト

```

<?xml version="1.0"?>
<nc:rpc message-id="16" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns="http://www.cisco.com/nxos:1.0:if_manager">
<nc:edit-config>
<nc:target>
<nc:running/>
</nc:target>
<nc:config>
<configure>
<__XML__MODE__exec_configure>
<interface>
<ethernet>
<interface>2/30</interface>
<__XML__MODE_if-ethernet>
<__XML__MODE_if-eth-base>
<description>
<desc_line>Marketing Network</desc_line>
</description>
</__XML__MODE_if-eth-base>
</__XML__MODE_if-ethernet>
</ethernet>
</interface>
</__XML__MODE__exec_configure>
</configure>
</nc:config>
</nc:edit-config>
</nc:rpc>]]>]]>

```

edit-config 応答

```

<?xml version="1.0"?>
<nc:rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns="http://www.cisco.com/nxos:1.0:if_manager" message-id="16">
<nc:ok/>
</nc:rpc-reply>]]>]]>

```

edit-config の operation 属性は、指定された操作が実行される構成のポイントを識別します。操作属性が指定されていない場合、構成は既存の構成データストアにマージされます。操作属性には、次の値を指定できます。

- create
- merge

- delete

次の例は、実行中の構成からインターフェイス Ethernet0/0 の構成を削除する方法を示しています。

edit-config: 削除操作の要求

```
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<edit-config>
<target>
<running/>
</target>
<default-operation>none</default-operation>
<config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
<top xmlns="http://example.com/schema/1.2/config">
<interface xc:operation="delete">
<name>Ethernet0/0</name>
</interface>
</top>
</config>
</edit-config>
</rpc>]]>]]>
```

edit-config への応答: 削除操作

```
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<ok/>
</rpc-reply>]]>]]>
```

NETCONF get-config インスタンス

次の例は、NETCONF get-config の使用を示しています。

サブツリー全体を取得するための Get-config リクエスト

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<get-config>
<source>
<running/>
</source>
<filter type="subtree">
<top xmlns="http://example.com/schema/1.2/config">
<users/>
</top>
</filter>
</get-config>
</rpc>]]>]]>
```

クエリの結果を含む Get-config 応答

```
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<data>
<top xmlns="http://example.com/schema/1.2/config">
<users>
<user>
<name>root</name>
<type>superuser</type>
<full-name>Charlie Root</full-name>
<company-info>
```

```

<dept>1</dept>
<id>1</id>
</company-info>
</user>
<!-- additional <user> elements appear here... -->
</users>
</top>
</data>
</rpc-reply>]]>]]>

```

NETCONF ロック インスタンス

次の例は、NETCONF ロック操作の使用を示しています。

次の例は、ロック要求、成功の応答、および失敗した試行への応答を示しています。

ロック要求

```

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

```

ロック取得成功時の応答

```

<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/> <!-- lock succeeded -->
</rpc-reply>]]>]]>

```

ロックの取得に失敗した場合の応答

```

<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <rpc-error> <!-- lock failed -->
    <error-type>protocol</error-type>
    <error-tag>lock-denied</error-tag>
    <error-severity>error</error-severity>
    <error-message>
      Lock failed, lock is already held
    </error-message>
    <error-info>
      <session-id>454</session-id>
      <!-- lock is held by NETCONF session 454 -->
    </error-info>
  </rpc-error>
</rpc-reply>]]>]]>

```

NETCONF ロック解除 インスタンス

次の例は、NETCONF ロック解除操作の使用を示しています。

ロック解除要求

```

<rpc message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <unlock>

```

```
<target>
<running/>
</target>
</unlock>
</rpc>
```

ロック解除要求への応答

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

NETCONF コミット インスタンス - 候補構成機能

次の例は、操作をコミットと返信をコミットを示しています。

操作をコミット

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

返信をコミット

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

NETCONF Confirmed-commit インスタンス

次の例は、confirmed-commit 操作と confirmed-commit 応答を表示しています。

確認されたコミット リクエスト

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<commit>
<confirmed/>
<confirm-timeout>120</confirm-timeout>
</commit>
</rpc>]]>]]>
```

確認されたコミット 応答

```
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<ok/>
</rpc-reply>]]>]]>
```

NETCONF rollback-on-error インスタンス

次の例は、エラー機能での NETCONF ロールバックの使用を示しています。文字列 `urn:ietf:params:netconf:capability:rollback-on-error:1.0` は、機能を識別します。

次の例は、エラー時のロールバックとこの要求への応答を構成する方法を示しています。

Rollback-on-error 機能

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <error-option>rollback-on-error</error-option>
  </edit-config>
</rpc>
```

Rollback-on-error リスponse

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

NETCONF 検証機能インスタンス

次の例は、NETCONF 検証機能の使用を示しています。文字列 `urn:ietf:params:netconf:capability:validate:1.0` は機能を識別します。

リクエストの検証

```
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <validate>
    <source>
      <candidate/>
    </source>
  </validate>
</rpc>
```

リクエストの検証への応答

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

その他の参考資料

ここでは、XML 管理インターフェイスの実装に関する追加情報について説明します。

標準

標準	タイトル
この機能がサポートする新しい規格または変更された規格はありません。既存の規格のサポートは、この機能によって変更されていません。	—

RFC

RFC	タイトル
RFC 4741	NETCONF Configuration Protocol
RFC 4742	セキュアシェル（SSH）経由の NETCONF 構成プロトコルの使用



PART I

Model-Driven Programmability

- [Managing Components, on page 263](#)
- [Converting CLI Commands to Network Configuration Format, on page 269](#)
- [gNMI - gRPC Network Management Interface, on page 275](#)
- [gNOI-gRPC ネットワーク操作インターフェイス \(319 ページ\)](#)
- [Model Driven Telemetry, on page 327](#)



CHAPTER 21

Managing Components

- [コンポーネント RPM パッケージについて \(263 ページ\)](#)
- [インストールの準備 \(266 ページ\)](#)
- [Cisco Artifactory からのコンポーネントのダウンロード \(267 ページ\)](#)
- [Installing RPM Packages, on page 267](#)

コンポーネント RPM パッケージについて



(注) Cisco NX-OS リリース 7.0(3)I6(2) 以降では、Cisco NX-OS イメージに、NX-OS プログラマブルインターフェイス コンポーネント RPM パッケージ（エージェント、Cisco ネイティブモデル、その他の必要なほとんどのモデル、およびインフラストラクチャ）が含まれています。そのため、イメージがロードされると、必要なソフトウェアのほぼすべてが自動的にインストールされます。この状況は、Cisco Artifactory から大量のソフトウェアをダウンロードしてインストールする必要がないことを意味します。例外は、必須の OpenConfig モデルです。Cisco Artifactory から OpenConfig モデルを明示的にダウンロードする必要があります。

一方、Cisco NX-OS リリース 7.0(3)I6(1) 以前のリリースでは、アップグレードが必要な場合、パッケージのダウンロードとインストールについて説明する次のセクションに従う必要があります。

NX-OS プログラマブルインターフェイス コンポーネント RPM パッケージは、Cisco Artifactory からダウンロードできます。必要なコンポーネント RPM パッケージには、次の 2 つのタイプがあります。

- 基本コンポーネント（必須）
- 共通モデルコンポーネント（OpenConfig モデルは明示的にダウンロードしてインストールする必要があります）

基本コンポーネント

基本コンポーネントは、次の必要な RPM パッケージで構成されます。

- **mtx-infra:** インフラストラクチャ

- **mtx-device:** Cisco ネイティブ モデル

モデル化された NX-OS インターフェイスにアクセスするには、次のエージェント パッケージの少なくとも 1 つをインストールする必要があります。

- **mtx-netconf-agent:** NETCONF エージェント
- **mtx-restconf-agent:** RESTCONF エージェント
- **mtx-grpc-agent:** gRPC エージェント

共通モデル コンポーネント

共通モデル コンポーネント RPM は、OpenConfig モデルをサポートします。OpenConfig モデルを使用するには、OpenConfig RPM をダウンロードしてインストールする必要があります。利便性のため、サポートされているすべての OpenConfig モデルの単一の結合パッケージ、mtx-openconfig-all があります。

単一の結合パッケージを推奨していますが、代わりに、選択したモデルと、次の表に記載されているサポート モデル間の依存関係を満たすための RPM をダウンロードしてインストールすることもできます。mtx-openconfig-all RPM は、個々のモデル RPM と互換性がありません。後者をインストールする場合はその前に前者をアンインストールし、また逆に、前者をインストールする場合はその前に後者をアンインストールする必要があります。

モデル名	モデル リビジョン	モデル Ver	パッケージ名	依存関係
openconfig-acl	2017-05-26	1.0.0	mtx-openconfig-acl	mtx-openconfig-interfaces
openconfig-bgp-policy	2017-07-30	4.0.1	mtx-openconfig-bgp-policy	mtx-openconfig-interfaces mtx-openconfig-routing-policy
openconfig-if-aggregate	2017-07-14	2.0.0	mtx-openconfig-if-aggregate	mtx-openconfig-if-ethernet mtx-openconfig-interfaces
openconfig-if-ethernet	2017-07-14	2.0.0	mtx-openconfig-if-ethernet	mtx-openconfig-interfaces
openconfig-if-ip	2016-05-26	1.0.2	mtx-openconfig-if-ip	mtx-openconfig-if-aggregate mtx-openconfig-if-ethernet mtx-openconfig-interfaces mtx-openconfig-vlan
openconfig-if-ip-ext	2018-01-05	2.3.0	mtx-openconfig-if-ip-ext	mtx-openconfig-if-aggregate mtx-openconfig-if-ethernet mtx-openconfig-if-ip mtx-openconfig-interfaces mtx-openconfig-vlan

モデル名	モデル リビジョン	モデル Ver	パッケージ名	依存関係
openconfig-interfaces	2017-07-14	2.0.0	mtx-openconfig-interfaces	-
openconfig-network-instance	2017-08-24	0.8.1	mtx-openconfig-network-instance	mtx-openconfig-bgp-policy mtx-openconfig-if-aggregate mtx-openconfig-if-ethernet mtx-openconfig-interfaces mtx-openconfig-routing-policy mtx-openconfig-vlan
openconfig-network-instance-policy	2017-02-15	0.1.0	mtx-openconfig-network-instance-policy	mtx-openconfig-routing-policy
openconfig-ospf-policy	2017-08-24	0.1.1	mtx-openconfig-ospf-policy	mtx-openconfig-interfaces mtx-openconfig-routing-policy
openconfig-platform	2018-01-16	0.8.0	mtx-openconfig-platform	-
openconfig-platform-linecard	2017-08-03	0.1.0	mtx-openconfig-platform-linecard	mtx-openconfig-platform
openconfig-platform-port	2018-01-20	0.3.0	mtx-openconfig-platform-port	mtx-openconfig-if-ethernet mtx-openconfig-interfaces mtx-openconfig-platform
openconfig-platform-transceiver	2018-01-22	0.4.1	mtx-openconfig-platform-transceiver	mtx-openconfig-if-ethernet mtx-openconfig-interfaces mtx-openconfig-platform
openconfig-relay-agent	2016-05-16	0.1.0	mtx-openconfig-relay-agent	mtx-openconfig-interfaces
openconfig-routing-policy	2016-05-12	2.0(1)	mtx-openconfig-routing-policy	-
openconfig-spanning-tree	2017-07-14	0.2.0	mtx-openconfig-spanning-tree	mtx-openconfig-interfaces
openconfig-system	2017-09-18	0.3.0	mtx-openconfig-system	-
openconfig-vlan	2017-07-14	2.0.0	mtx-openconfig-vlan	mtx-openconfig-if-aggregate mtx-openconfig-if-ethernet mtx-openconfig-interfaces

インストールの準備

ここでは、インストールの準備と、NX-OS プログラマブル インターフェイス コンポーネントを管理するためのその他の有用な情報について説明します。

デバイスで Bash シェルを開く

スイッチへの RPM のインストールは、Bash シェルで実行されます。デバイスで機能 **feature bash** が構成されていることを確認します。

```
Switch# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Switch(config)# feature bash-shell
Switch(config)# end
Switch# run bash sudo su
bash-4.2#
```

Bash からデバイスの CLI プロンプトに戻るには、**exit** または **Ctrl-D** を入力します。

デバイスの準備状況の確認

RPM をインストールする前に、次の CLI **show** コマンドを使用してデバイスの準備状況を確認できます。

- **show module**: すべてのモジュールが稼働しているかどうかを示します。

```
Switch# show module
```

- **show system redundancy status**: スタンバイ デバイスが稼働しており、HA モードになっているかどうかを示します。スタンバイ同期が進行中の場合、RPM のインストールが失敗する可能性があります。

```
Switch# show system redundancy status
```

ラインカードが起動しない場合は、Bash シェルで **createrepo /rpms** コマンドを入力します。

```
bash-4.2# createrepo /rpms
```

デバイスへのファイルのコピー

SCP を使用すると、次の形式のコマンドを使用してファイルをデバイスにコピーできます。

copy scp://username@source_ip/path_to_agent_rpm bootflash: vrf management

例:

```
Switch# copy scp://jdoe@192.0.20.123//myrpms/mtx-infra.1.0.0.r082616.x86_64.rpm bootflash:
vrf management
```

インストールされている **NX-OS** プログラマブル インターフェイス **RPM** の表示

インストールされているすべての NXOS プログラマブル インターフェイス RPM を表示するには、デバイスで次のコマンドを実行します。

```
bash-4.2# yum list installed | grep mt*
```

Cisco Artifactory からのコンポーネントのダウンロード

NX-OS プログラマブル インターフェイス コンポーネント RPM は、次の URL の Cisco Artifactory からダウンロードできます。RPM は、NX-OS リリースごとのディレクトリに整理されています。正しい NX-OS リリース ディレクトリから RPM をダウンロードしていることを確認してください。

<https://devhub.cisco.com/artifactory/open-nxos-agents>

NX-OS プログラマブル インターフェイス コンポーネント RPM は、次の命名規則に従っています。

`<package>-<version>-<NX-OS release>.<architecture>.rpm`

次のセクションの説明に従って、目的のインストールする NX-OS プログラマブル インターフェイス コンポーネント RPM パッケージを選択して、デバイスにダウンロードします。

Installing RPM Packages

プログラマブル インターフェイス ベースおよび共通モデル コンポーネント RPM パッケージのインストール

始める前に

- Cisco Artifactory から、次のパッケージをダウンロードします。
 - mt*infra
 - mt*device
 - mt*-netconf-agent/mt*-restconf-agent/mt*-grpc-agent（少なくとも 1 つ）
 - mt*-openconfig-all（または、選択した個々のモデル）
- [デバイスの準備状況の確認（266 ページ）](#) の CLI コマンドを使用して、アクティブ デバイスとスタンバイ デバイスのすべてのラインカードが稼働しており、準備ができていることを確認します。

手順

Step 1 ダウンロードした RPM をデバイスにコピーします。

例:

```
Switch# copy scp://jdoe@192.0.20.123/myrpms/mtx-infra-2.0.0.0-9.2.1.lib32_n9000.rpm bootflash: vrf
management
Switch# copy scp://jdoe@192.0.20.123/myrpms/mtx-device-2.0.0.0-9.2.1.lib32_n9000.rpm bootflash: vrf
management
Switch# copy scp://jdoe@192.0.20.123/myrpms/mtx-netconf-agent-2.0.0.0-9.2.1.lib32_n9000.rpm bootflash:
vrf management
Switch# copy scp://jdoe@192.0.20.123/myrpms/mtx-openconfig-all-1.0.0.0-9.2.1.lib32_n9000.rpm bootflash:
vrf management
```

Step 2 Bash シェルから、RPM をインストールします。

例:

```
bash-4.2# cd /bootflash
bash-4.2# dnf install mtx-infra-2.0.0.0-9.2.1.lib32_n9000.rpm mtx-device-2.0.0.0-9.2.1.lib32_n9000.rpm
mtx-netconf-agent-2.0.0.0-9.2.1.lib32_n9000.rpm mtx-openconfig-all-1.0.0.0-9.2.1.lib32_n9000.rpm
```

Step 3 Bash シェルから、インストールを確認します。

例:

```
bash-4.2# dnf list installed | grep mtx
```



CHAPTER 22

Converting CLI Commands to Network Configuration Format

- [XMLIN に関する情報（269 ページ）](#)
- [XMLIN のライセンス要件（269 ページ）](#)
- [XMLIN ツールのインストールおよび使用（270 ページ）](#)
- [show コマンド出力の XML への変換（271 ページ）](#)
- [XMLIN の構成例（271 ページ）](#)

XMLIN に関する情報

XMLIN ツールは、CLI コマンドをネットワーク構成（NETCONF）プロトコル形式に変換します。NETCONF は、ネットワーク デバイスの構成をインストール、処理、削除する機能を提供するネットワーク管理プロトコルです。これは、構成データとプロトコル メッセージに XML ベースのエンコーディングを使用します。NETCONF プロトコルの NX-OS 実装は、`<get>`、`<edit-config>`、`<close-session>`、`<kill-session>`、および `<exec-command>` のプロトコル操作をサポートします。

XMLIN ツールは、show、EXEC、および構成コマンドに対応する NETCONF `<get>`、`<exec-command>`、および `<edit-config>` リクエストに変換します。複数の構成コマンドを単一の NETCONF `<edit-config>` インスタンスにまとめることができます。

XMLIN ツールはまた、show コマンドの出力を XML 形式に変換します。

XMLIN のライセンス要件

表 26: XMLIN ライセンス要件

製品	ライセンス要件
Cisco NX-OS	XMLIN にはライセンスは必要ありません。ライセンス パッケージに含まれていない機能はすべて Cisco NX-OS システム イメージにバンドルされており、追加費用は一切発生しません。NX-OS ライセンス方式の詳細については、『Cisco NX-OS Licensing Guide』を参照してください。

XMLIN ツールのインストールおよび使用

XMLIN ツールをインストールして、構成 コマンドを NETCONF フォーマットに変換するために使用できます。

始める前に

XMLIN ツールは、対応する機能セットまたは必要なハードウェア機能がデバイスで利用できない場合でも、コマンドの NETCONF インスタンスを生成できます。ただし、**xmlin** コマンドを入力する前に、いくつかの機能セットをインストールする必要がある場合があります。

手順の概要

- switch# **xmlin**
- switch(xmlin)# **configure terminal**
- コンフィギュレーション コマンド
- (任意) switch(config)(xmlin)# **end**
- (任意) switch(config-if-verify)(xmlin)# **show commands**
- (任意) switch(config-if-verify)(xmlin)# **exit**

手順の詳細

手順

	コマンドまたはアクション	目的
Step 1	switch# xmlin	
Step 2	switch(xmlin)# 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
Step 3	コンフィギュレーション コマンド	構成 コマンドを NETCONF フォーマットに変換します。
Step 4	(任意) switch(config)(xmlin)# end	対応する <edit-config> 要求を生成します。 (注) show コマンドに対して XML インスタンスを生成する前に、 end コマンドを入力して現在の XML 構成を終了する必要があります。
Step 5	(任意) switch(config-if-verify)(xmlin)# show commands	show コマンドを NETCONF フォーマットに変換します。
Step 6	(任意) switch(config-if-verify)(xmlin)# exit	EXEC モードに戻ります。

show コマンド出力の XML への変換

show コマンドの出力を XML に変換できます。

始める前に

変換するコマンドのすべての機能がインストールされ、デバイス上で有効になっていることを確認します。そうしない場合、コマンドは機能不全になります。

terminal verify-only コマンドを使用すると、デバイスに入力しなくても機能が有効になっていることを確認できます。

コマンドに対するすべての必須ハードウェアがデバイス上に存在することを確認します。そうしない場合、コマンドは機能不全になります。

XMLIN ツールがインストールされていることを確認します。

手順の概要

1. switch# *show-command* | **xmlin**

手順の詳細

手順

	コマンドまたはアクション	目的
Step 1	switch# <i>show-command</i> xmlin	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 (注) 構成コマンドと一緒にこのコマンドを使用することはできません。

XMLIN の構成例

次の例は、XMLIN ツールがデバイス上にどのようにインストールされ、一連の構成コマンドを <edit-config> インスタンスに変換するためにどのように使用されるかを示しています。

```
switch# xmlin
*****
Loading the xmlin tool. Please be patient.
*****
Cisco Nexus Operating System (NX-OS) Software
TAC support: http://www.cisco.com/tac
Copyright © 2002-2013, Cisco Systems, Inc. All rights reserved.
The copyrights to certain works contained in this software are
owned by other third parties and used and distributed under
```

```

license. Certain components of this software are licensed under
the GNU General Public License (GPL) version 2.0 or the GNU
Lesser General Public License (LGPL) Version 2.1. A copy of each
such license is available at
http://www.opensource.org/licenses/gpl-2.0.php and
http://www.opensource.org/licenses/lgpl-2.1.php

switch(xmlin)# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
switch(config)(xmlin)# interface ethernet 2/1
% Success
switch(config-if-verify)(xmlin)# cdp enable
% Success
switch(config-if-verify)(xmlin)# end
<?xml version="1.0"?>
<nf:rpc xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns="http://www.cisco.com/nxos:6.2.2.:configure_"
xmlns:m="http://www.cisco.com/nxos:6.2.2.:_exec"
xmlns:ml="http://www.cisco.com/nxos:6.2.2.:configure__if-eth-base" message-id="1">
  <nf:edit-config>
    <nf:target>
      <nf:running/>
    </nf:target>
    <nf:config>
      <m:configure>
        <m:terminal>
          <interface>
            <__XML_PARAM_interface>
              <__XML_value>Ethernet2/1</__XML_value>
              <m1:cdp>
                <m1:enable/>
              </m1:cdp>
            </__XML_PARAM_interface>
          </interface>
        </m:terminal>
      </m:configure>
    </nf:config>
  </nf:edit-config>
</nf:rpc>
]]>]]>

```

次の例は、**show** コマンドに対して XML インスタンスを生成する前に現在の XML 構成を終了するために **end** コマンドを入力する方法を表示しています。

```

switch(xmlin)# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
switch(config)(xmlin)# interface ethernet 2/1
switch(config-if-verify)(xmlin)# show interface ethernet 2/1
*****
Please type "end" to finish and output the current XML document before building a new one.
*****
% Command not successful

switch(config-if-verify)(xmlin)# end
<?xml version="1.0"?>
<nf:rpc xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns="http://www.cisco.com/nxos:6.2.2.:configure_"
xmlns:m="http://www.cisco.com/nxos:6.2.2.:_exec" message-id="1">
  <nf:edit-config>
    <nf:target>
      <nf:running/>
    </nf:target>
  </nf:edit-config>
</nf:rpc>

```

```

    <m:configure>
      <m:terminal>
        <interface>
          <__XML__PARAM__interface>
            <__XML__value>Ethernet2/1</__XML__value>
          </__XML__PARAM__interface>
        </interface>
      </m:terminal>
    </m:configure>
  </nf:config>
</nf:edit-config>
</nf:rpc>
]]>]]>

switch(xmlin) # show interface ethernet 2/1
<?xml version="1.0"?>
<nf:rpc xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns="http://www.cisco.com/nxos:6.2.2.:if_manager" message-id="1">
  <nf:get>
    <nf:filter type="subtree">
      <show>
        <interface>
          <__XML__PARAM__ifeth>
            <__XML__value>Ethernet2/1</__XML__value>
          </__XML__PARAM__ifeth>
        </interface>
      </show>
    </nf:filter>
  </nf:get>
</nf:rpc>
]]>]]>
switch(xmlin) # exit
switch#

```

次の例は、**show interface brief** コマンドの出力を XML に変換する方法を示しています。

```

switch# show interface brief | xmlin
<?xml version="1.0"?>
<nf:rpc xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns="http://www.cisco.com/nxos:6.2.2.:if_manager"

message-id="1">
  <nf:get>
    <nf:filter type="subtree">
      <show>
        <interface>
          <brief/>
        </interface>
      </show>
    </nf:filter>
  </nf:get>
</nf:rpc>
]]>]]>

```




CHAPTER 23

gNMI - gRPC Network Management Interface

This chapter contains the following topics:

- [About gNMI, on page 275](#)
- [gNMI RPC and SUBSCRIBE, on page 276](#)
- [gNMI に関する注意事項と制限事項 \(277 ページ\)](#)
- [Configuring gNMI, on page 280](#)
- [gNMI - gRPC Network Management Interface, on page 282](#)
- [サーバー証明書の構成 \(282 ページ\)](#)
- [Generating Key/Certificate Examples, on page 283](#)
- [キー/証明書の生成の例 \(284 ページ\)](#)
- [Cisco NX-OS リリース 9.3\(3\) 以降のキー/証明書の生成と構成の例 \(285 ページ\)](#)
- [Enabling gNMI, on page 287](#)
- [Verifying gNMI, on page 288](#)
- [gRPC クライアント証明書認証 \(293 ページ\)](#)
- [新しいクライアントルート CA 証明書の生成 \(293 ページ\)](#)
- [NX-OS デバイスでの生成されたルート CA 証明書の構成 \(294 ページ\)](#)
- [gRPC へのトラストポイントの関連付け \(294 ページ\)](#)
- [証明書の詳細の検証 \(295 ページ\)](#)
- [任意の gNMI クライアントのクライアント証明書認証を使用した接続の確認 \(296 ページ\)](#)
- [Clients, on page 297](#)
- [Sample DME Subscription - JSON Encoding, on page 297](#)
- [Sample DME Subscription - PROTO Encoding, on page 298](#)
- [Capabilities, on page 299](#)
- [Get, on page 303](#)
- [Set, on page 304](#)
- [Subscribe, on page 305](#)
- [Streaming Syslog, on page 309](#)
- [Troubleshooting, on page 315](#)

About gNMI

gNMI uses gRPC (Google Remote Procedure Call) as its transport protocol.

Cisco NX-OS supports gNMI for dial-in subscription to telemetry applications running on switches. Although the past release supported telemetry events over gRPC, the switch pushed the telemetry data to the telemetry receivers. This method was called dial out.

With gNMI, applications can pull information from the switch. They subscribe to specific telemetry services by learning the supported telemetry capabilities and subscribing to only the telemetry services that it needs.

Cisco NX-OS Release 9.3(1) and later supports gNMI version 0.5.0.

Cisco NX-OS Release 9.3(5) and later supports gNMI version 0.5.0.

Table 27: Supported gNMI RPCs

gNMI RPC	Supported
Capabilities	Yes
Get	Yes
Set	Yes
Subscribe	Yes

gNMI RPC and SUBSCRIBE

The NX-OS 9.3(1) release supports gNMI version 0.5.0. Cisco NX-OS Release 9.3(1) supports the following parts of gNMI version 0.5.0.

Table 28: SUBSCRIBE Options

Type	Sub Type	Supported?	Description
Once		Yes	Switch sends current values only once for all specified paths
Poll		Yes	Whenever the switch receives a Poll message, the switch sends the current values for all specified paths.
Stream	Sample	Yes	Once per stream sample interval, the switch sends the current values for all specified paths. The supported sample interval range is from 1 through 604800 seconds. The default sample interval is 10 seconds.

Type	Sub Type	Supported?	Description
	On_Change	Yes	The switch sends current values as its initial state, but then updates the values only when changes, such as create, modify, or delete occur to any of the specified paths.
	Target_Defined	No	

Optional SUBSCRIBE Flags

For the SUBSCRIBE option, some optional flags are available that modify the response to the options listed in the table. In release 9.3(1), the `updates_only` optional flag is supported, which is applicable to `ON_CHANGE` subscriptions. If this flag is set, the switch suppresses the initial snapshot data (current state) that is normally sent with the first response.

The following flags are not supported:

- aliases
- allow_aggregation
- extensions
- heart-beat interval
- prefix
- qos
- suppress_redundant

gNMI に関する注意事項と制限事項

gNMI に関する注意事項と制限事項は次のとおりです。

- Cisco NX-OS リリース 9.3(5) 以降、`Get` および `Set` がサポートされています。
- gNMI クエリは、パス内のワイルドカードをサポートしていません。
- Nexus デバイス向けの gRPC トラフィックは、デフォルト クラスのコントロールプレーンポリサー（CoPP）にヒットします。gRPC ドロップの可能性を抑えるには、管理クラス of gRPC 構成ポートを使用して、カスタム CoPP ポリシーを構成してください。
- 管理 VRF とデフォルト VRF の両方で gRPC をイネーブルにし、後でデフォルト VRF でディセーブルにすると、管理 VRF の gNMI 通知が機能しなくなります。

gNMI 通知を有効にするには、コマンドを入力して gRPC を有効にする必要があります。**feature grpc** これにより、管理 VRF で gNMI 通知が有効になります。管理 VRF に加えてデフォルト VRF で gNMI 通知が必要な場合は、コマンドを適用する必要があります。**grpc**

use-vrf default コマンドを入力してデフォルト VRF で無効になっている場合、管理 VRF の gNMI 通知は機能しません。 **no grpc use-vrf default**

回避策として、コマンドを入力して gRPC を完全に無効にし、コマンドと既存の gRPC コンフィギュレーション コマンドを入力して再プロビジョニングします。 **no feature grpcfeature grpc** たとえば、**grpc certificate** または **grpc port**。また、管理 VRF の既存の通知に再登録する必要があります。

- 次のような既存の CLI 設定を使用して OpenConfig ルーティングポリシーをサブスクライブしようとすると、OpenConfig モデルの現在の実装により空の値が返されます。

```
ip prefix-list bgp_v4_drop seq 5 deny 125.2.0.0/16 le 32
ipv6 prefix-list bgp_v6_drop seq 5 deny cafe:125:2::/48 le 128
```

using the xpath

```
openconfig-routing-policy:/routing-policy/defined-sets/prefix-sets/prefix-set[name=bgp_v4_drop]/config
openconfig-routing-policy:/routing-policy/defined-sets/prefix-sets/prefix-set[name=bgp_v6_drop]/config
```

- サーバー証明書認証のみが実行されます。クライアント証明書がサーバーによって認証されません。
- gRPC 証明書が明示的に設定されている場合、保存されたスタートアップ コンフィギュレーションを使用して以前の Cisco NX-OS 9.3(x) イメージにリロードした後、gRPC 機能は接続を受け入れません。この問題を確認するには、コマンドを入力します。ステータス行に次のようなエラーが表示されます。 **show grpc gnmi service statistics**
Status: Not running - Initializing...Port not available or certificate invalid.
サービスを復元するには、適切な証明書コマンドを設定解除して設定します。
- Cisco NX-OS リリース 9.3(3) 以降では、カスタム gRPC 証明書を設定している場合、コマンドを入力すると設定が失われます。 **reload ascii** デフォルトの day-1 証明書に戻ります。 **reload ascii** コマンドを入力した後は、スイッチをリロードします。スイッチが再び起動したら、gRPC カスタム証明書を再設定する必要があります。



(注) これは、コマンドを入力した場合に適用されます。 **grpc certificate**

- gRPC のデフォルト以外の VRF の到達可能性は、L3VNI/EVPN および IP 経由でのみサポートされます。ただし、デフォルト以外の VRF および VXLAN フラッドおよびラーニングでの MPLS を介した到達可能性はサポートされていません。
- origin、use_models、またはその両方の使用は、gNMI サブスクリプションではオプションです。
- gNMI サブスクリプションは、Cisco DME およびデバイス YANG データモデルをサポートします。Cisco NX-OS リリース 9.3(3)以降、サブスクライブは OpenConfig モデルをサポートします。

- 9.3(x) より前の Cisco NX-OS においてサポートされるプラットフォームの詳細については、そのリリース向けガイドのプログラマビリティ機能のプラットフォーム サポートを参照してください。Cisco NX-OS リリース 9.3(x) 以降、サポートされるプラットフォームの詳細については、[Nexus Switch Platform Matrix](#)を参照してください。

- この機能は、JSON および `gnmi.proto` エンコーディングをサポートします。この機能は、`protobuf.any` エンコーディングをサポートしていません。
- 各 gNMI メッセージの最大サイズは 12 MB です。収集されたデータの量が最大値 12 MB を超えると、収集されたデータはドロップされます。gNMI ON_CHANGE モードにのみ適用されます。

この状況は、より小規模で詳細なデータ収集セットを処理する、焦点を絞ったサブスクリプションを作成することで回避できます。したがって、1つの上位レベルのパスにサブスクライブする代わりに、パスの異なる下位レベルの部分に対して複数のサブスクリプションを作成してください。

- すべてのサブスクリプションで、最大 150K の集約 MO がサポートされます。より多くの MO に登録すると、収集データがドロップする可能性があります。
- この機能はサブスクリプション要求の単一パス プレフィックスをサポートしていませんが、サブスクリプションには空のプレフィックス フィールドを含めることができます。
- この機能は、Cisco DME およびデバイス YANG データモデルをサポートします。Openconfig YANG はサポートされていません。
- gNMI をサポートする gRPC プロセスは、CPU 使用率を CPU の 75% に、メモリを 1.5 GB に制限する HIGH_PRIO 制御グループを使用します。
- `show grpc gnmi` コマンドには、次の考慮事項があります。
 - このリリースでは、コマンドは XML 化されていません。
 - gRPC エージェントは、コールが終了した後、最大 1 時間 gNMI コールを保持します。
 - コールの合計数が 2000 を超えると、gRPC エージェントは、内部クリーンアップルーチンに基づいて終了したコールを消去します。
- Cisco NX-OS リリース 10.2(3)F 以降では、デバイス YANG エフェメラルデータ（アカウンティングログおよびマルチキャスト）のサブスクリプションの変更がサポートされています。

gRPC サーバーは管理 VRF で実行されます。その結果、gRPC プロセスはこの VRF でのみ通信し、管理インターフェイスはすべての gRPC 呼び出しをサポートする必要があります。

gRPC 機能には、各スイッチの合計 2 つの gRPC サーバーのデフォルト VRF が含まれるようになりました。VRF ごとに 1 つの gRPC サーバーを実行することも、管理 VRF で 1 つの gRPC サーバーのみを実行することもできます。デフォルト VRF で gRPC をサポートすると、大量のトラフィック負荷が望ましくない管理 VRF からの gRPC コールの処理を柔軟にオフロードできます。

2 つの gRPC サーバーを構成する場合は、次の点に注意してください。

- VRF 境界は厳密に適用されるため、各 gRPC サーバーは相互に独立して要求を処理します。要求は VRF 間を通過しません。
- 2 台のサーバーは HA またはフォールトトレラントではありません。一方の gRPC サーバーは他方をバックアップせず、それらの間でスイッチオーバーまたはスイッチバックはありません。
- gRPC サーバーの制限は VRF 単位です。

gNMI の制限事項は次のとおりです。

- パスでは複数レベルのワイルドカード「...」は使用できません
- パスの先頭にワイルドカード「*」を使用することはできません
- キー名でワイルドカード「*」を使用することはできません
- キーにワイルドカードと値を混在させることはできません

次の表に、gNMI のワイルドカードサポートの詳細を示します。

表 29: gNMI 要求のワイルドカードサポート

リクエストの種類	ワイルドカードサポート
gNMI GET	YES
gNMI 設定	NO
gNMI サブスクリプション（1 回）	YES
gNMI SUBSCRIBE、POLL	YES
gNMI SUBSCRIBE、STREAM、SAMPLE	YES
gNMI SUBSCRIBE、STREAM、TARGET_DEFINED	YES
gNMI SUBSCRIBE、STREAM、ON_CHANGE	NO

Configuring gNMI

Configure the gNMI feature through the **grpc gnmi** commands.

To import certificates used by the **grpc certificate** command onto the switch, see the [Installing Identity Certificates](#) section of the Cisco Nexus 3500 Series NX-OS Security Configuration Guide, Release 9.3(x).



Note When modifying the installed identity certificates or **grpc port** and **grpc certificate** values, the gRPC server might restart to apply the changes. When the gRPC server restarts, any active subscription is dropped and you must resubscribe.

SUMMARY STEPS

1. **configure terminal**
2. **feature grpc**
3. (Optional) **grpc port** *port-id*
4. (Optional) **grpc certificate** *certificate-id*
5. **grpc gnmi max-concurrent-call** *number*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	configure terminal Example: <pre>switch-1# configure terminal switch-1(config)#</pre>	Enters global configuration mode.
Step 2	feature grpc Example: <pre>switch-1# feature grpc switch-1(config)#</pre>	Enables the gRPC agent, which supports the gNMI interface for dial-in.
Step 3	(Optional) grpc port <i>port-id</i> Example: <pre>switch-1(config)# grpc port 50051</pre>	Configure the port number. The range of <i>port-id</i> is 1024–65535. 50051 is the default. Note This command is available beginning with Cisco NX-OS Release 9.3(3).
Step 4	(Optional) grpc certificate <i>certificate-id</i> Example: <pre>switch-1(config)# grpc certificate cert-1</pre>	Specify the certificate trustpoint ID. For more information, see the Installing Identity Certificates section of the Cisco Nexus 3000 Series NX-OS Security Configuration Guide, Release 9.3(x) for importing the certificate to the switch. Note This command is available beginning with Cisco NX-OS Release 9.3(3).

	Command or Action	Purpose
Step 5	grpc gnmi max-concurrent-call <i>number</i> Example: <pre>switch-1(config)# grpc gnmi max-concurrent-call 16 switch-1(config)#</pre>	Sets the limit of simultaneous dial-in calls to the gNMI server on the switch. Configure a limit from 1 through 16. The default limit is 8. The maximum value that you configure is for each VRF. If you set a limit of 16 and gNMI is configured for both management and default VRFs, each VRF supports 16 simultaneous gNMI calls. This command does not affect and ongoing or in-progress gNMI calls. Instead, gRPC enforces the limit on new calls, so any in-progress calls are unaffected and allowed to complete. Note The configured limit does not affect the gRPCConfigOper service.

gNMI - gRPC Network Management Interface

This section defines a gRPC-based protocol for the modification and retrieval of the configuration from a target device, as well as, the control and generation of telemetry streams from a target device to a data collection system. The intention is that a single gRPC service definition can cover both configuration and telemetry - allowing a single implementation on the target, as well as a single NMS element to interact with the device via telemetry and configuration RPCs.

サーバー証明書の構成

TLS 証明書を設定し、スイッチに正常にインポートした場合の **show grpc gnmi service statistics** コマンドの出力例を次に示します。

```
switch(config)# sh grpc gnmi service statistics

===== gRPC Endpoint
Vrf : management
Server address : [::]:50051

Cert notBefore : Nov 5 16:48:58 2015 GMT
Cert notAfter : Nov 5 16:48:58 2035 GMT
Client Root Cert notBefore : n/a
Client Root Cert notAfter : n/a

Max concurrent calls : 8
Listen calls : 1
Active calls : 0
KeepAlive Timeout : 120

Number of created calls : 1
Number of bad calls : 0

Subscription stream/once/poll : 0/0/0
```

```
Max gNMI::Get concurrent : 6
Max grpc message size : 25165824
gNMI Synchronous calls : 3
gNMI Synchronous errors : 3
gNMI Adapter errors : 3
gNMI Dtx errors : 0
```

gNMIはgRPCを介して通信し、TLSを使用してスイッチとクライアント間のチャンネルをセキュアにします。デフォルトのハードコードされたgRPC証明書は、スイッチに同梱されなくなりました。デフォルトの動作は、次に示すように、スイッチで生成される有効期限が1日の自己署名キーと証明書です。

証明書の有効期限が切れているか、正常にインストールできなかった場合は、1-D デフォルト証明書が表示されます。次に、**show grpc gnmi service statistics** コマンドの出力を示します。

```
#show grpc gnmi service statistics

=====
gRPC Endpoint
=====

Vrf          : management
Server address : [::]:50051

Cert notBefore : Wed Mar 11 19:43:01 PDT 2020
Cert notAfter  : Thu Mar 12 19:43:01 PDT 2020

Max concurrent calls      : 8
Listen calls              : 1
Active calls              : 0

Number of created calls   : 1
Number of bad calls       : 0

Subscription stream/once/poll : 0/0/0
```

有効期限は1日ですが、この一時証明書を使用してテストを簡単に行えます。長期的には、新しいキー/証明書を生成する必要があります。

Generating Key/Certificate Examples



Note Use this procedure for Cisco NX-OS Release 9.3(2) and earlier.

For Cisco NX-OS Release 9.3(3) and later, see

Typically, there are two possible scenarios:

1. The server and client can use a Self-Signed Certificate. Self-Signed Certificates are less secure and are not to be used in production environments.
 - a. Generating self-signed certificates on the switch:

1. Generating self-signed certificates on the switch:
 1. Login to the Bash shell:
 2. cd to the location where you want to store the key/cert

```

3. switch# openssl req -x509 -newkey rsa:2048 -keyout self_sign2048.key -out
self_sign2048.pem -days 365 -nodes
Generating a 2048 bit RSA private key
.....
.....
writing new private key to 'self_sign2048.key'
-----
You are about to be asked to enter information that will be
incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name
or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
string is too long, it needs to be less than 2 bytes long
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:CA
Locality Name (eg, city) []:San Jose
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Cisco
Systems
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:benton.cisco.com
Email Address []:
switch#
switch# ls -al | grep self
-rw-r--r-- 1 root root 912 Jun 6 15:16 self_sign2048.key
-rw-r--r-- 1 root root 952 Jun 6 15:16 self_sign2048.pem
switch#

2. Modify /etc/mtx.conf file. The mtx.conf file is not persistent after reloads but
persistent through "no feature grpc" & "feature grpc"
3. Under the "grpc" section in the file add the key and cert location
[grpc]
key = /bootflash/self_sign2048.key
cert = /bootflash/self_sign2048.pem
4. Go to cli and run
1. no feature grpc
2. feature grpc
3. show grpc internal gnmi service statistics
1. Verify that the new cert and key are used by grpc process.

5. Optional: If you want the changes to be persistent across reloads
1. Create a new file /etc/mtx.conf.user
2. Add the following
[grpc]
key = /bootflash/self_sign2048.key
cert = /bootflash/self_sign2048.pem
3. To get the mtx.conf file to accept the changes either
1. Install, activate or de-activate a MTX RPM
2. Or reload the box

```

2. You can generate a CA signed certificate.

キー/証明書の生成の例

キー/証明書を生成するには、次の例に従います。

- Cisco NX-OS リリース 9.3(3) 以降のキー/証明書の生成と構成の例 (285 ページ)

Cisco NX-OS リリース 9.3(3) 以降のキー/証明書の生成と構成の例

次に、キー/証明書を生成する例を示します。



(注) このタスクは、スイッチで証明書を生成する方法の例です。任意の Linux 環境で証明書を生成することもできます。実稼働環境では、CA 署名付き証明書の使用を検討する必要があります。

アイデンティティ証明書の生成の詳細については、『Cisco Nexus 9000 Series NX-OS Security Configuration Guide, Release 9.3(x)』の「Installing Identity Certificates」セクションを参照してください。https://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus9000/sw/93x/security/configuration/guide/b-cisco-nexus-9000-nx-os-security-configuration-guide-93x/b-cisco-nexus-9000-nx-os-security-configuration-guide-93x_chapter_011010.html#task_2088148

手順

Step 1 自己署名キーと pem ファイルを生成します。

```
switch# run bash sudo su
bash-4.3# openssl req -x509 -newkey rsa:2048 -keyout self_sign2048.key -out self_sign2048.pem -days 365 -nodes
```

a) switch# openssl req -x509 -newkey rsa:2048 -keyout self_sign2048.key -out self_sign2048.pem -days 365 -nodes

Step 2 キーファイルと pem ファイルを生成した後、トラストポイント CA アソシエーションで使用するためにキーファイルと pem ファイルをバンドルする必要があります。

```
switch# run bash sudo su
bash-4.3# cd /bootflash/
bash-4.3# openssl pkcs12 -export -out self_sign2048.pfx -inkey self_sign2048.key -in self_sign2048.pem -certfile self_sign2048.pem -password pass:Ciscolab123!
bash-4.3# exit
```

Step 3 pkcs12 バンドルをトラストポイントに入力して、トラストポイント CA アソシエーションを設定します。

```
switch(config)# crypto ca trustpoint mytrustpoint
switch(config-trustpoint)# crypto ca import mytrustpoint pkcs12 self_sign2048.pfx Ciscolab123!
```

Step 4 セットアップを確認します。

```
switch(config)# show crypto ca certificates
Trustpoint: mytrustpoint
certificate:
subject= /C=US/O=Cisco Systems, Inc./OU=CSG/L=San Jose/ST=CA/street=3700 Cisco
Way/postalCode=95134/CN=ems.cisco.com/serialNumber=FGE18420K0R
issuer= /C=US/O=Cisco Systems, Inc./OU=CSG/L=San Jose/ST=CA/street=3700 Cisco
```

```

Way/postalCode=95134/CN=ems.cisco.com/serialNumber=FGE18420K0R
serial=0413
notBefore=Nov  5 16:48:58 2015 GMT
notAfter=Nov  5 16:48:58 2035 GMT
SHA1 Fingerprint=2E:99:2C:CE:2F:C3:B4:EC:C7:E2:52:3A:19:A2:10:D0:54:CA:79:3E
purposes: sslserver sslclient

CA certificate 0:
subject= /C=US/O=Cisco Systems, Inc./OU=CSG/L=San Jose/ST=CA/street=3700 Cisco
Way/postalCode=95134/CN=ems.cisco.com/serialNumber=FGE18420K0R
issuer= /C=US/O=Cisco Systems, Inc./OU=CSG/L=San Jose/ST=CA/street=3700 Cisco
Way/postalCode=95134/CN=ems.cisco.com/serialNumber=FGE18420K0R
serial=0413
notBefore=Nov  5 16:48:58 2015 GMT
notAfter=Nov  5 16:48:58 2035 GMT
SHA1 Fingerprint=2E:99:2C:CE:2F:C3:B4:EC:C7:E2:52:3A:19:A2:10:D0:54:CA:79:3E
purposes: sslserver sslclient

```

Step 5 トラストポイントを使用するように gRPC を構成します。

```

switch(config)# grpc certificate mytrustpoint
switch(config)# show run grpc

!Command: show running-config grpc
!Running configuration last done at: Thu Jul  2 12:24:02 2020
!Time: Thu Jul  2 12:24:05 2020

version 9.3(5) Bios:version 05.38
feature grpc

grpc gnmi max-concurrent-calls 16
grpc use-vrf default
grpc certificate mytrustpoint

```

Step 6 gRPC が証明書を使用していることを確認します。

```

switch# show grpc gnmi service statistics

=====
gRPC Endpoint
=====

Vrf : management
Server address : [::]:50051

Cert notBefore : Nov 5 16:48:58 2015 GMT
Cert notAfter : Nov 5 16:48:58 2035 GMT

Max concurrent calls : 16
Listen calls : 1
Active calls : 0

Number of created calls : 953
Number of bad calls : 0

Subscription stream/once/poll : 476/238/238

Max gNMI::Get concurrent : 5
Max grpc message size : 8388608
gNMI Synchronous calls : 10
gNMI Synchronous errors : 0
gNMI Adapter errors : 0

```

```
gNMI Dtx errors : 0
```

Enabling gNMI

This procedure starts the gNMI agent which runs on the management VRF and port 50051.

You can configure it to run on the additional default VRF or configure the maximal concurrent connection number.

Before you begin

You might need to configure the key/cert. If you do not, the server generates a pair of temporary key/certs under `/opt/mtx/etc/grpc.pem` and `/opt/mtx/etc/grpc.key`. You need to download the key/cert to its clients. It is valid for one-day use.

SUMMARY STEPS

1. `configure terminal`
2. `feature grpc`
3. `grpc gnmi max-concurrent-calls number`
4. `grpc use-vrf default`

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	configure terminal Example: <code>switch# configure terminal</code>	Enters global configuration mode.
Step 2	feature grpc Example: <code>switch(config)# feature grpc</code>	Enables the gRPC agent, which supports the gNMI interface.
Step 3	grpc gnmi max-concurrent-calls number Example: <code>switch(config)# grpc gnmi max-concurrent-calls 16</code>	Sets the limit of simultaneous dial-in calls to the gNMI server on the switch. Configure a limit from 1 through 16. The default limit is 8. The maximum value that you configure is for each VRF. If you set a limit of 16 and gNMI is configured for both the management and default VRF, each VRF supports 16 simultaneous gNMI calls.

	Command or Action	Purpose
		This command does not affect ongoing or in-progress gNMI calls. Instead, gRPC enforces the limit on new calls. Any in-progress calls are unaffected and allowed to complete. Note The configured limit does not affect the gRPCConfigOper service.
Step 4	grpc use-vrf default Example: <pre>switch(config)# grpc use-vrf default</pre>	Enables the gRPC agent to accept incoming (dial-in) RPC requests from the default VRF. This step enables the default VRF to process incoming RPC requests. By default, the management VRF processes incoming RPC requests when the gRPC feature is enabled. Note Both VRFs process requests individually, so that requests do not cross between VRFs.

Verifying gNMI

To verify the gNMI configuration, enter the following command:

Command	Description
show grpc gnmi service statistics	Displays a summary of the agent running status, respectively for the management VRF, or the default VRF (if configured). It also displays: • Basic overall counters • Certificate expiration time Note If the certificate is expired, the agent cannot accept requests.
show grpc gnmi rpc summary	Displays the following: • Number of capability RPCs received. • Capability RPC errors. • Number of Get RPCs received. • Get RPC errors. • Number of Set RPCs received. • Set RPC errors. • More error types and counts.

Command	Description
show grpc gnmi transactions	The show grpc gnmi transactions command is the most dense and contains considerable information. It is a history buffer of the most recent 50 gNMI transactions that are received by the switch. As new RPCs come in, the oldest history entry is removed from the end. The following explains what is displayed: • RPC – This shows the type of RPC that was received (Get, Set, Capabilities) • DataType – For a Get only. Has values ALL, CONFIG, and STATE. • Session – shows the unique session-id that is assigned to this transaction. It can be used to correlate data that is found in other log files. • Time In -- shows timestamp of when the RPC was received by the gNMI handler. • Duration – time delta in ms from receiving the request to giving response. • Status – the status code of the operation returned to the client (0 = Success, !0 == error). This section is data that is kept per path within a single gNMI transaction. For example, a single Get or Set • subtype – for a Set RPC, shows the specific operation that is requested per path (Delete, Update, Replace). For Get, there is no subtype. • dtx – shows that this path was processed in DTX “fast” path or not. A dash ‘-’ means no, an asterisk ‘*’ means yes. • st – Status for this path. The meaning is as follows: • OK: path is valid and processed by infra successfully. • ERR: path is either invalid or generated error by infra • --: path not processed yet, might or might not be valid and has not been sent to infra yet. • path – the path

show grpc gnmi service statistics Example

```

=====
gRPC Endpoint
=====

Vrf : management
Server address : [::]:50051

Cert notBefore : Mar 13 19:05:24 2020 GMT
Cert notAfter  : Nov 20 19:05:24 2033 GMT

Max concurrent calls : 8
Listen calls : 1
Active calls : 0

Number of created calls : 1
Number of bad calls : 0

Subscription stream/once/poll : 0/0/0

Max gNMI::Get concurrent : 5
Max grpc message size : 8388608
gNMI Synchronous calls : 74
gNMI Synchronous errors : 0
gNMI Adapter errors : 0
gNMI Dtx errors : 0

```

show grpc gnmi rpc summary Example

```

=====
gRPC Endpoint
=====

Vrf : management
Server address : [::]:50051

Cert notBefore : Mar 31 20:55:02 2020 GMT
Cert notAfter  : Apr  1 20:55:02 2020 GMT

Capability rpcs : 1
Capability errors : 0
Get rpcs : 53
Get errors : 19
Set rpcs : 23
Set errors : 8
Resource Exhausted : 0
Option Unsupported : 6
Invalid Argument : 18
Operation Aborted : 1
Internal Error : 2
Unknown Error : 0

RPC Type      State      Last Activity  Cnt Req  Cnt Resp  Client
-----
Subscribe     Listen     04/01 07:39:21      0         0

```

show grpc gnmi transactions Example

```

=====
gRPC Endpoint

```

=====

Vrf : management
Server address : [::]:50051

Cert notBefore : Mar 31 20:55:02 2020 GMT
Cert notAfter : Apr 1 20:55:02 2020 GMT

RPC	DataType	Session	Time In	Duration(ms)	Status
Set	-	2361443608	04/01 07:43:49	173	0
subtype: dtx: st: path:					
Delete	-	OK	/System/intf-items/lb-items/LbRtdIf-list[id=lo789]		
Set	-	2293989720	04/01 07:43:45	183	0
subtype: dtx: st: path:					
Replace	-	OK	/System/intf-items/lb-items/LbRtdIf-list[id=lo6]		
Set	-	2297110560	04/01 07:43:41	184	0
subtype: dtx: st: path:					
Update	-	OK	/System/intf-items/lb-items/LbRtdIf-list[id=lo7]		
Set	-	0	04/01 07:43:39	0	10
Set	-	3445444384	04/01 07:43:33	3259	0
subtype: dtx: st: path:					
Delete	-	OK	/System/intf-items/lb-items/LbRtdIf-list[id=lo789]		
Delete	-	OK	/System/intf-items/lb-items/LbRtdIf-list[id=lo790]		
Delete	-	OK	/System/intf-items/lb-items/LbRtdIf-list[id=lo791]		
Delete	-	OK	/System/intf-items/lb-items/LbRtdIf-list[id=lo792]		
Delete	-	OK	/System/intf-items/lb-items/LbRtdIf-list[id=lo793]		
Delete	-	OK	/System/intf-items/lb-items/LbRtdIf-list[id=lo794]		
Delete	-	OK	/System/intf-items/lb-items/LbRtdIf-list[id=lo795]		
Delete	-	OK	/System/intf-items/lb-items/LbRtdIf-list[id=lo796]		
Delete	-	OK	/System/intf-items/lb-items/LbRtdIf-list[id=lo797]		
Delete	-	OK	/System/intf-items/lb-items/LbRtdIf-list[id=lo798]		
Delete	-	OK	/System/intf-items/lb-items/LbRtdIf-list[id=lo799]		
Delete	-	OK	/System/intf-items/lb-items/LbRtdIf-list[id=lo800]		
Delete	-	OK	/System/intf-items/lb-items/LbRtdIf-list[id=lo801]		
Delete	-	OK	/System/intf-items/lb-items/LbRtdIf-list[id=lo802]		
Delete	-	OK	/System/intf-items/lb-items/LbRtdIf-list[id=lo803]		
Delete	-	OK	/System/intf-items/lb-items/LbRtdIf-list[id=lo804]		
Delete	-	OK	/System/intf-items/lb-items/LbRtdIf-list[id=lo805]		
Delete	-	OK	/System/intf-items/lb-items/LbRtdIf-list[id=lo806]		
Delete	-	OK	/System/intf-items/lb-items/LbRtdIf-list[id=lo807]		
Delete	-	OK	/System/intf-items/lb-items/LbRtdIf-list[id=lo808]		
Set	-	2297474560	04/01 07:43:26	186	0
subtype: dtx: st: path:					
Update	-	OK	/System/ipv4-items/inst-items/dom-items/Dom-list[name=foo]/rt-items/Route-list[prefix=0.0.0.0/0]/nh-items/Nexthop-list[nhAddr=192.168.1.1/32][nhVrf=foo][nhIf=unspecified]/tag		
Set	-	2294408864	04/01 07:43:17	176	13
subtype: dtx: st: path:					
Delete	-	ERR	/System/intf-items/lb-items/LbRtdIf-list/descr		
Set	-	0	04/01 07:43:11	0	3
subtype: dtx: st: path:					
Update	-	--	/System/intf-items/lb-items/LbRtdIf-list[id=lo4]/descr		
Update	-	ERR	/system/processes		

```

Set          -          2464255200      04/01 07:43:05      708      0
subtype: dtx: st: path:
Delete -      OK /System/intf-items/lb-items/LbRtdIf-list[id=lo2]
Delete -      OK /System/intf-items/lb-items/LbRtdIf-list[id=lo777]
Delete -      OK /System/intf-items/lb-items/LbRtdIf-list[id=lo778]
Delete -      OK /System/intf-items/lb-items/LbRtdIf-list[id=lo779]
Delete -      OK /System/intf-items/lb-items/LbRtdIf-list[id=lo780]
Replace -     OK /System/intf-items/lb-items/LbRtdIf-list[id=lo3]/descr
Replace -     OK /System/intf-items/lb-items/LbRtdIf-list[id=lo4]/descr
Replace -     OK /System/intf-items/lb-items/LbRtdIf-list[id=lo5]/descr
Update -      OK /System/intf-items/lb-items/LbRtdIf-list[id=lo3]/descr
Update -      OK /System/intf-items/lb-items/LbRtdIf-list[id=lo4]/descr
Update -      OK /System/intf-items/lb-items/LbRtdIf-list[id=lo5]/descr

Set          -          3491213208      04/01 07:42:58      14      0
subtype: dtx: st: path:
Replace -     OK /System/intf-items/lb-items/LbRtdIf-list[id=lo3]/descr

Set          -          3551604840      04/01 07:42:54      35      0
subtype: dtx: st: path:
Delete -      OK /System/intf-items/lb-items/LbRtdIf-list[id=lo1]

Set          -          2362201592      04/01 07:42:52      13      13
subtype: dtx: st: path:
Delete -      ERR /System/intf-items/lb-items/LbRtdIf-list[id=lo3]/lbrtdif-items
/operSt

Set          -          0                04/01 07:42:47      0      3
subtype: dtx: st: path:
Delete -      ERR /System/*

Set          -          2464158360      04/01 07:42:46      172     3
subtype: dtx: st: path:
Delete -      ERR /system/processes/shabang

Set          -          2295440864      04/01 07:42:46      139     3
subtype: dtx: st: path:
Delete -      ERR /System/invalid/path

Set          -          3495739048      04/01 07:42:44      10      0

Get          ALL          3444580832      04/01 07:42:40      3      0
subtype: dtx: st: path:
-          -      OK /System/bgp-items/inst-items/disPolBatch

Get          ALL          0                04/01 07:42:36      0      3
subtype: dtx: st: path:
-          -      -- /system/processes/process[pid=1]

Get          ALL          3495870472      04/01 07:42:36      2      0
subtype: dtx: st: path:
-          *      OK /system/processes/process[pid=1]

Get          ALL          2304485008      04/01 07:42:36      33      0
subtype: dtx: st: path:
-          *      OK /system/processes

Get          ALL          2464159088      04/01 07:42:36      251     0
subtype: dtx: st: path:
-          -      OK /system

```



```

Get          ALL          2293232352      04/01 07:42:35      258          0
subtype: dtx: st: path:
-           -           OK /system

Get          ALL          0              04/01 07:42:33      0            12
subtype: dtx: st: path:
-           -           -- /intf-items

```

gRPC クライアント証明書認証

10.1(1) リリース以降、gRPC に追加の認証方式が提供されます。10.1(1) リリースより前の gRPC サービスは、サーバー証明書のみをサポートしていました。10.1(1) 以降では、クライアント証明書のサポートも追加するように認証が拡張され、gRPC でサーバー証明書とクライアント証明書の両方を検証できるようになっています。この機能拡張により、さまざまなクライアントにパスワードなしの認証が提供されます。

新しいクライアント ルート CA 証明書の生成

次に、クライアントルートに新しい証明書を生成する例を示します。

- 信頼できる認証局（CA）

DigiCert などの信頼できる CA を使用する場合は、次の手順を実行します。

手順の概要

1. CA 証明書ファイルをダウンロードします。
2. [Cisco NX-OS セキュリティ構成ガイド](#) の手順に従って、NX-OS にインポートします。

手順の詳細

手順

	コマンドまたはアクション	目的
Step 1	CA 証明書ファイルをダウンロードします。	
Step 2	Cisco NX-OS セキュリティ構成ガイド の手順に従って、NX-OS にインポートします。	• トラストポイント CA のアソシエーションの作成 の手順に従って、トラストポイント ラベルを作成します。 • CA の認証 の手順に従って、信頼できる CA 証明書を使用してトラストポイントを認証します。 （注） cat [CA_cert_file] からの CA 証明書を使用します。

NX-OS デバイスでの生成されたルート CA 証明書の構成

クライアント root に対する新しい証明書が正常に生成されたときの、スイッチで証明書を構成するためのコマンド例とその出力を次に示します。

```
switch(config)# crypto ca trustpoint my_client_trustpoint
enticate my_client_trustpoint
switch(config-trustpoint)# crypto ca authenticate my_client_trustpoint
input (cut & paste) CA certificate (chain) in PEM format;
end the input with a line containing only END OF INPUT :
-----BEGIN CERTIFICATE-----
MIIDUDCCAjigAwIBAgIJAJLisBKCGjQOMA0GCSqGSIb3DQEBCwUAMD0xCzAJBgNV
BAYTA1VTMqswCQYDVQQIDAJDQTERMA8GA1UEBwwIU2FuIEpvc2UxDjAMBGNVBAoM
BUNpc2NvMB4XDTIwMTAxNDIwNTYyNl0XDTQwMTAwOTIwNTYyNl0wPTELMakGA1UE
BhMCVVMxZzA1BjBGNVBAgMAkNBMRERDwYDVQQHDAhTYW4gSm9zZTEOMAwGA1UECgwF
Q2l2Y28wgGElMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCDEX7qZ2EdogZU4
EW0NSpB3EjY0nSlFLOw/iLKSXfIiQJD0Qhaw16fDnnYZj6vzWEa0ls8canqHCXQ1
gUyxFOdGDxa6neQFTqLowSA6UCSQA+eenN2PIpMOjfdFpaPiHu3mmcTIlxP39Ti3
/y548NNORSepApBNkZ1rJSB6Cu9AIFMZgrZXfQDKBGSUOf/CPnvIDZeLcun+zpUu
CxJLA76Et4buPMysuRqMGHIX8CYw8MtjmuCuCTHXNN3lghhgpFxfRw/69pykjU3R
YOrwlSukvYQhtefHuTHBmqym7MFoBEchwrlC5YTduDzmOvtkhsmpogRe3BiIBx45
AnZdttdilAgMBAAGjUzBRMB0GA1UdDgQWBBSH3IqRrm+mtB5GNsoLXFb3bAVg5TAF
BgNVHSMEGDAWgBSH3IqRrm+mtB5GNsoLXFb3bAVg5TAPBgNVHRMBAf8EBTADAQH/
MA0GCSqGSIb3DQEBCwUAA4IBAQA4Fpc6lRKzBGJQ/7oK1FNcTX/YXkneXdk7Zrj
8W0RS0Khxgke97d2Cw15P5reXO27kvXsnsz/VZn7JYGUvGS1xTlcCb6x6wNBR4Qr
t9qDBu+LykwqNOFe4VCAv6e4cMXNbH2wHBVS/NSoWnM2FGZ10VppjEGFm6OM+N6z
8n4/rWslfWfbn7T7xHH+N10Ffc+8q8h37opyCnb0ILj+a4rnyus8xXJPQb05DfJe
ahPNfdEsXKDOWNkrSDtmKwtWDqdtjSQc4xioKHoshnNgWBjboVPlMQ64UrajBycV
z9snWBm6p9SdTsV92YwFltRGUqpcI9olsBgH7FUVU1hmHDWE
-----END CERTIFICATE-----
END OF INPUT
Fingerprint(s): SHA1 Fingerprint=0A:61:F8:40:A0:1A:C7:AF:F2:F7:D9:C7:12:AE:29:15:52:9D:D2:AE
```

```
Do you accept this certificate? [yes/no]:yes
switch(config)#
```

NOTE: Use the CA Certificate from the .pem file content.

```
switch# show crypto ca certificates
Trustpoint: my_client_trustpoint
CA certificate 0:
subject=C = US, ST = CA, L = San Jose, O = Cisco
issuer=C = US, ST = CA, L = San Jose, O = Cisco
serial=B7E30B8F4168FB87
notBefore=Oct 1 17:29:47 2020 GMT
notAfter=Sep 26 17:29:47 2040 GMT
SHA1 Fingerprint=E4:91:4E:D4:41:D2:7D:C0:5A:E8:F7:2D:32:81:B3:37:94:68:89:10
purposes: sslserver sslclient
```

gRPC へのトラストポイントの関連付け

クライアント ルートに新しい証明書を正常に構成した場合に、スイッチ上でトラストポイントを gRPC に関連付ける出力例を次に示します。



(注) クライアント認証用のルート証明書を構成または削除すると、gRPCプロセスが再起動します。

```
# switch(config)# feature grpc

switch(config)# grpc client root certificate my_client_trustpoint
switch(config)# show run grpc

!Command: show running-config grpc
!Running configuration last done at: Wed Dec 16 20:18:35 2020
!Time: Wed Dec 16 20:18:40 2020

version 10.1(1) Bios:version N/A
feature grpc

grpc gnmi max-concurrent-calls 14
grpc use-vrf default
grpc certificate my_trustpoint
grpc client root certificate my_client_trustpoint
grpc port 50003
```

証明書の詳細の検証

スイッチの gRPC にトラストポイントを正常に関連付けられた場合の、証明書の詳細を検証するための出力例を次に示します。

```
switch# show grpc gnmi service statistics

=====
gRPC Endpoint
=====

Vrf : management
Server address : [::]:50003

Cert notBefore : Mar 13 19:05:24 2020 GMT
Cert notAfter : Nov 20 19:05:24 2033 GMT
Client Root Cert notBefore : Oct 1 17:29:47 2020 GMT
Client Root Cert notAfter : Sep 26 17:29:47 2040 GMT

Max concurrent calls : 14
Listen calls : 1
Active calls : 0

Number of created calls : 1
Number of bad calls : 0

Subscription stream/once/poll : 0/0/0

Max gNMI::Get concurrent : 5
Max grpc message size : 8388608
gNMI Synchronous calls : 0
gNMI Synchronous errors : 0
gNMI Adapter errors : 0
gNMI Dtx errors : 0
```

任意の gNMI クライアントのクライアント証明書認証を使用した接続の確認

クライアント証明書は、秘密キー（pkey）と CA チェーン（cchain）を使用して要求を行います。現在では、パスワードはオプションです。

Performing GetRequest, encoding = JSON to 172.19.199.xxx with the following gNMI Path

```
-----
[elem {
  name: "System"
}
elem {
  name: "bgp-items"
}
]
The GetResponse is below
-----
```

```
notification {
  timestamp: 1608071208072199559
  update {
    path {
      elem {
        name: "System"
      }
      elem {
        name: "bgp-items"
      }
    }
    val {
      json_val: ""
    }
  }
}
```

gRPC からトラストポイント参照を削除するには（no コマンド）、次のコマンドを使用します。

```
[no] grpc client root certificate <my_client_trustpoints>
switch(config)# no grpc client root certificate my_client_trustpoint
```

コマンドは、gRPC エージェントのトラストポイント参照だけを削除します。トラストポイント CA 証明書は削除されません。スイッチ上の gRPC サーバーへのクライアント証明書認証を使用する接続は確立されませんが、ユーザー名とパスワードによる基本認証は通過します。



-
- （注） クライアントの証明書が中間 CA によって署名されているが、上記の構成からインポートされたルート CA によって直接署名されていない場合、grpc クライアントは、ユーザー、中間 CA 証明書、およびルート CA 証明書を含む完全な証明書チェーンを提供する必要があります。
-

Sample DME Subscription - JSON Encoding

```

gnmi-console --host <iip> --port 50051 -u <user> -p <pass> --tls --
operation=Subscribe --rpc bl_payload/once/01_subscribe_bgp_dme.json

[Subscribe]-----
### Reading from file ' bl_payload/once/01_subscribe_bgp_dme.json '

### Generating request : 1 -----
### Comment : ONCE request
### Delay : 2 sec(s) ...
### Delay : 2 sec(s) DONE
subscribe {
  subscription {
    path {
      origin: "DME"
      elem {
        name: "sys"
      }
      elem {
        name: "bgp"
      }
    }
  }
  mode: ONCE
  use_models {
    name: "DME"
    organization: "Cisco Systems, Inc."
    version: "1.0.0"
  }
}

Received response 1 -----
update {
  timestamp: 1549061991079
  update {
    path {
      elem {
        name: "sys"
      }
      elem {
        name: "bgp"
      }
    }
  }
  "\" } } } ] } } ] } } ] } } ] } } } }"
}
duplicates: 1093487956
}
}
/Received -----
Received response 2 -----
sync_response: true
/Received -----
( gnmi) [root@tm-ucs-1 gnmi-console]#

```

Sample DME Subscription - PROTO Encoding

```

gnmi-console --host >iip> --port 50051 -u <user> -p <pass> --tls --
operation=Subscribe --rpc /root/gnmi-console/testing_bl/once/61_subscribe_bgp_dme_gpb.json

[Subscribe]-----
### Reading from file ' /root/gnmi-console/testing_bl/once/61_subscribe_bgp_dme_gpb.json '
Wed Jun 26 11:49:17 2019
### Generating request : 1 -----
### Comment : ONCE request
### Delay : 2 sec(s) ...
### Delay : 2 sec(s) DONE
subscribe {
  subscription {
    path {
      origin: "DME"
      elem {
        name: "sys"
      }
      elem {
        name: "bgp"
      }
    }
    mode: SAMPLE
  }
  mode: ONCE
  use_models {
    name: "DME"
    organization: "Cisco Systems, Inc."
    version: "1.0.0"
  }
  encoding: PROTO
}
Wed Jun 26 11:49:19 2019
Received response 1 -----
update {
  timestamp: 1561574967761
  prefix {
    elem {
      name: "sys"
    }
    elem {
      name: "bgp"
    }
  }
  update {
    path {
      elem {
      }
      elem {
        name: "version_str"
      }
    }
    val {
      string_val: "1.0.0"
    }
  }
  update {
    path {

```

```

elem {
}
elem {
  name: "node_id_str"
}
}
val {
  string_val: "n9k-tm2"
}
}
update {
  path {
    elem {
    }
    elem {
      name: "encoding_path"
    }
  }
  val {
    string_val: "sys/bgp"
  }
}
update {
  path {
    elem {
    }
    elem {
      /Received -----
      Wed Jun 26 11:49:19 2019
      Received response 2 -----
      sync_response: true
      /Received -----
      (_gnmi) [root@tm-ucs-1 gnmi-console]#

```

Capabilities

About Capabilities

The Capabilities RPC returns the list of capabilities of the gNMI service. The response message to the RPC request includes the gNMI service version, the versioned data models, and data encodings supported by the server.

Guidelines and Limitations for Capabilities

Following are the guidelines and limitations for Capabilities:

- Beginning with Cisco NX-OS Release 9.3(3), Capabilities supports the OpenConfig model.
- The gNMI feature supports Subscribe and Capability as options of the gNMI service.
- The feature supports JSON and gnmi.proto encoding. The feature does not support protobuf.any encoding.
- Each gNMI message has a maximum size of 12 MB. If the amount of collected data exceeds the 12-MB maximum, the collected data is dropped.

You can avoid this situation by creating more focused subscriptions that handle smaller, more granular data-collection sets. So, instead of subscribing to one higher-level path, create multiple subscriptions for different, lower-level parts of the path.

- All paths within the same subscription request must have the same sample interval. If the same path requires different sample intervals, create multiple subscriptions.
- The feature does not support a path prefix in the Subscription request, but the Subscription can contain an empty prefix field.
- The feature supports Cisco DME and Device YANG data models. Openconfig YANG is not supported.
- The gRPC process that supports gNMI uses the HIGH_PRIO cgroup, which limits the CPU usage to 75% of CPU and memory to 1.5 GB.
- The **show grpc gnmi** command has the following considerations:
 - The commands are not XMLized in this release.
 - The gRPC agent retains gNMI calls for a maximum of 1 hour after the call has ended.
 - If the total number of calls exceeds 2000, the gRPC agent purges ended calls based on an internal cleanup routine.

The gRPC server runs in the management VRF. As a result, the gRPC process communicates only in this VRF forcing the management interface to support all gRPC calls.

gRPC functionality now includes the default VRF for a total of 2 gRPC servers on each switch. You can run one gRPC server in each VRF, or run only one gRPC server in the management VRF. Supporting a gRPC in the default VRF adds flexibility to offload processing gRPC calls from the management VRF, where significant traffic load might not be desirable.

If two gRPC servers are configured, be aware of the following:

- VRF boundaries are strictly enforced, so each gRPC server processes requests independent of the other, and requests do not cross between VRFs.
- The two servers are not HA or fault tolerant. One gRPC server does not back up the other, and there is no switchover or switchback between them.
- Any limits for the gRPC server are per VRF.

Example Client Output for Capabilities

In this example, all the OpenConfig model RPMs have been installed on the switch.

The following is an example of client output for Capabilities.

```
hostname user$ ./gnmi_cli -a 172.19.193.166:50051 -ca_crt ./grpc.pem -insecure -capabilities
supported_models: <
  name: "Cisco-NX-OS-device"
  organization: "Cisco Systems, Inc."
  version: "2019-11-13"
>
supported_models: <
  name: "openconfig-acl"
  organization: "OpenConfig working group"
  version: "1.0.0"
```



```
>
supported_models: <
  name: "openconfig-bgp-policy"
  organization: "OpenConfig working group"
  version: "4.0.1"
>
supported_models: <
  name: "openconfig-interfaces"
  organization: "OpenConfig working group"
  version: "2.0.0"
>
supported_models: <
  name: "openconfig-if-aggregate"
  organization: "OpenConfig working group"
  version: "2.0.0"
>
supported_models: <
  name: "openconfig-if-ethernet"
  organization: "OpenConfig working group"
  version: "2.0.0"
>
supported_models: <
  name: "openconfig-if-ip"
  organization: "OpenConfig working group"
  version: "2.3.0"
>
supported_models: <
  name: "openconfig-if-ip-ext"
  organization: "OpenConfig working group"
  version: "2.3.0"
>
supported_models: <
  name: "openconfig-lacp"
  organization: "OpenConfig working group"
  version: "1.0.2"
>
supported_models: <
  name: "openconfig-lldp"
  organization: "OpenConfig working group"
  version: "0.2.1"
>
supported_models: <
  name: "openconfig-network-instance"
  organization: "OpenConfig working group"
  version: "0.11.1"
>
supported_models: <
  name: "openconfig-network-instance-policy"
  organization: "OpenConfig working group"
  version: "0.1.1"
>
supported_models: <
  name: "openconfig-ospf-policy"
  organization: "OpenConfig working group"
  version: "0.1.1"
>
supported_models: <
  name: "openconfig-platform"
  organization: "OpenConfig working group"
  version: "0.12.2"
>
supported_models: <
  name: "openconfig-platform-cpu"
  organization: "OpenConfig working group"
```

```

    version: "0.1.1"
  >
  supported_models: <
    name: "openconfig-platform-fan"
    organization: "OpenConfig working group"
    version: "0.1.1"
  >
  supported_models: <
    name: "openconfig-platform-linecard"
    organization: "OpenConfig working group"
    version: "0.1.1"
  >
  supported_models: <
    name: "openconfig-platform-port"
    organization: "OpenConfig working group"
    version: "0.3.2"
  >
  supported_models: <
    name: "openconfig-platform-psu"
    organization: "OpenConfig working group"
    version: "0.2.1"
  >
  supported_models: <
    name: "openconfig-platform-transceiver"
    organization: "OpenConfig working group"
    version: "0.7.0"
  >
  supported_models: <
    name: "openconfig-relay-agent"
    organization: "OpenConfig working group"
    version: "0.1.0"
  >
  supported_models: <
    name: "openconfig-routing-policy"
    organization: "OpenConfig working group"
    version: "2.0.1"
  >
  supported_models: <
    name: "openconfig-spanning-tree"
    organization: "OpenConfig working group"
    version: "0.2.0"
  >
  supported_models: <
    name: "openconfig-system"
    organization: "OpenConfig working group"
    version: "0.3.0"
  >
  supported_models: <
    name: "openconfig-telemetry"
    organization: "OpenConfig working group"
    version: "0.5.1"
  >
  supported_models: <
    name: "openconfig-vlan"
    organization: "OpenConfig working group"
    version: "3.0.2"
  >
  supported_models: <
    name: "DME"
    organization: "Cisco Systems, Inc."
  >
  supported_models: <
    name: "Cisco-NX-OS-Syslog-oper"
    organization: "Cisco Systems, Inc."

```

```

    version: "2019-08-15"
>
supported_encodings: JSON
supported_encodings: PROTO
gNMI_version: "0.5.0"

hostname user$

```

Get

About Get

The purpose of the Get RPC is to allow a client to retrieve a snapshot of the data tree from the device. Multiple paths may be requested in a single request. A simplified form of XPATH according to the gNMI Path Conventions, [Schema path encoding conventions for gNMI](#) are used for the path.

For detailed information on the Get operation, refer to the Retrieving Snapshots of State Information section in the gNMI specification: [gRPC Network Management Interface \(gNMI\)](#)

Guidelines and Limitations for Get

The following are guidelines and limitations for Get and Set:

- GetRequest.encoding supports only JSON.
- For GetRequest.type, only DataType CONFIG and STATE have direct correlation and expression in YANG. OPERATIONAL is not supported.
- A single request cannot have both OpenConfig (OC) YANG and device YANG paths. A request must have only OC YANG paths or device YANG paths, but not both.
- GetRequest for root path ("/": everything from **all** models) is not allowed.
- GetRequest for the top level of the device model ("/System") is not allowed.
- gNMI Get returns all default values (ref. report-all mode in [RFC 6243](#) [4]).
- Subscribe supports the model `Cisco-NX-OS-syslog-oper`.
- Get does not support the model `Cisco-NX-OS-syslog-oper`.
- Query from the path `/system` does not return data from the path `/system/processes`. The specific path `/system/processes` should be used to query `openconfig-procmon` data.
- The following optional items are not supported:
 - Path prefix
 - Path alias
 - Wildcards in path
- A single GetRequest can have up to 10 paths.

- If the size of value field to be returned in GetResponse is over 12 MB, the system returns error status `grpc::RESOURCE_EXHAUSTED`.
- The maximum gRPC receive buffer size is set to 8 MB.
- The number of total concurrent sessions for Get is limited to five.
- Performing a Get operation when a large configuration is applied to the switch might cause the gRPC process to consume all available memory. If a memory exhaustion condition is hit, the following syslog is generated:

```
MTX-API: The memory usage is reaching the max memory resource limit (3072) MB
```

If this condition is hit several times consecutively, the following syslog is generated:

```
The process has become unstable and the feature should be restarted.
```

We recommend that you restart the gRPC feature at this point to continue normal processing of gNMI transactions.

Set

About Set

The Set RPC is used by a client to change the configuration of the device. The operations, which may be applied to the device data, are (in order) delete, replace, and update. All operations in a single Set request are treated as a transaction, meaning that all operations are successful or the device is rolled-back to the original state. The Set operations are applied in the order that is specified in the SetRequest. If a path is mentioned multiple times, the changes are applied even if they overwrite each other. The final state of the data is achieved with the final operation in the transaction. It is assumed that all paths specified in the SetRequest::delete, replace, update fields are CONFIG data paths and writable by the client.

For detailed information on the Set operation, refer to the Modifying State section of the gNMI Specification <https://github.com/openconfig/reference/blob/1cf43d2146f9ba70abb7f04f6b0f6caa504cef05/rpc/gnmi/gnmi-specification.md>.

Guidelines and Limitations for Set

The following are guidelines and limitations for Set:

- SetRequest.encoding supports only JSON.
- A single request cannot have both OpenConfig (OC) YANG and device YANG paths. A request must have only OC YANG paths or device YANG paths, but not both.
- Subscribe supports the model `Cisco-NX-OS-syslog-oper`.
- Query from the path `/system` does not return data from the path `/system/processes`. The specific path `/system/processes` should be used to query `openconfig-procmon` data.
- The following optional items are not supported:
 - Path prefix
 - Path alias

- Wildcards in path
- A single SetRequest can have up to 20 paths.
- The maximum gRPC receive buffer size is set to 8 MB.
- The number of total concurrent sessions for Get is limited to five.
- Performing a Set operation when a large configuration is applied to the switch might cause the gRPC process to consume all available memory. If a memory exhaustion condition is hit, the following syslog is generated:

```
MTX-API: The memory usage is reaching the max memory resource limit (3072) MB
```

If this condition is hit several times consecutively, the following syslog is generated:

```
The process has become unstable and the feature should be restarted.
```

We recommend that you restart the gRPC feature at this point to continue normal processing of gNMI transactions.

- For the Set::Delete RPC, an MTX log message warns if the configuration being operated on may be too large:

```
Configuration size for this namespace exceeds operational limit. Feature may become unstable and require restart.
```

Subscribe

Guidelines and Limitations for Subscribe

Following are the guidelines and limitations for Subscribe:

- Beginning with Cisco NX-OS Release 9.3(3), Subscribe supports the OpenConfig model.
- The gNMI feature supports Subscribe and Capability as options of the gNMI service.
- The feature supports JSON and gnmi.proto encoding. The feature does not support protobuf.any encoding.
- Each gNMI message has a maximum size of 12 MB. If the amount of collected data exceeds the 12-MB maximum, the collected data is dropped.

You can avoid this situation by creating more focused subscriptions that handle smaller, more granular data-collection sets. So, instead of subscribing to one higher-level path, create multiple subscriptions for different, lower-level parts of the path.

- All paths within the same subscription request must have the same sample interval. If the same path requires different sample intervals, create multiple subscriptions.
- The feature does not support a path prefix in the Subscription request, but the Subscription can contain an empty prefix field.
- The feature supports Cisco DME and Device YANG data models. Openconfig YANG is not supported.
- The gRPC process that supports gNMI uses the HIGH_PRIO cgroup, which limits the CPU usage to 75% of CPU and memory to 1.5 GB.

- The **show grpc gnmi** command has the following considerations:
 - The commands are not XMLized in this release.
 - The gRPC agent retains gNMI calls for a maximum of 1 hour after the call has ended.
 - If the total number of calls exceeds 2000, the gRPC agent purges ended calls based on an internal cleanup routine.

The gRPC server runs in the management VRF. As a result, the gRPC process communicates only in this VRF forcing the management interface to support all gRPC calls.

gRPC functionality now includes the default VRF for a total of 2 gRPC servers on each switch. You can run one gRPC server in each VRF, or run only one gRPC server in the management VRF. Supporting a gRPC in the default VRF adds flexibility to offload processing gRPC calls from the management VRF, where significant traffic load might not be desirable.

If two gRPC servers are configured, be aware of the following:

- VRF boundaries are strictly enforced, so each gRPC server processes requests independent of the other, and requests do not cross between VRFs.
- The two servers are not HA or fault tolerant. One gRPC server does not back up the other, and there is no switchover or switchback between them.
- Any limits for the gRPC server are per VRF.

gNMI Payload

gNMI uses a specific payload format to subscribe to:

- DME Streams
- YANG Streams

Subscribe operations are supported with the following modes:

- ONCE: Subscribe and receive data once and close session.
- POLL: Subscribe and keep session open, client sends poll request each time data is needed.
- STREAM: Subscribe and receive data at specific cadence. The payload accepts values in nanoseconds
1 second = 1000000000.
- ON_CHANGE: Subscribe, receive a snapshot, and only receive data when something changes in the tree.

Setting modes:

- Each mode requires 2 settings, inside sub and outside sub
- ONCE: SAMPLE, ONCE
- POLL: SAMPLE, POLL
- STREAM: SAMPLE, STREAM
- ON_CHANGE: ON_CHANGE, STREAM

Origin

- DME: Subscribing to DME model
- device: Subscribing to YANG model

Name

- DME = subscribing to DME model
- Cisco-NX-OS-device = subscribing to YANG model

Encoding

- JSON = Stream will be send in JSON format.
- PROTO = Stream will be sent in protobuf.any format.

Sample gNMI Payload for DME Stream



Note Different clients have their own input format.

```
{
  "SubscribeRequest":
  [
    {
      "_comment" : "ONCE request",
      "_delay" : 2,
      "subscribe":
      {
        "subscription":
        [
          {
            "_comment" : "1st subscription path",
            "path":
            {
              "origin": "DME",
              "elem":
              [
                {
                  "name": "sys"
                },
                {
                  "name": "bgp"
                }
              ]
            },
            "mode": "SAMPLE"
          }
        ],
        "mode": "ONCE",
        "allow_aggregation" : false,
        "use_models":
        [
          {
            "_comment" : "1st module",
            "name": "DME",
            "organization": "Cisco Systems, Inc.",
            "version": "1.0.0"
          }
        ]
      }
    }
  ]
}
```

```

    }
    ],
    "encoding": "JSON"
  }
}
]
}

```

Sample gNMI Payload YANG Stream

```

{
  "SubscribeRequest":
  [
    {
      "_comment" : "ONCE request",
      "_delay" : 2,
      "subscribe":
      {
        "subscription":
        [
          {
            "_comment" : "1st subscription path",
            "path":
            {
              "origin": "device",
              "elem":
              [
                {
                  "name": "System"
                },
                {
                  "name": "bgp-items"
                }
              ]
            },
            "mode": "SAMPLE"
          }
        ],
        "mode": "ONCE",
        "allow_aggregation" : false,
        "use_models":
        [
          {
            "_comment" : "1st module",
            "name": "Cisco-NX-OS-device",
            "organization": "Cisco Systems, Inc.",
            "version": "0.0.0"
          }
        ],
        "encoding": "JSON"
      }
    }
  ]
}

```


Streaming Syslog

About Streaming Syslog for gNMI

gRPC Network Management Interface (gNMI) defines a gRPC-based protocol for the modification and retrieval of configuration from a target device. Also, it defines the control and generation of telemetry streams from a target device to a data collection system. The intention is that a single gRPC service definition can cover both configuration and telemetry - allowing a single implementation on the target, as well as a single NMS element to interact with the device via telemetry and configuration RPCs.

gNMI Subscribe is a new way of monitoring the network as it provides a real-time view of what's going on in your system by pushing the structured data as per gNMI Subscribe request.

Beginning with the Cisco NX-OS Release 9.3(3), support is added for gNMI Subscribe functionality.

gNMI Subscribe Support Detail

- Syslog-oper model streaming
 - stream_on_change

Telemetry Support Detail

- YANG model
 - Events only

This feature applies to Cisco Nexus 3500 platform switches with 8 GB or more of memory.

Guidelines and Limitations for Streaming Syslog - gNMI

The following are guidelines and limitations for Streaming Syslog:

- An invalid syslog is not supported. For example, a syslog with a filter or query condition
- Only the following paths are supported:
 - Cisco-NX-OS-Syslog-oper:syslog
 - Cisco-NX-OS-Syslog-oper:syslog/messages
- The following modes are not supported:
 - Stream sample
 - POLL
- A request must be in the YANG model format.
- You can use the internal application or write your own application.
- The payload comes from the controller and gNMI sends a response.
- Encoding formats are JSON and PROTO.

Syslog Native YANG Model

The YangModels are located [here](#).



Note The time-zone field is set only when the **clock format show-timezone syslog** is entered. By default, it's not set, therefore the time-zone field is empty.

```
PYANG Tree for Syslog Native Yang Model:
>>> pyang -f tree Cisco-NX-OS-infra-syslog-oper.yang
module: Cisco-NX-OS-syslog-oper
+--ro syslog
+---ro messages
+---ro message* [message-id]
+---ro message-id int32
+---ro node-name? string
+---ro time-stamp? uint64
+---ro time-of-day? string
+---ro time-zone? string
+---ro category? string
+---ro group? string
+---ro message-name? string
+---ro severity? System-message-severity
+---ro text? string
```

Subscribe Request Example

The following is an example of a Subscribe request:

```
{
  "SubscribeRequest":
  [
    {
      "_comment" : "STREAM request",
      "_delay" : 2,
      "subscribe":
      {
        "subscription":
        [
          {
            "_comment" : "1st subscription path",
            "path":
            {
              "origin": "syslog-oper",
              "elem":
              [
                {
                  "name": "syslog"
                },
                {
                  "name": "messages"
                }
              ]
            }
          },
          {
            "mode": "ON_CHANGE"
          }
        ],
        "mode": "ON_CHANGE",
        "allow_aggregation" : false,
      }
    }
  ]
}
```

```

        "use_models":
        [
            {
                "_comment" : "1st module",
                "name": "Cisco-NX-OS-Syslog-oper",
                "organization": "Cisco Systems, Inc.",
                "version": "0.0.0"
            }
        ],
        "encoding": "JSON"
    }
}
]
}

```

Sample PROTO Output

This is a sample of PROTO output.

#####

[Subscribe]-----

Reading from file ' /root/gnmi-console/testing_bl/stream_on_change/OC_SYSLOG.json '

Sat Aug 24 14:38:06 2019

Generating request : 1 -----

Comment : STREAM request

Delay : 2 sec(s) ...

Delay : 2 sec(s) DONE

subscribe {

subscription {

path {

origin: "syslog-oper"

elem {

name: "syslog"

}

elem {

name: "messages"

}

}

mode: ON_CHANGE

}

use_models {

```

name: "Cisco-NX-OS-Syslog-oper"

organization: "Cisco Systems, Inc."

version: "0.0.0"

}

encoding: PROTO

}

Thu Nov 21 14:26:41 2019
Received response 3 -----
update {
timestamp: 1574375201665688000
prefix {
origin: "Syslog-oper"
elem {
name: "syslog"
}
elem {
name: "messages"
}
}
update {
path {
elem {
name: "message-id"
}
}
val {
uint_val: 529
}
}
update {
path {
elem {
name: "node-name"
}
}
val {
string_val: "task-n9k-1"
}
}
update {
path {
elem {
name: "message-name"
}
}
val {
string_val: "VSHD_SYSLOG_CONFIG_I"
}
}
update {
path {
elem {
name: "text"
}
}
val {
string_val: "Configured from vty by admin on console0"
}
}

```

```
}
update {
  path {
    elem {
      name: "group"
    }
  }
  val {
    string_val: "VSHD"
  }
}
update {
  path {
    elem {
      name: "category"
    }
  }
  val {
    string_val: "VSHD"
  }
}
update {
  path {
    elem {
      name: "time-of-day"
    }
  }
  val {
    string_val: "Nov 21 2019 14:26:40"
  }
}
update {
  path {
    elem {
      name: "time-zone"
    }
  }
  val {
    string_val: ""
  }
}
update {
  path {
    elem {
      name: "time-stamp"
    }
  }
  val {
    uint_val: 1574375200000
  }
}
update {
  path {
    elem {
      name: "severity"
    }
  }
  val {
    uint_val: 5
  }
}
}

/Received -----
```

Sample JSON Output

This is a sample JSON output.

```
[Subscribe]-----
### Reading from file ' testing_bl/stream_on_change/OC_SYSLOG.json '

Tue Nov 26 11:47:00 2019
### Generating request : 1 -----
### Comment : STREAM request
### Delay : 2 sec(s) ...
### Delay : 2 sec(s) DONE
subscribe {
  subscription {
    path {
      origin: "syslog-oper"
      elem {
        name: "syslog"
      }
      elem {
        name: "messages"
      }
    }
    mode: ON_CHANGE
  }
  use_models {
    name: "Cisco-NX-OS-Syslog-oper"
    organization: "Cisco Systems, Inc."
    version: "0.0.0"
  }
}

Tue Nov 26 11:47:15 2019
Received response 5 -----
update {
  timestamp: 1574797636002053000
  prefix {
  }
  update {
    path {
      origin: "Syslog-oper"
      elem {
        name: "syslog"
      }
    }
    val {
      json_val: "[ { \"messages\" : [[
        {\"message-id\":657},{\"node-name\": \"task-n9k-1\", \"time-stamp\": \"1574797635000\", \"time-of-day\": \"Nov
        26 2019
        11:47:15\", \"severity\":3, \"message-name\": \"HDR_L2LEN_ERR\", \"category\": \"ARP\", \"group\": \"ARP\", \"text\": \"arp
        [30318] Received packet with incorrect layer 2 address length (8 bytes), Normal pkt with
        S/D MAC: 003a.7d21.d55e ffff.ffff.ffff eff_ifc mgmt0(9), log_ifc mgmt0(9), phy_ifc
        mgmt0(9)\", \"time-zone\": \"\"} ]] } ]"
    }
  }
}

/Received -----
```

Troubleshooting

Gathering TM-Trace Logs

```

1. tmtrace.bin -f gnmi-logs gnmi-events gnmi-errors following are available
2. Usage:

bash-4.3# tmtrace.bin -d gnmi-events | tail -30 Gives the last 30
}
}
}
[06/21/19 15:58:38.969 PDT f8f 3133] [3981658944][tm_transport_internal.c:43] dn:
Cisco-NX-OS-device:System/cdp-items, sub_id: 0,
sub_id_str: 2329, dc_start_time: 0, length: 124, sync_response:1
[06/21/19 15:58:43.210 PDT f90 3133] [3621780288][tm_ec_yang_data_processor.c:93] TM_EC:
[Y] Data received for 2799743488: 49
{
"cdp-items" : {
"inst-items" : {
"if-items" : {
"If-list" : [
{
"id" : "mgmt0",
"ifstats-items" : {
"v2Sent" : "74",
"validV2Rcvd" : "79"
}
}
}
}
}
}
}
[06/21/19 15:58:43.210 PDT f91 3133] [3981658944][tm_transport_internal.c:43] dn:
Cisco-NX-OS-device:System/cdp-items, sub_id: 0,
sub_id_str: 2329, dc_start_time: 0, length: 141, sync_response:1
[06/21/19 15:59:01.341 PDT f92 3133] [3981658944][tm_transport_internal.c:43] dn:
Cisco-NX-OS-device:System/intf-items, sub_id:
4091, sub_id_str: , dc_start_time: 1561157935518, length: 3063619, sync_response:0
[06/21/19 15:59:03.933 PDT f93 3133] [3981658944][tm_transport_internal.c:43] dn:
Cisco-NX-OS-device:System/cdp-items, sub_id:
4091, sub_id_str: , dc_start_time: 1561157940881, length: 6756, sync_response:0
[06/21/19 15:59:03.940 PDT f94 3133] [3981658944][tm_transport_internal.c:43] dn:
Cisco-NX-OS-device:System/lldp-items, sub_id:
4091, sub_id_str: , dc_start_time: 1561157940912, length: 8466, sync_response:1
bash-4.3#

```

Gathering MTX-Internal Logs

1. Modify the following file with below /opt/mtx/conf/mtxlogger.cfg

```

<config name="nxos-device-mgmt">
  <container name="mgmtConf">
    <container name="logging">
      <leaf name="enabled" type="boolean" default="false">true</leaf>
      <leaf name="allActive" type="boolean" default="false">true<
/leaf>
    <container name="format">

```

```

        <leaf name="content" type="string" default="$DATETIME$
$COMPONENTID$ $TYPE$: $MSG$">$DATETIME$ $COMPONENTID$ $TYPE$
$SRCFILE$ @ $SRCLINE$ $FCNINFO$: $MSG$</leaf>
        <container name="componentID">
            <leaf name="enabled" type="boolean" default="true"></leaf>
        </container>
        <container name="dateTime">
            <leaf name="enabled" type="boolean" default="true"></leaf>
            <leaf name="format" type="string" default="%y%m%d.%H%M%S"><
/leaf>
            </container>
            <container name="fcn">
                <leaf name="enabled" type="boolean" default="true"></leaf>
                <leaf name="format" type="string"
default="$CLASS$: $FCNNAME$ ($ARGS$) @$LINE$"></leaf>
                </container>
            </container>
            <container name="facility">
                <leaf name="info" type="boolean" default="true">true</leaf>
                <leaf name="warning" type="boolean" default="true">true<
/leaf>
                <leaf name="error" type="boolean" default="true">true</leaf>
                <leaf name="debug" type="boolean" default="false">true<
/leaf>
                </container>
                <container name="dest">
                    <container name="console">
                        <leaf name="enabled" type="boolean" default="false">true<
/leaf>
                    </container>
                    <container name="file">
                        <leaf name="enabled" type="boolean" default="false">true<
/leaf>
                    <leaf name="name" type="string" default="mtx-internal.log"><
/leaf>

        <leaf name="location" type="string" default="./mtxlogs">
/volatile</leaf>
            <leaf name="mbytes-rollover" type="uint32" default="10"
>50</leaf>
            <leaf name="hours-rollover" type="uint32" default="24"
>24</leaf>
            <leaf name="startup-rollover" type="boolean" default="
false">true</leaf>
            <leaf name="max-rollover-files" type="uint32" default="10"
>10</leaf>
            </container>
            </container>
            <list name="logitems" key="id">
                <listitem>
                    <leaf name="id" type="string">*</leaf>
                    <leaf name="active" type="boolean" default="false"
>false</leaf>
                </listitem>
                <listitem>
                    <leaf name="id" type="string">MTX-EvtMgr</leaf>
                    <leaf name="active" type="boolean" default="true"
>true</leaf>
                </listitem>
                <listitem>
                    <leaf name="id" type="string">TM-ADPT</leaf>
                    <leaf name="active" type="boolean" default="true"
>false</leaf>

```



```

        </listitem>
      </listitem>
      <leaf name="id" type="string">TM-ADPT-JSON</leaf>
      <leaf name="active" type="boolean" default="true"
>false</leaf>
    </listitem>
    <listitem>
      <leaf name="id" type="string">SYSTEM</leaf>
      <leaf name="active" type="boolean" default="true"
>true</leaf>
    </listitem>
    <listitem>
      <leaf name="id" type="string">LIBUTILS</leaf>
      <leaf name="active" type="boolean" default="true"
>true</leaf>
    </listitem>
    <listitem>
      <leaf name="id" type="string">MTX-API</leaf>
      <leaf name="active" type="boolean" default="true"
>true</leaf>
    </listitem>
    <listitem>
      <leaf name="id" type="string">Model-*</leaf>
      <leaf name="active" type="boolean" default="true"
>true</leaf>
    </listitem>
    <listitem>
      <leaf name="id" type="string">Model-Cisco-NX-OS-
device</leaf>
      <leaf name="active" type="boolean" default="true"
>false</leaf>
    </listitem>
    <listitem>
      <leaf name="id" type="string">Model-openconfig-bgp<
/leaf>
      <leaf name="active" type="boolean" default="true"
>false</leaf>
    </listitem>
    <listitem>
      <leaf name="id" type="string">INST-MTX-API</leaf>
      <leaf name="active" type="boolean" default="true"
>true</leaf>
    </listitem>
    <listitem>
      <leaf name="id" type="string">INST-ADAPTER-NC</leaf>
      <leaf name="active" type="boolean" default="true"
>true</leaf>
    </listitem>
    <listitem>
      <leaf name="id" type="string">INST-ADAPTER-RC</leaf>
      <leaf name="active" type="boolean" default="true"
>true</leaf>
    </listitem>
    <listitem>
      <leaf name="id" type="string">INST-ADAPTER-GRPC</leaf>
      <leaf name="active" type="boolean" default="true"
>true</leaf>
    </listitem>
  </list>
</container>
</container>
</config>

```

2. Run "no feature grpc" / "feature grpc"

3. The /volataile directory houses the mtx-internal.log, the log rolls over over time so be sure to grab what you need before thenbash-4.3# cd /volatile/

```
bash-4.3# cd /volaiflels -al
total 148
drwxrwxrwx 4 root root 340 Jun 21 15:47 .
drwxrwxr-t 64 root network-admin 1600 Jun 21 14:45 ..
-rw-rw-rw- 1 root root 103412 Jun 21 16:14 grpc-internal-log
-rw-r--r-- 1 root root 24 Jun 21 14:44 mtx-internal-19-06-21-14-46-21.log
-rw-r--r-- 1 root root 24 Jun 21 14:46 mtx-internal-19-06-21-14-46-46.log
-rw-r--r-- 1 root root 175 Jun 21 15:11 mtx-internal-19-06-21-15-11-57.log
-rw-r--r-- 1 root root 175 Jun 21 15:12 mtx-internal-19-06-21-15-12-28.log
-rw-r--r-- 1 root root 175 Jun 21 15:13 mtx-internal-19-06-21-15-13-17.log
-rw-r--r-- 1 root root 175 Jun 21 15:13 mtx-internal-19-06-21-15-13-42.log
-rw-r--r-- 1 root root 24 Jun 21 15:13 mtx-internal-19-06-21-15-14-22.log
-rw-r--r-- 1 root root 24 Jun 21 15:14 mtx-internal-19-06-21-15-19-05.log
-rw-r--r-- 1 root root 24 Jun 21 15:19 mtx-internal-19-06-21-15-47-09.log
-rw-r--r-- 1 root root 24 Jun 21 15:47 mtx-internal.log
-rw-rw-rw- 1 root root 355 Jun 21 14:44 netconf-internal-log
-rw-rw-rw- 1 root root 0 Jun 21 14:45 nginx_logflag
drwxrwxrwx 3 root root 60 Jun 21 14:45 uwsgipy
drwxrwxrwx 2 root root 40 Jun 21 14:43 virtual-instance
bash-4.3#.
```



第 24 章

gNOI-gRPC ネットワーク操作インターフェイス

- gNOI について (319 ページ)
- サポートされる gNOI RPC (320 ページ)
- システム proto (320 ページ)
- OS プロトコル (322 ページ)
- 証明書 Proto (323 ページ)
- ファイル Proto (323 ページ)
- gNOI 工場リセット (324 ページ)
- 注意事項と制約事項 (325 ページ)
- gNOI の確認 (326 ページ)

gNOI について

gRPC ネットワーク オペレーション インターフェイス (gNOI) は、ネットワーク デバイス 上で操作 コマンド を実行 するための gRPC ベース のマイクロ サービス セット を定義 します。サポート されている 操作 コマンド は、Ping、Traceroute、Time、SwitchControlProcessor、Reboot、RebootStatus、CancelReboot、Activate、および Verify です。

gNOI は gRPC を トランスポート プロトコル として 使用 し、設定 は gNMI の設定 と同じ です。設定 の詳細 については、「gNMI の設定」を参照 してください。https://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus9000/sw/93x/programmability/guide/b-cisco-nexus-9000-series-nx-os-programmability-guide-93x/b-cisco-nexus-9000-series-nx-os-programmability-guide-93x_chapter_0110001.html#id_107728

gNOI RPC 要求 を送信 するには、各 RPC に gNOI クライアント インターフェイス を実装 する クライアント が必要です。

Cisco NX-OS リリース 10.1(1) では、gNOI は限られた 数のコンポーネント に対して リモート プロシージャ コール (RPC) を定義 し、その一部 はハードウェア (光 インターフェイス など) に関連 しています。

Proto ファイルは gRPC マイクロサービス用に定義されており、GitHub で入手できます。
<https://github.com/openconfig/gnoi>

サポートされる gNOI RPC

サポートされている gNOI RPC は次のとおりです。

表 30:

プロトコル	gNOI RPC	サポート対象
System	ping	○
	トレースルート	はい
	時間	はい
	SwitchControl プロセッサ	はい
	リブート	はい
	RebootStatus	はい
	CancelReboot	はい
OS	アクティブ化	はい
	インターフェイス	はい
Cert	LoadCertificate	はい
File	get	はい
	Stat	はい
	削除	対応

システム proto

システム proto サービスは、設定およびテレメトリ パイプラインの外部でターゲットを管理できるようにする操作可能な RPC のコレクションです。

次に、システム proto の RPC サポートの詳細を示します。

RPC	サポート	説明	制限事項
ping	ping/ping6 cli コマンド	ターゲットで ping コマンドを実行し、結果をストリームバックします。一部のターゲットでは、すべての結果が使用可能になるまで結果がストリーミングされない場合があります。パケット数が明示的に指定されていない場合は、5 が使用されます。	do_not_resolve オプションはサポートされていません。
トレースルート	tracert/tracert6 cli コマンド	ターゲットで tracert コマンドを実行し、結果をストリームバックします。一部のターゲットでは、すべての結果が使用可能になるまで結果がストリーミングされない場合があります。最大ホップカウント 30 が使用されます。	initial_ttl、max_ttl、wait、do_not_fragment、do_not_resolve、および l4protocol オプションはサポートされていません。
時間	ローカル時刻	ターゲットの現在の時刻を返します。通常、ターゲットが応答しているかどうかをテストするために使用されます。	-
SwitchControl プロセッサ	system switchover cli コマンド	現在のルートプロセッサから指定されたルートプロセッサに切り替えます。スイッチオーバーは即座に発生します。応答がクライアントに返されることが保証されない場合があります。	スイッチオーバーは即座に発生します。その結果、応答がクライアントに返されることが保証されない場合があります。

RPC	サポート	説明	制限事項
リブート	reload module	ターゲットをリブートします。	message オプションはサポートされません。 delay オプションはスイッチのリロードでサポートされます。path オプションは1つのモジュール番号を受け入れます。
RebootStatus	show version [module] cli コマンド	ターゲットのリブートのステータスを返します。	-
CancelReboot	reload cancel	保留中の再起動要求をキャンセルします。	-



(注) SetPackage RPC はサポートされていません。

OS プロトコル

OS サービスは、ターゲット上の OS インストールに対するインターフェイスを提供します。OS パッケージのファイル形式は、プラットフォームによって異なります。プラットフォームは、提供された OS パッケージが有効でブート可能であることを検証する必要があります。これには、既知の良好なハッシュに対するハッシュチェックを含める必要があります。ハッシュは OS パッケージに埋め込むことをお勧めします。

ターゲットは、独自の永続ストレージと OS インストールプロセスを管理します。一連の個別の OS パッケージを保存し、着信する新しい OS パッケージ用に常にプロアクティブにスペースを解放します。ターゲットには、有効な着信 OS パッケージ用の十分なスペースが常にあることが保証されます。現在実行中の OS パッケージは削除しないでください。クライアントは、最後に正常にインストールされたパッケージが使用可能であることを期待する必要があります。

次に、OS プロトコルの RPC サポートの詳細を示します。

RPC	サポート	説明	制限事項
アクティブ化	install all nxos bootflash:///img_name	要求された OS バージョンを、次のリブート時に使用されるバージョンとして設定します。この RPC は、ターゲットを再起動します。	再起動に失敗した場合は、ロールバックまたは回復できません。
検証	show version	[検証 (Verify)] は、実行中の OS バージョンを確認します。この RPC は、ターゲットの起動中に成功するまで複数回呼び出される場合があります。	-



(注) インストール RPC はサポートされていません。

証明書 Proto

証明書管理サービスは、ターゲットによってエクスポートされます。ローテーション、インストール、およびその他の証明書プロトコル RPC はサポートされていません。

次に、Cert proto の RPC サポートの詳細を示します。

RPC	サポート	説明	制限事項
LoadCertificate	crypto ca import <trustpoint> pkcs12 <file> <passphrase>	CA 証明書のバンドルをロードします。	-

ファイル Proto

ファイル proto は、file.proto RPC の機能に基づいてメッセージをストリーミングします。ここに記載されていない Put およびその他の RPC は、ファイル Proto ではサポートされていません。

Get、Stat、および Remove RPC は、bootflash、bootflash://sup-remote、logflash、logflash://sup-remote、usb、volatile、volatile://sup-remote、および debug のファイル システムをサポートします。

次に、ファイル proto の RPC サポートの詳細を示します。

RPC	説明	制限事項
結果	Get はターゲットからファイルの内容を読み取り、ストリーミングします。ファイルは連続したメッセージによってストリーミングされます。各メッセージには最大 64KB のデータが含まれます。最後のメッセージが送信された後、送信されたデータのハッシュが送信され、ストリームが閉じられます。ファイルが存在しない場合、またはファイルの読み取り中にエラーが発生した場合は、エラーが返されます。	ファイルサイズの上限は 32MB です。
Stat	Stat は、ターゲット上のファイルに関するメタデータを返します。ファイルが存在しない場合、またはファイルのメタデータへのアクセス中にエラーが発生した場合は、エラーが返されます。	-
削除	Remove は、ターゲットから指定されたファイルを削除します。ファイルが存在しない場合、ディレクトリである場合、または削除操作でエラーが発生した場合は、エラーが返されます。	-

gNOI 工場リセット

gNOIの初期設定へのリセット操作を行うと、指定されたモジュールのすべての永続ストレージが消去されます。これには、構成、すべてのログデータ、およびフラッシュとSSDのすべての内容が含まれます。リセットは直前のブートイメージでブートし、ライセンスを含むすべてのストレージを消去します。gNOIの初期設定へのリセットは、次の2つのモードをサポートしています。

- 再フォーマットと再パーティションのみが可能な高速消去。
- データをセキュアに消去してワイプし、回復不可能にする、セキュア消去。

factory_reset.proto で定義されている gNOI の初期設定へのリセット操作は、デバイス上のすべての永続ストレージを消去します。こちらの [factory_reset.proto](https://github.com/openconfig/gnoi/blob/master/factory_reset/factory_reset.proto) リンクを参照してください:

https://github.com/openconfig/gnoi/blob/master/factory_reset/factory_reset.proto

次に、gNOI FactoryReset サービスの例を示します。

```
/ The FactoryReset service exported by Targets.
service FactoryReset {
    // The Start RPC allows the Client to instruct the Target to immediately
    // clean all existing state and boot the Target in the same condition as it is
    // shipped from factory. State includes storage, configuration, logs,
    // certificates and licenses.
    //
    // Optionally allows rolling back the OS to the same version shipped from
    // factory.
    //
    // Optionally allows for the Target to zero-fill permanent storage where state
    // data is stored.
    //
    // If any of the optional flags is set but not supported, a gRPC Status with
    // code INVALID_ARGUMENT must be returned with the details value set to a
    // properly populated ResetError message.
    rpc Start(StartRequest) returns (StartResponse);
}

message StartRequest {
    // Instructs the Target to rollback the OS to the same version as it shipped
    // from factory.
    bool factory_os = 1;
    // Instructs the Target to zero fill persistent storage state data.
    bool zero_fill = 2;
}
```

次に、gNOI の工場リセットで使用される引数の詳細を示します。

- **factory_os = false:** 工場出荷時の OS バージョンにロールバックするかどうかを指定します。NX-OS では **true** に設定することはサポートされていません。現在のブート イメージを保持する必要があります。
- **zero_fill:** 時間のかかる包括的なセキュア消去を実行するかどうかを指定します。
 - **zero_fill = true:** factory-reset module all preserve-image force を指定します。
 - **zero_fill = false:** factory-reset module all bypass-secure-erase preserve-image force を指定します。

注意事項と制約事項

gNOI 機能には、次の注意事項と制約事項があります。

- 最大 16 のアクティブな gNOI RPC がサポートされます。
- Cisco Nexus 9000 シリーズ スイッチは、1 つの gNMI サービスと 2 つの gNOI マイクロサービスを持つ 1 つのエンドポイントを実行します。

- 10.1(1) リリースでは、gNOI RPC は同等の CLI を使用して実装されます。既存の CLI 制限または有効なオプションはそのまま適用されます。
- 10.2(1)F リリース以降、file.proto および cert.proto RPC がサポートされています。
- Nexus デバイス向けの gRPC トラフィックは、デフォルト クラスのコントロールプレーンポリサー（CoPP）にヒットします。gRPC ドロップの可能性を抑えるには、管理クラスの gRPC 構成ポートを使用して、カスタム CoPP ポリシーを構成してください。

gNOI の確認

gNOI の構成を確認するには、次のコマンドを入力します。

コマンド	説明
clear grpc gnoi rpc	カウンタまたは呼び出しをクリーンアップするために使用されます。
debug grpc events {events errors} show grpc nxsdk event-history {events errors}	イベント履歴からイベントとエラーをデバッグします。
show grpc internal gnoi rpc {summary detail}	有用性のために「internal」キーワードコマンドが追加されました。



CHAPTER 25

Model Driven Telemetry

- [About Telemetry, on page 327](#)
- [Licensing Requirements for Telemetry, on page 329](#)
- [Installing and Upgrading Telemetry, on page 329](#)
- [Guidelines and Limitations for Model Driven Telemetry, on page 330](#)
- [Configuring Telemetry Using the CLI, on page 337](#)
- [Configuring Telemetry Using the NX-API, on page 358](#)
- [Telemetry Path Labels, on page 372](#)
- [Native Data Source Paths, on page 389](#)
- [Streaming Syslog, on page 401](#)
- [Additional References, on page 408](#)

About Telemetry

Collecting data for analyzing and troubleshooting has always been an important aspect in monitoring the health of a network.

Cisco NX-OS provides several mechanisms such as SNMP, CLI, and Syslog to collect data from a network. These mechanisms have limitations that restrict automation and scale. One limitation is the use of the pull model, where the initial request for data from network elements originates from the client. The pull model does not scale when there is more than one network management station (NMS) in the network. With this model, the server sends data only when clients request it. To initiate such requests, continual manual intervention is required. This continual manual intervention makes the pull model inefficient.

A push model continuously streams data out of the network and notifies the client. Telemetry enables the push model, which provides near-real-time access to monitoring data.

Telemetry Components and Process

Telemetry consists of four key elements:

- **Data Collection** — Telemetry data is collected from the Data Management Engine (DME) database in branches of the object model specified using distinguished name (DN) paths. The data can be retrieved periodically (frequency-based) or only when a change occurs in any object on a specified path (event-based). You can use the NX-API to collect frequency-based data.

- **Data Encoding** — The telemetry encoder encapsulates the collected data into the desired format for transporting.

NX-OS encodes telemetry data in the Google Protocol Buffers (GPB) and JSON format.

- **Data Transport** — NX-OS transports telemetry data using HTTP for JSON encoding and the Google remote procedure call (gRPC) protocol for GPB encoding. The gRPC receiver supports message sizes greater than 4 MB. (Telemetry data using HTTPS is also supported if a certificate is configured.)

Starting with Cisco NX-OS Release 9.2(1), telemetry now supports streaming to IPv6 destinations and IPv4 destinations.

Starting with Cisco NX-OS Release 7.0(3)I7(1), UDP and secure UDP (DTLS) are supported as telemetry transport protocols. You can add destinations that receive UDP. The encoding for UDP and secure UDP can be GPB or JSON.

Use the following command to configure the UDP transport to stream data using a datagram socket either in JSON or GPB:

```
destination-group num
  ip address xxx.xxx.xxx.xxx port xxxx protocol UDP encoding {JSON | GPB }
```

Example for an IPv4 destination:

```
destination-group 100
  ip address 171.70.55.69 port 50001 protocol UDP encoding GPB
```

Example for an IPv6 destination:

```
destination-group 100
  ipv6 address 10:10::1 port 8000 protocol gRPC encoding GPB
```

The UDP telemetry is with the following header:

```
typedef enum tm_encode_ {
    TM_ENCODE_DUMMY,
    TM_ENCODE_GPB,
    TM_ENCODE_JSON,
    TM_ENCODE_XML,
    TM_ENCODE_MAX,
} tm_encode_type_t;

typedef struct tm_pak_hdr_ {
    uint8_t version; /* 1 */
    uint8_t encoding;
    uint16_t msg_size;
    uint8_t secure;
    uint8_t padding;
} __attribute__((packed, aligned (1))) tm_pak_hdr_t;
```

Use the first 6 bytes in the payload to process telemetry data using UDP, using one of the following methods:

- Read the information in the header to determine which decoder to use to decode the data, JSON or GPB, if the receiver is meant to receive different types of data from multiple endpoints.
- Remove the header if you are expecting one decoder (JSON or GPB) but not the other.



Note Depending on the receiving operation system and the network load, using the UDP protocol may result in packet drops.

- **Telemetry Receiver** — A telemetry receiver is a remote management system or application that stores the telemetry data.

The GPB encoder stores data in a generic key-value format. The encoder requires metadata in the form of a compiled `.proto` file to translate the data into GPB format.

In order to receive and decode the data stream correctly, the receiver requires the `.proto` file that describes the encoding and the transport services. The encoding decodes the binary stream into a key value string pair.

A telemetry `.proto` file that describes the GPB encoding and gRPC transport is available on Cisco's GitLab: <https://github.com/CiscoDevNet/nx-telemetry-proto>

High Availability of the Telemetry Process

High availability of the telemetry process is supported with the following behaviors:

- **System Reload** — During a system reload, any telemetry configuration and streaming services are restored.
- **Process Restart** — If the telemetry process freezes or restarts for any reason, configuration, and streaming services are restored when telemetry is restarted.

Licensing Requirements for Telemetry

Product	License Requirement
Cisco NX-OS	Telemetry requires no license. Any feature not included in a license package is bundled with the Cisco NX-OS image and is provided at no extra charge to you. For a complete explanation of the Cisco NX-OS licensing scheme, see the <i>Cisco NX-OS Licensing Guide</i> .

Installing and Upgrading Telemetry

Installing the Application

The telemetry application is packaged as a feature RPM and included with the NX-OS release. The RPM is installed by default as part of the image bootstrap. After installation, you can start the application using the **feature telemetry** command. The RPM file is located in the `/rpms` directory and is named as follows:

telemetry-version-build_ID.libn32_n9000.rpm

telemetry-version-build_ID.libn32_n3000.rpm

As in the following example:

```
telemetry-2.0.0-7.0.3.I5.1.lib32_n9000.rpm
```

```
telemetry-2.0.0-7.0.3.I5.1.lib32_n3000.rpm
```

Installing Incremental Updates and Fixes

Copy the RPM to the device bootflash and use the following commands from the `bash` prompt:

```
feature bash
run bash sudo su
```

Then copy the RPM to the device bootflash. Use the following commands from the `bash` prompt:

```
dnf upgrade telemetry_new_version.rpm
```

The application is upgraded and the change appears when the application is started again.

Downgrading to a Previous Version

To downgrade the telemetry application to a previous version, use the following command from the `bash` prompt:

```
dnf downgrade telemetry
```

Verifying the Active Version

To verify the active version, run the following command from the switch `exec` prompt:

```
show install active
```



Note The `show install active` command will only show the active installed RPM after an upgrade has occurred. The default RPM that comes bundled with the NX-OS will not be displayed.

Guidelines and Limitations for Model Driven Telemetry

Telemetry has the following configuration guidelines and limitations:

- Cisco NX-OS releases that support the data management engine (DME) Native Model support Telemetry.
- Support is in place for the following:
 - DME data collection
 - NX-API data sources
 - Google protocol buffer (GPB) encoding over Google Remote Procedure Call (gRPC) transport
 - JSON encoding over HTTP
- The smallest sending interval (cadence) supported is five seconds for a depth of 0. The minimum cadence values for depth values greater than 0 depends on the size of the data being streamed out. Configuring any cadences below the minimum value may result in undesirable system behavior.
- Telemetry supports up to five remote management receivers (destinations). Configuring more than five remote receivers may result in undesirable system behavior.
- Telemetry can consume up to 20% of the CPU resource.

- To configure SSL certificate-based authentication and the encryption of streamed data, you can provide a self-signed SSL certificate with **certificate SSL cert path hostname "CN"** command.
- The following are guidelines for setting the telemetry cadence on YANG paths:
 - YANG streaming collection takes one thread. If multiple YANG paths exist in telemetry, each must run on a different cadence to prevent simultaneous scheduling and any resulting delay.
 - Before configuring the telemetry cadence for YANG paths, determine the total streaming time and configure the cadence to a value greater than the total streaming time. See *Configuring Cadence for YANG Path*.

Configuration Commands After Downgrading to an Older Release

After a downgrade to an older release, some configuration commands or command options can fail because the older release may not support them. When downgrading to an older release, unconfigure and reconfigure the telemetry feature after the new image comes up. This sequence avoids the failure of unsupported commands or command options.

The following example shows this procedure:

- Copy the telemetry configuration to a file:

```
switch# show running-config | section telemetry
feature telemetry
telemetry
  destination-group 100
    ip address 1.2.3.4 port 50004 protocol gRPC encoding GPB
    use-chunking size 4096
  sensor-group 100
    path sys/bgp/inst/dom-default depth 0
  subscription 600
    dst-grp 100
    snsr-grp 100 sample-interval 7000
switch# show running-config | section telemetry > telemetry_running_config
switch# show file bootflash:telemetry_running_config
feature telemetry
telemetry
  destination-group 100
    ip address 1.2.3.4 port 50004 protocol gRPC encoding GPB
    use-chunking size 4096
  sensor-group 100
    path sys/bgp/inst/dom-default depth 0
  subscription 600
    dst-grp 100
    snsr-grp 100 sample-interval 7000
switch#
```

- Execute the downgrade operation. When the image comes up and the switch is ready, copy the telemetry configurations back to the switch.

```
switch# copy telemetry_running_config running-config echo-commands
`switch# config terminal`
`switch(config)# feature telemetry`
`switch(config)# telemetry`
`switch(config-telemetry)# destination-group 100`
`switch(conf-tm-dest)# ip address 1.2.3.4 port 50004 protocol gRPC encoding GPB`
`switch(conf-tm-dest)# sensor-group 100`
`switch(conf-tm-sensor)# path sys/bgp/inst/dom-default depth 0`
```

```
`switch(conf-tm-sensor)# subscription 600`
`switch(conf-tm-sub)# dst-grp 100`
`switch(conf-tm-sub)# snsr-grp 100 sample-interval 7000`
`switch(conf-tm-sub)# end`
Copy complete, now saving to disk (please wait)...
Copy complete.
switch#
```

gRPC Error Behavior

The switch client disables the connection to the gRPC receiver if the gRPC receiver sends 20 errors. Unconfigure then reconfigure the receiver's IP address under the destination group to enable the gRPC receiver. Errors include:

- The gRPC client sends the wrong certificate for secure connections.
- The gRPC receiver takes too long to handle client messages and incurs a timeout. Avoid timeouts by processing messages using a separate message processing thread.

Telemetry Compression for gRPC Transport

Telemetry compression support is available for gRPC transport. You can use the **use-compression gzip** command to enable compression. (Disable compression with the **no use-compression gzip** command.)

The following example enables compression:

```
switch(config)# telemetry
switch(config-telemetry)# destination-profile
switch(config-tm-dest-profile)# use-compression gzip
```

The following example shows that compression is enabled:

```
switch(conf-tm-dest)# show telemetry transport 0 stats
```

```
Session Id:                0
Connection Stats
  Connection Count         0
  Last Connected:         Never
  Disconnect Count        0
  Last Disconnected:      Never
Transmission Stats
  Compression:            gzip
  Source Interface:       loopback1(1.1.3.4)
  Transmit Count:         0
  Last TX time:           None
  Min Tx Time:            0 ms
  Max Tx Time:            0 ms
  Avg Tx Time:            0 ms
  Cur Tx Time:            0 ms
```

```
switch2(config-if)# show telemetry transport 0 stats
```

```
Session Id: 0
Connection Stats
Connection Count 0
Last Connected: Never
Disconnect Count 0
Last Disconnected: Never
Transmission Stats
Compression: disabled
```



```

Source Interface: loopback1(1.1.3.4)
Transmit Count: 0
Last TX time: None
Min Tx Time: 0 ms
Max Tx Time: 0 ms
Avg Tx Time: 0 ms
Cur Tx Time: 0 ms
switch2(config-if)#

```

The following is an example of use-compression as a POST payload.

```

{
  "telemetryDestProfile": {
    "attributes": {
      "adminSt": "enabled"
    },
    "children": [
      {
        "telemetryDestOptCompression": {
          "attributes": {
            "name": "gzip"
          }
        }
      ]
    ]
  }
}

```

Support for gRPC Chunking

Starting with Release 9.2(1), support for gRPC chunking has been added. For streaming to occur successfully, you must enable chunking if gRPC has to send an amount of data greater than 12 MB to the receiver.

The gRPC user must do the gRPC chunking. The gRPC client side does the fragmentation, and the gRPC server side does the reassembly. Telemetry is still bound to memory and data can be dropped if the memory size is more than the allowed limit of 12 MB for telemetry. In order to support chunking, use the telemetry .proto file that is available at Cisco's GitLab, which has been updated for gRPC chunking, as described in *Telemetry Components and Process*.

The chunking size is from 64 through 4096 bytes.

Following shows a configuration example through the NX-API CLI:

```

feature telemetry
!
telemetry
  destination-group 1
    ip address 171.68.197.40 port 50051 protocol gRPC encoding GPB
    use-chunking size 4096
  destination-group 2
    ip address 10.155.0.15 port 50001 protocol gRPC encoding GPB
    use-chunking size 64
  sensor-group 1
    path sys/intf depth unbounded
  sensor-group 2
    path sys/intf depth unbounded
  subscription 1
    dst-grp 1
    snsr-grp 1 sample-interval 10000
  subscription 2
    dst-grp 2

```

```
snsr-grp 2 sample-interval 15000
```

Following shows a configuration example through the NX-API REST:

```
{
  "telemetryDestGrpOptChunking": {
    "attributes": {
      "chunkSize": "2048",
      "dn": "sys/tm/dest-1/chunking"
    }
  }
}
```

The following error message appears on systems that do not support gRPC chunking, such as the Cisco MDS series switches:

```
MDS-9706-86(conf-tm-dest)# use-chunking size 200
ERROR: Operation failed: [chunking support not available]
```

NX-API Sensor Path Limitations

NX-API can collect and stream switch information not yet in the DME using **show** commands. However, using the NX-API instead of streaming data from the DME has inherent scale limitations as outlined:

- The switch backend dynamically processes NX-API calls such as **show** commands,
- NX-API spawns several processes that can consume up to a maximum of 20% of the CPU.
- NX-API data translates from the CLI to XML to JSON.

The following is a suggested user flow to help limit excessive NX-API sensor path bandwidth consumption:

1. Check whether the **show** command has NX-API support. You can confirm whether NX-API supports the command from the VSH with the pipe option: `show <command> | json` or `show <command> | json pretty`.



Note Avoid commands that take the switch more than 30 seconds to return JSON output.

2. Refine the **show** command to include any filters or options.
 - Avoid enumerating the same command for individual outputs; for example, **show vlan id 100**, **show vlan id 101**, and so on. Instead, use the CLI range options; for example, **show vlan id 100-110,204**, whenever possible to improve performance.
 - If only the summary or counter is needed, then avoid dumping a whole show command output to limit the bandwidth and data storage that is required for data collection.
3. Configure telemetry with sensor groups that use NX-API as their data sources. Add the **show** commands as sensor paths
4. Configure telemetry with a cadence of five times the processing time of the respective **show** command to limit CPI usage.
5. Receive and process the streamed NX-API output as part of the existing DME collection.

Telemetry VRF Support

Telemetry VRF support allows you to specify a transport VRF, which means that the telemetry data stream can egress through front-panel ports and avoid possible competition between SSH or NGINX control sessions.

You can use the **use-vrf vrf-name** command to specify the transport VRF.

The following example specifies the transport VRF:

```
switch(config)# telemetry
switch(config-telemetry)# destination-profile
switch(config-tm-dest-profile)# use-vrf test_vrf
```

The following is an example of use-vrf as a POST payload:

```
{
  "telemetryDestProfile": {
    "attributes": {
      "adminSt": "enabled"
    },
    "children": [
      {
        "telemetryDestOptVrf": {
          "attributes": {
            "name": "default"
          }
        }
      ]
    ]
  }
}
```

Certificate Trustpoint Support

Beginning in NX-OS release 10.1(1), the **trustpoint** keyword is added in the existing global level command.

The following is the command syntax:

```
switch(config-telemetry)# certificate ?
trustpoint      specify trustpoint label
WORD            .pem certificate filename (Max Size 256)
switch(config-telemetry)# certificate trustpoint
WORD            trustpoint label name (Max Size 256)
switch(config-telemetry)# certificate trustpoint trustpoint1 ?
WORD            Hostname associated with certificate (Max Size 256)
switch(config-telemetry)#certificate trustpoint trustpoint1 foo.test.google.fr
```

Destination Hostname Support

Beginning in NX-OS release 10.1(1), the **host** keyword is added in destination-group command.

The following is the example for the destination hostname support:

```
switch(config-telemetry)# destination-group 1
switch(conf-tm-dest)# ?
certificate Specify certificate
host Specify destination host
ip Set destination IPv4 address
ipv6 Set destination IPv6 address
...
switch(conf-tm-dest)# host ?
```

```

A.B.C.D|A:B::C:D|WORD  IPv4 or IPv6 address or DNS name of destination
switch(conf-tm-dest)#

switch(conf-tm-dest)# host abc port 11111 ?
protocol  Set transport protocol
switch(conf-tm-dest)# host abc port 11111 protocol ?
HTTP
UDP
gRPC
switch(conf-tm-dest)# host abc port 11111 protocol gRPC ?
encoding  Set encoding format
switch(conf-tm-dest)# host abc port 11111 protocol gRPC encoding ?
Form-data  Set encoding to Form-data only
GPB        Set encoding to GPB only
GPB-compact Set encoding to Compact-GPB only
JSON       Set encoding to JSON
XML        Set encoding to XML
switch(conf-tm-dest)# host ip address 1.1.1.1 port 2222 protocol HTTP encoding JSON
<CR>

```

Support for Node ID

Beginning in NX-OS release 10.1(1), you can configure a custom Node ID string for a telemetry receiver through the **use-nodeid** command. By default, the host name is used, but support for a node ID enables you to set or change the identifier for the `node_id_str` of the telemetry receiver data.

You can assign the node ID through the telemetry destination profile, by using the **usenode-id** command. This command is optional.

The following example shows configuring the node ID.

```

switch-1(config)# telemetry
switch-1(config-telemetry)# destination-profile
switch-1(conf-tm-dest-profile)# use-nodeid test-srvr-10
switch-1(conf-tm-dest-profile)#

```

The following example shows a telemetry notification on the receiver after the node ID is configured.

```

Telemetry receiver:
=====
node_id_str: "test-srvr-10"
subscription_id_str: "1"
encoding_path: "sys/ch/psuslot-1/psu"
collection_id: 3896
msg_timestamp: 1559669946501

```

Use the **use-nodeid** sub-command under the **host** command. The destination level **use-nodeid** configuration preceeds the global level configuration.

The following example shows the command syntax:

```

switch(config-telemetry)# destination-group 1
switch(conf-tm-dest)# host 172.19.216.78 port 18112 protocol http enc json
switch(conf-tm-dest-host)# use-nodeid ?
WORD  Node ID (Max Size 128)
switch(conf-tm-dest-host)# use-nodeid session_1:18112

```

The following example shows the output from the Telemetry receiver:

```

>> Message size 923
Telemetry msg received @ 23:41:38 UTC
  Msg Size: 11
  node_id_str   : session_1:18112
  collection_id : 3118
  data_source   : DME

```

```
encoding_path : sys/ch/psuslot-1/psu
collection_start_time : 1598485314721
collection_end_time : 1598485314721
data :
```

Support for Streaming of YANG Models

Starting with Release 9.2(1), telemetry supports the YANG ("Yet Another Next Generation") data modeling language. Telemetry supports data streaming for both device YANG and OpenConfig YANG.

For more information on the YANG data modeling language, see *Infrastructure Overview* and *RESTConf Agent*.

Configuring Telemetry Using the CLI

Configuring Telemetry Using the NX-OS CLI

The following steps enable streaming telemetry and configuring the source and destination of the data stream. These steps also include optional steps to enable and configure SSL/TLS certificates and GPB encoding.

Before you begin

Your switch must be running Cisco NX-OS Release 7.3(0)I5(1) or a later release.

SUMMARY STEPS

1. (Optional) **openssl** *argument*
2. **configure terminal**
3. **feature telemetry**
4. **feature nxapi**
5. **nxapi use-vrf management**
6. **telemetry**
7. (Optional) **certificate** *certificate_path* *host_URL*
8. (Optional) Specify a transport VRF or enable telemetry compression for gRPC transport.
9. **sensor-group** *sgrp_id*
10. (Optional) **data-source** *data-source-type*
11. **path** *sensor_path* **depth** *0* [**filter-condition** *filter*] [**alias** *path_alias*]
12. **destination-group** *dgrp_id*
13. (Optional) **ip address** *ip_address* **port** *port* **protocol** *procedural-protocol* **encoding** *encoding-protocol*
14. (Optional) **ipv6 address** *ipv6_address* **port** *port* **protocol** *procedural-protocol* **encoding** *encoding-protocol*
15. **ip_version address** *ip_address* **port** *portnum*
16. (Optional) **use-chunking size** *chunking_size*
17. **subscription** *sub_id*
18. **snsr-grp** *sgrp_id* **sample-interval** *interval*
19. **dst-grp** *dgrp_id*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	(Optional) openssl <i>argument</i> Example: Generate an SSL/TLS certificate using a specific argument, such as the following: - To generate a private RSA key: openssl genrsa -cipher -out filename.key cipher-bit-length For example: <pre>switch# openssl genrsa -des3 -out server.key 2048</pre> - To write the RSA key: openssl rsa -in filename.key -out filename.key For example: <pre>switch# openssl rsa -in server.key -out server.key</pre> - To create a certificate that contains the public or private key: openssl req -encoding-standard -new -new filename.key -out filename.csr -subj '/CN=localhost' For example: <pre>switch# openssl req -sha256 -new -key server.key -out server.csr -subj '/CN=localhost'</pre> - To create a public key: openssl x509 -req -encoding-standard -days timeframe -in filename.csr -signkey filename.key -out filename.csr For example: <pre>switch# openssl x509 -req -sha256 -days 365 -in server.csr -signkey server.key -out server.crt</pre>	Create an SSL or TLS certificate on the server that receives the data, where the <i>private.key</i> file is the private key and the <i>public.crt</i> is the public key.
Step 2	configure terminal Example: <pre>switch# configure terminal switch(config)#</pre>	Enter the global configuration mode.
Step 3	feature telemetry	Enable the streaming telemetry feature.
Step 4	feature nxapi	Enable NX-API.

	Command or Action	Purpose
Step 5	nxapi use-vrf management	Enable the VRF management to be used for NX-API communication.
Step 6	telemetry Example: switch(config)# telemetry switch(config-telemetry)#	Enter configuration mode for streaming telemetry.
Step 7	(Optional) certificate <i>certificate_path</i> <i>host_URL</i> Example: switch(config-telemetry)# certificate /bootflash/server.key localhost	Use an existing SSL/TLS certificate.
Step 8	(Optional) Specify a transport VRF or enable telemetry compression for gRPC transport. Example: switch(config-telemetry)# destination-profile switch(conf-tm-dest-profile)# use-vrf default switch(conf-tm-dest-profile)# use-compression gzip switch(conf-tm-dest-profile)# use-retry size 10 switch(conf-tm-dest-profile)# source-interface loopback1	• Enter the destination-profile command to specify the default destination profile. • Enter any of the following commands: • use-vrf <i>vrf</i> to specify the destination VRF. • use-compression gzip to specify the destination compression method. • use-retry size <i>size</i> to specify the send retry details, with a retry buffer size between 10–1500 megabytes. • source-interface <i>interface-name</i> to stream data from the configured interface to a destination with the source IP address. Note After configuring the use-vrf command, you must configure a new destination IP address within the new VRF. However, you may re-use the same destination IP address by unconfiguring and reconfiguring the destination. This action ensures that the telemetry data streams to the same destination IP address in the new VRF.
Step 9	sensor-group <i>sgrp_id</i> Example: switch(config-telemetry)# sensor-group 100 switch(conf-tm-sensor)#	Create a sensor group with ID <i>sgrp_id</i> and enter sensor group configuration mode. Currently only numeric ID values are supported. The sensor group defines nodes that will be monitored for telemetry reporting.
Step 10	(Optional) data-source <i>data-source-type</i> Example: switch(config-telemetry)# data-source NX-API	Select a data source. Select from either YANG, DME or NX-API as the data source. Note DME is the default data source.

	Command or Action	Purpose
Step 11	path <i>sensor_path</i> depth 0 [filter-condition <i>filter</i>] [alias <i>path_alias</i>] Example: - The following command is applicable for DME, not for NX-API or YANG: <pre>switch(conf-tm-sensor)# path sys/bd/bd-[vlan-100] depth 0 filter-condition eq(l2Bd.operSt, "down")</pre> Use the following syntax for state-based filtering to trigger only when operSt changes from up to down, with no notifications of when the MO changes. <pre>switch(conf-tm-sensor)# path sys/bd/bd-[vlan-100] depth 0 filter-condition and(updated(l2Bd.operSt),eq(l2Bd.operSt,"down"))</pre> Use the following syntax to distinguish the path on the UTR side. <pre>switch(conf-tm-sensor)# path sys/ch/ftslot-1/ft alias ft_1</pre> - The following command is applicable for NX-API, not for DME or YANG: <pre>switch(conf-tm-sensor)# path "show interface" depth 0</pre> - The following command is applicable for device YANG: <pre>switch(conf-tm-sensor)# path Cisco-NX-OS-device:System/bgp-items/inst-items</pre> - The following commands are applicable for OpenConfig YANG: <pre>switch(conf-tm-sensor)# path openconfig-bgp:bgp switch(conf-tm-sensor)# path Cisco-NX-OS-device:System/bgp-items/inst-items alias bgp_alias</pre> - The following command is applicable for NX-API: <pre>switch(conf-tm-sensor)# path "show interface" depth 0 alias sh_int_alias</pre> - The following command is applicable for OpenConfig: <pre>switch(conf-tm-sensor)# path openconfig-bgp:bgp alias oc_bgp_alias</pre>	Add a sensor path to the sensor group. - Beginning with the Cisco NX-OS 9.3(5) release, the alias keyword is introduced. - The depth setting specifies the retrieval level for the sensor path. Depth settings of 0 - 32, unbounded are supported. Note depth 0 is the default depth. NX-API-based sensor paths can only use depth 0. If a path is subscribed for the event collection, the depth only supports 0 and unbounded. Other values would be treated as 0. - The optional filter-condition parameter can be specified to create a specific filter for event-based subscriptions. For state-based filtering, the filter returns both when a state has changed and when an event has occurred during the specified state. That is, a filter condition for the DN sys/bd/bd-[vlan] of eq(l2Bd.operSt, "down") triggers when the operSt changes, and when the DN's property changes while the operSt remains down, such as a no shutdown command is issued while the VLAN is operationally down. Note query-condition parameter — For DME, based on the DN, the query-condition parameter can be specified to fetch MOTL and ephemeral data with the following syntax: query-condition "rsp-foreign-subtree=applied-config"; query-condition "rsp-foreign-subtree=ephemeral". - For the YANG model, the sensor path format is as follows: <i>module_name: YANG_path</i>, where <i>module_name</i> is the name of the YANG model file. For example: - For device YANG: Cisco-NX-OS-device:System/bgp-items/inst-items - For OpenConfig YANG: openconfig-bgp:bgp Note The depth, filter-condition, and query-condition parameters are not supported for YANG currently.

	Command or Action	Purpose
		For the openconfig YANG models, go to https://github.com/YangModels/yang/tree/master/vendor/cisco/nx and navigate to the appropriate folder for the latest release. Instead of installing a specific model, you can install the openconfig-all RPM which has all the OpenConfig models. See <i>Adding Patch RPMs from Bash</i> topic for more information on installing patch RPMs. For example: <pre>install add mtx-openconfig-bgp-1.0.0.0-7.0.3.IHD8.1.lib32.n9000.rpm activate</pre>
Step 12	destination-group <i>dgrp_id</i> Example: <pre>switch(conf-tm-sensor)# destination-group 100 switch(conf-tm-dest)#</pre>	Create a destination group and enter destination group configuration mode. Currently <i>dgrp_id</i> only supports numeric ID values.
Step 13	(Optional) ip address <i>ip_address</i> port <i>port</i> protocol <i>procedural-protocol</i> encoding <i>encoding-protocol</i> Example: <pre>switch(conf-tm-sensor)# ip address 171.70.55.69 port 50001 protocol gRPC encoding GPB switch(conf-tm-sensor)# ip address 171.70.55.69 port 50007 protocol HTTP encoding JSON switch(conf-tm-sensor)# ip address 171.70.55.69 port 50009 protocol UDP encoding JSON</pre>	Specify an IPv4 IP address and port to receive encoded telemetry data. Note gRPC is the default transport protocol. GPB is the default encoding.
Step 14	(Optional) ipv6 address <i>ipv6_address</i> port <i>port</i> protocol <i>procedural-protocol</i> encoding <i>encoding-protocol</i> Example: <pre>switch(conf-tm-sensor)# ipv6 address 10:10::1 port 8000 protocol gRPC encoding GPB switch(conf-tm-sensor)# ipv6 address 10:10::1 port 8001 protocol HTTP encoding JSON switch(conf-tm-sensor)# ipv6 address 10:10::1 port 8002 protocol UDP encoding JSON</pre>	Specify an IPv6 IP address and port to receive encoded telemetry data. Note gRPC is the default transport protocol. GPB is the default encoding.
Step 15	ip_version address <i>ip_address</i> port <i>portnum</i> Example: - For IPv4: <pre>switch(conf-tm-dest)# ip address 1.2.3.4 port 50003</pre> - For IPv6:	Create a destination profile for the outgoing data, where <i>ip_version</i> is either ip (for IPv4) or ipv6 (for IPv6). When the destination group is linked to a subscription, telemetry data is sent to the IP address and port that is specified by this profile.

	Command or Action	Purpose
	<code>switch(conf-tm-dest) # ipv6 address 10:10::1 port 8000</code>	
Step 16	(Optional) <code>use-chunking size chunking_size</code> Example: <code>switch(conf-tm-dest) # use-chunking size 64</code>	Enable gRPC chunking and set the chunking size, between 64-4096 bytes. See the section "Support for gRPC Chunking" for more information.
Step 17	<code>subscription sub_id</code> Example: <code>switch(conf-tm-dest) # subscription 100 switch(conf-tm-sub) #</code>	Create a subscription node with ID and enter the subscription configuration mode. Currently <i>sub_id</i> only supports numeric ID values. Note When subscribing to a DN, check whether the DN is supported by DME using REST to ensure that events will stream.
Step 18	<code>snsr-grp sgrp_id sample-interval interval</code> Example: <code>switch(conf-tm-sub) # snsr-grp 100 sample-interval 15000</code>	Link the sensor group with ID <i>sgrp_id</i> to this subscription and set the data sampling interval in milliseconds. An interval value of 0 creates an event-based subscription, in which telemetry data is sent only upon changes under the specified MO. An interval value greater than 0 creates a frequency-based subscription, in which telemetry data is sent periodically at the specified interval. For example, an interval value of 15000 results in the sending of telemetry data every 15 seconds.
Step 19	<code>dst-grp dgrp_id</code> Example: <code>switch(conf-tm-sub) # dst-grp 100</code>	Link the destination group with ID <i>dgrp_id</i> to this subscription.

Configuring Cadence for YANG Paths

The cadence for YANG paths must be greater than the total streaming time. If the total streaming time and cadence are incorrectly configured, gathering telemetry data can take longer than the streaming interval. In this situation, you can see:

- Queues that incrementally fill because telemetry data is accumulating faster than it is streaming to the receiver.
- Stale telemetry data which is not from the current interval.

Configure the cadence to a value greater than the total streaming time.

SUMMARY STEPS

1. `show telemetry control database sensor-groups`
2. `sensor group number`

	Command or Action	Purpose
	<pre>subscription 1 dst-grp 1 snsr-grp 1 sample-interval 5000 snsr-grp 2 sample-interval 5000</pre>	
Step 2	sensor group number Example: <pre>switch-1(config-telemetry)# sensor group 1</pre>	If the total streaming time is not less than the cadence, enter the sensor group for which you want to set the interval.
Step 3	subscription number Example: <pre>switch-1(conf-tm-sensor)# subscription 100</pre>	Edit the subscription for the sensor group.
Step 4	snsr-grp number sample-interval milliseconds Example: <pre>switch-1(conf-tm-sub)# snsr-grp number sample-interval 5000</pre>	For the appropriate sensor group, set the sample interval to a value greater than the total streaming time. In this example, the sample interval is set to 5.000 seconds, which is valid because it is larger than the total streaming time of 2.664 seconds.
Step 5	show system resources Example: <pre>switch-1# show system resources Load average: 1 minute: 0.38 5 minutes: 0.43 15 minutes: 0.43 Processes: 555 total, 3 running CPU states : 24.17% user, 4.32% kernel, 71.50% idle CPU0 states: 0.00% user, 2.12% kernel, 97.87% idle CPU1 states: 86.00% user, 11.00% kernel, 3.00% idle CPU2 states: 8.08% user, 3.03% kernel, 88.88% idle CPU3 states: 0.00% user, 1.02% kernel, 98.97% idle Memory usage: 16400084K total, 5861652K used, 10538432K free Current memory status: OK</pre>	Check the CPU usage. If the CPU user state shows high usage, as shown in this example, your cadence and streaming value are not configured correctly. Repeat this procedure to properly configure the cadence.

Configuration Examples for Telemetry Using the CLI

The following steps describe how to configure a single telemetry DME stream with a ten second cadence with GPB encoding.

```
switch# configure terminal
switch(config)# feature telemetry
switch(config)# telemetry
switch(config-telemetry)# destination-group 1
switch(config-tm-dest)# ip address 171.70.59.62 port 50051 protocol gRPC encoding GPB
switch(config-tm-dest)# exit
switch(config-telemetry)# sensor group sg1
switch(config-tm-sensor)# data-source DME
```

```
switch(config-tm-dest)# path interface depth unbounded query-condition keep-data-type
switch(config-tm-dest)# subscription 1
switch(config-tm-dest)# dst-grp 1
switch(config-tm-dest)# snsr grp 1 sample interval 10000
```

This example creates a subscription that streams data for the `sys/bgp` root MO every 5 seconds to the destination IP 1.2.3.4 port 50003.

```
switch(config)# telemetry
switch(config-telemetry)# sensor-group 100
switch(conf-tm-sensor)# path sys/bgp depth 0
switch(conf-tm-sensor)# destination-group 100
switch(conf-tm-dest)# ip address 1.2.3.4 port 50003
switch(conf-tm-dest)# subscription 100
switch(conf-tm-sub)# snsr-grp 100 sample-interval 5000
switch(conf-tm-sub)# dst-grp 100
```

This example creates a subscription that streams data for `sys/intf` every 5 seconds to destination IP 1.2.3.4 port 50003, and encrypts the stream using GPB encoding that is verified using the `test.pem`.

```
switch(config)# telemetry
switch(config-telemetry)# certificate /bootflash/test.pem foo.test.google.fr
switch(conf-tm-telemetry)# destination-group 100
switch(conf-tm-dest)# ip address 1.2.3.4 port 50003 protocol gRPC encoding GPB
switch(config-dest)# sensor-group 100
switch(conf-tm-sensor)# path sys/bgp depth 0
switch(conf-tm-sensor)# subscription 100
switch(conf-tm-sub)# snsr-grp 100 sample-interval 5000
switch(conf-tm-sub)# dst-grp 100
```

This example creates a subscription that streams data for `sys/cdp` every 15 seconds to destination IP 1.2.3.4 port 50004.

```
switch(config)# telemetry
switch(config-telemetry)# sensor-group 100
switch(conf-tm-sensor)# path sys/cdp depth 0
switch(conf-tm-sensor)# destination-group 100
switch(conf-tm-dest)# ip address 1.2.3.4 port 50004
switch(conf-tm-dest)# subscription 100
switch(conf-tm-sub)# snsr-grp 100 sample-interval 15000
switch(conf-tm-sub)# dst-grp 100
```

This example creates a cadence-based collection of `show` command data every 750 seconds.

```
switch(config)# telemetry
switch(config-telemetry)# destination-group 1
switch(conf-tm-dest)# ip address 172.27.247.72 port 60001 protocol gRPC encoding GPB
switch(conf-tm-dest)# sensor-group 1
switch(conf-tm-sensor)# data-source NX-API
switch(conf-tm-sensor)# path "show system resources" depth 0
switch(conf-tm-sensor)# path "show version" depth 0
switch(conf-tm-sensor)# path "show environment power" depth 0
switch(conf-tm-sensor)# path "show environment fan" depth 0
switch(conf-tm-sensor)# path "show environment temperature" depth 0
switch(conf-tm-sensor)# path "show process cpu" depth 0
switch(conf-tm-sensor)# path "show nve peers" depth 0
switch(conf-tm-sensor)# path "show nve vni" depth 0
switch(conf-tm-sensor)# path "show nve vni 4002 counters" depth 0
```

```

switch(conf-tm-sensor)# path "show int nve 1 counters" depth 0
switch(conf-tm-sensor)# path "show policy-map vlan" depth 0
switch(conf-tm-sensor)# path "show ip access-list test" depth 0
switch(conf-tm-sensor)# path "show system internal access-list resource utilization" depth 0
switch(conf-tm-sensor)# subscription 1
switch(conf-tm-sub)# dst-grp 1
switch(conf-tm-dest)# snsr-grp 1 sample-interval 750000

```

This example creates an event-based subscription for `sys/fm`. Data is streamed to the destination only if there is a change under the `sys/fm` MO.

```

switch(config)# telemetry
switch(config-telemetry)# sensor-group 100
switch(conf-tm-sensor)# path sys/fm depth 0
switch(conf-tm-sensor)# destination-group 100
switch(conf-tm-dest)# ip address 1.2.3.4 port 50005
switch(conf-tm-dest)# subscription 100
switch(conf-tm-sub)# snsr-grp 100 sample-interval 0
switch(conf-tm-sub)# dst-grp 100

```

During operation, you can change a sensor group from frequency-based to event-based, and change event-based to frequency-based by changing the sample-interval. This example changes the sensor-group from the previous example to frequency-based. After the following commands, the telemetry application will begin streaming the `sys/fm` data to the destination every 7 seconds.

```

switch(config)# telemetry
switch(config-telemetry)# subscription 100
switch(conf-tm-sub)# snsr-grp 100 sample-interval 7000

```

Multiple sensor groups and destinations can be linked to a single subscription. The subscription in this example streams the data for Ethernet port 1/1 to four different destinations every 10 seconds.

```

switch(config)# telemetry
switch(config-telemetry)# sensor-group 100
switch(conf-tm-sensor)# path sys/intf/phys-[eth1/1] depth 0
switch(conf-tm-sensor)# destination-group 100
switch(conf-tm-dest)# ip address 1.2.3.4 port 50004
switch(conf-tm-dest)# ip address 1.2.3.4 port 50005
switch(conf-tm-sensor)# destination-group 200
switch(conf-tm-dest)# ip address 5.6.7.8 port 50001 protocol HTTP encoding JSON
switch(conf-tm-dest)# ip address 1.4.8.2 port 60003
switch(conf-tm-dest)# subscription 100
switch(conf-tm-sub)# snsr-grp 100 sample-interval 10000
switch(conf-tm-sub)# dst-grp 100
switch(conf-tm-sub)# dst-grp 200

```

A sensor group can contain multiple paths, a destination group can contain multiple destination profiles, and a subscription can be linked to multiple sensor groups and destination groups, as shown in this example.

```

switch(config)# telemetry
switch(config-telemetry)# sensor-group 100
switch(conf-tm-sensor)# path sys/intf/phys-[eth1/1] depth 0
switch(conf-tm-sensor)# path sys/epId-1 depth 0
switch(conf-tm-sensor)# path sys/bgp/inst/dom-default depth 0

```

```

switch(config-telemetry)# sensor-group 200
switch(conf-tm-sensor)# path sys/cdp depth 0
switch(conf-tm-sensor)# path sys/ipv4 depth 0

switch(config-telemetry)# sensor-group 300
switch(conf-tm-sensor)# path sys/fm depth 0
switch(conf-tm-sensor)# path sys/bgp depth 0

switch(conf-tm-sensor)# destination-group 100
switch(conf-tm-dest)# ip address 1.2.3.4 port 50004
switch(conf-tm-dest)# ip address 4.3.2.5 port 50005

switch(conf-tm-dest)# destination-group 200
switch(conf-tm-dest)# ip address 5.6.7.8 port 50001

switch(conf-tm-dest)# destination-group 300
switch(conf-tm-dest)# ip address 1.2.3.4 port 60003

switch(conf-tm-dest)# subscription 600
switch(conf-tm-sub)# snsrg 100 sample-interval 7000
switch(conf-tm-sub)# snsrg 200 sample-interval 20000
switch(conf-tm-sub)# dst-grp 100
switch(conf-tm-sub)# dst-grp 200

switch(conf-tm-dest)# subscription 900
switch(conf-tm-sub)# snsrg 200 sample-interval 7000
switch(conf-tm-sub)# snsrg 300 sample-interval 0
switch(conf-tm-sub)# dst-grp 100
switch(conf-tm-sub)# dst-grp 300

```

You can verify the telemetry configuration using the **show running-config telemetry** command, as shown in this example.

```

switch(config)# telemetry
switch(config-telemetry)# destination-group 100
switch(conf-tm-dest)# ip address 1.2.3.4 port 50003
switch(conf-tm-dest)# ip address 1.2.3.4 port 50004
switch(conf-tm-dest)# end
switch# show run telemetry

!Command: show running-config telemetry
!Time: Thu Oct 13 21:10:12 2016

version 7.0(3)I5(1)
feature telemetry

telemetry
destination-group 100
ip address 1.2.3.4 port 50003 protocol gRPC encoding GPB
ip address 1.2.3.4 port 50004 protocol gRPC encoding GPB

```

You can specify transport VRF and telemetry data compression for gRPC using the **use-vrf** and **use-compression gzip** commands, as shown in this example.

```

switch(config)# telemetry
switch(config-telemetry)# destination-profile
switch(conf-tm-dest-profile)# use-vrf default
switch(conf-tm-dest-profile)# use-compression gzip
switch(conf-tm-dest-profile)# sensor-group 1

```

```

switch(conf-tm-sensor)# path sys/bgp depth unbounded
switch(conf-tm-sensor)# destination-group 1
switch(conf-tm-dest)# ip address 1.2.3.4 port 50004
switch(conf-tm-dest)# subscription 1
switch(conf-tm-sub)# dst-grp 1
switch(conf-tm-sub)# snsr-grp 1 sample-interval 10000

```

Displaying Telemetry Configuration and Statistics

Use the following NX-OS CLI **show** commands to display telemetry configuration, statistics, errors, and session information.

show telemetry yang direct-path cisco-nxos-device

This command displays YANG paths that are directly encoded to perform better than other paths.

```

switch# show telemetry yang direct-path cisco-nxos-device
1) Cisco-NX-OS-device:System/lldp-items
2) Cisco-NX-OS-device:System/acl-items
3) Cisco-NX-OS-device:System/mac-items
4) Cisco-NX-OS-device:System/intf-items
5) Cisco-NX-OS-device:System/procsys-items/sysload-items
6) Cisco-NX-OS-device:System/ospf-items
7) Cisco-NX-OS-device:System/procsys-items
8) Cisco-NX-OS-device:System/ipqos-items/queuing-items/policy-items/out-items
9) Cisco-NX-OS-device:System/mac-items/static-items
10) Cisco-NX-OS-device:System/ch-items
11) Cisco-NX-OS-device:System/cdp-items
12) Cisco-NX-OS-device:System/bd-items
13) Cisco-NX-OS-device:System/eps-items
14) Cisco-NX-OS-device:System/ipv6-items

```

show telemetry control database

This command displays the internal databases that reflect the configuration of telemetry.

```

switch# show telemetry control database ?
<CR>
> Redirect it to a file
>> Redirect it to a file in append mode
destination-groups Show destination-groups
destinations Show destinations
sensor-groups Show sensor-groups
sensor-paths Show sensor-paths
subscriptions Show subscriptions
| Pipe command output to filter

```

```
switch# show telemetry control database
```

```
Subscription Database size = 1
```

```

-----
Subscription ID      Data Collector Type
-----
100                  DME NX-API

```

```
Sensor Group Database size = 1
```

```

-----
Sensor Group ID  Sensor Group type  Sampling interval(ms)  Linked subscriptions
-----

```



```

100          Timer          10000(Running)          1

Sensor Path Database size = 1

-----
Subscribed Query Filter  Linked Groups  Sec Groups  Retrieve level  Sensor Path
-----
No                      1              0          Full          sys/fm

Destination group Database size = 2

-----
Destination Group ID  Refcount
-----
100                  1

Destination Database size = 2

-----
Dst IP Addr      Dst Port  Encoding  Transport  Count
-----
192.168.20.111   12345      JSON     HTTP      1
192.168.20.123  50001      GPB       gRPC      1

```

show telemetry control database sensor-paths

This command displays sensor path details for telemetry configuration, including counters for encoding, collection, transport, and streaming.

```
switch-1(conf-tm-sub)# show telemetry control database sensor-paths
```

```

Sensor Path Database size = 4

-----
Row ID      Subscribed Linked Groups  Sec Groups  Retrieve level  Path(GroupId) : Query :
Filter
-----
1           No          1              0          Full          sys/cdp(1) : NA : NA

GPB Encoded Data size in bytes (Cur/Min/Max): 0/0/0
JSON Encoded Data size in bytes (Cur/Min/Max): 65785/65785/65785
Collection Time in ms (Cur/Min/Max): 10/10/55
Encoding Time in ms (Cur/Min/Max): 8/8/9
Transport Time in ms (Cur/Min/Max): 0/0/0
Streaming Time in ms (Cur/Min/Max): 18/18/65

2           No          1              0          Self          show module(2) : NA : NA

GPB Encoded Data size in bytes (Cur/Min/Max): 0/0/0
JSON Encoded Data size in bytes (Cur/Min/Max): 1107/1106/1107
Collection Time in ms (Cur/Min/Max): 603/603/802
Encoding Time in ms (Cur/Min/Max): 0/0/0
Transport Time in ms (Cur/Min/Max): 0/0/1
Streaming Time in ms (Cur/Min/Max): 605/605/803

3           No          1              0          Full          sys/bgp(1) : NA : NA

GPB Encoded Data size in bytes (Cur/Min/Max): 0/0/0
JSON Encoded Data size in bytes (Cur/Min/Max): 0/0/0
Collection Time in ms (Cur/Min/Max): 0/0/44
Encoding Time in ms (Cur/Min/Max): 0/0/0
Transport Time in ms (Cur/Min/Max): 0/0/0
Streaming Time in ms (Cur/Min/Max): 1/1/44

4           No          1              0          Self          show version(2) : NA : NA

GPB Encoded Data size in bytes (Cur/Min/Max): 0/0/0
JSON Encoded Data size in bytes (Cur/Min/Max): 2442/2441/2442

```

```
Collection Time in ms (Cur/Min/Max): 1703/1703/1903
Encoding Time in ms (Cur/Min/Max): 0/0/0
Transport Time in ms (Cur/Min/Max): 0/0/0
Streaming Time in ms (Cur/Min/Max): 1703/1703/1904
```

```
switch-1(conf-tm-sub)#
```

show telemetry control stats

This command displays the statistics about the internal databases about configuration of telemetry.

```
switch# show telemetry control stats
show telemetry control stats entered
```

Error Description	Error Count
Chunk allocation failures	0
Sensor path Database chunk creation failures	0
Sensor Group Database chunk creation failures	0
Destination Database chunk creation failures	0
Destination Group Database chunk creation failures	0
Subscription Database chunk creation failures	0
Sensor path Database creation failures	0
Sensor Group Database creation failures	0
Destination Database creation failures	0
Destination Group Database creation failures	0
Subscription Database creation failures	0
Sensor path Database insert failures	0
Sensor Group Database insert failures	0
Destination Database insert failures	0
Destination Group Database insert failures	0
Subscription insert to Subscription Database failures	0
Sensor path Database delete failures	0
Sensor Group Database delete failures	0
Destination Database delete failures	0
Destination Group Database delete failures	0
Delete Subscription from Subscription Database failures	0
Sensor path delete in use	0
Sensor Group delete in use	0
Destination delete in use	0
Destination Group delete in use	0
Delete destination(in use) failure count	0
Failed to get encode callback	0
Sensor path Sensor Group list creation failures	0
Sensor path prop list creation failures	0
Sensor path sec Sensor path list creation failures	0
Sensor path sec Sensor Group list creation failures	0
Sensor Group Sensor path list creation failures	0
Sensor Group Sensor subs list creation failures	0
Destination Group subs list creation failures	0
Destination Group Destinations list creation failures	0
Destination Destination Groups list creation failures	0
Subscription Sensor Group list creation failures	0
Subscription Destination Groups list creation failures	0
Sensor Group Sensor path list delete failures	0
Sensor Group Subscriptions list delete failures	0
Destination Group Subscriptions list delete failures	0
Destination Group Destinations list delete failures	0
Subscription Sensor Groups list delete failures	0
Subscription Destination Groups list delete failures	0
Destination Destination Groups list delete failures	0
Failed to delete Destination from Destination Group	0

```

Failed to delete Destination Group from Subscription      0
Failed to delete Sensor Group from Subscription          0
Failed to delete Sensor path from Sensor Group           0
Failed to get encode callback                            0
Failed to get transport callback                         0
switch# Destination Database size = 1

```

```

-----
Dst IP Addr      Dst Port  Encoding  Transport  Count
-----
192.168.20.123  50001      GPB       gRPC       1

```

show telemetry data collector brief

This command displays the brief statistics about the data collection.

```
switch# show telemetry data collector brief
```

```

-----
Collector Type      Successful Collections  Failed Collections
-----
DME                  143                      0

```

show telemetry data collector details

This command displays detailed statistics about the data collection which includes breakdown of all sensor paths.

```
switch# show telemetry data collector details
```

```

-----
Succ Collections    Failed Collections    Sensor Path
-----
150                 0                     sys/fm

```

show telemetry event collector errors

This command displays the errors statistic about the event collection.

```
switch# show telemetry event collector errors
```

```

-----
Error Description                                     Error Count
-----
APIC-Cookie Generation Failures                       - 0
Authentication Failures                               - 0
Authentication Refresh Failures                       - 0
Authentication Refresh Timer Start Failures           - 0
Connection Timer Start Failures                       - 0
Connection Attempts                                   - 3
Dme Event Subscription Init Failures                   - 0
Event Data Enqueue Failures                           - 0
Event Subscription Failures                           - 0
Event Subscription Refresh Failures                   - 0
Pending Subscription List Create Failures              - 0
Subscription Hash Table Create Failures                - 0
Subscription Hash Table Destroy Failures               - 0

```

```
Subscription Hash Table Insert Failures      - 0
Subscription Hash Table Remove Failures     - 0
Subscription Refresh Timer Start Failures    - 0
Websocket Connect Failures                  - 0
```

show telemetry event collector stats

This command displays the statistics about the event collection which includes breakdown of all sensor paths.

```
switch# show telemetry event collector stats
```

```
-----
Collection Count  Latest Collection Time  Sensor Path
-----
```

show telemetry control pipeline stats

This command displays the statistics for the telemetry pipeline.

```
switch# show telemetry pipeline stats
```

Main Statistics:

Timers:

Errors:

```
Start Fail      =      0
```

Data Collector:

Errors:

```
Node Create Fail =      0
```

Event Collector:

Errors:

```
Node Create Fail =      0   Node Add Fail      =      0
Invalid Data     =      0
```

Memory:

```
Allowed Memory Limit      = 1181116006 bytes
Occupied Memory           = 93265920 bytes
```

Queue Statistics:

Request Queue:

High Priority Queue:

Info:

```
Actual Size      =      50   Current Size      =      0
Max Size         =      0   Full Count       =      0
```

Errors:

```
Enqueue Error    =      0   Dequeue Error    =      0
```

Low Priority Queue:

Info:

```
Actual Size      =      50   Current Size      =      0
Max Size         =      0   Full Count       =      0
```

Errors:

```
Enqueue Error    =      0   Dequeue Error    =      0
```

Data Queue:

High Priority Queue:

Info:

```
Actual Size      =      50   Current Size      =      0
```

```

Max Size          =      0    Full Count          =      0

Errors:
  Enqueue Error    =      0    Dequeue Error       =      0

Low Priority Queue:
Info:
  Actual Size      =     50    Current Size        =      0
  Max Size         =      0    Full Count          =      0

Errors:
  Enqueue Error    =      0    Dequeue Error       =      0

```

show telemetry transport

This command displays all configured transport sessions.

```
switch# show telemetry transport
```

Session Id	IP Address	Port	Encoding	Transport	Status
0	192.168.20.123	50001	GPB	gRPC	Connected

show telemetry transport <session-id>

This command displays detailed session information for a specific transport session.

```
switch# show telemetry transport 0
```

```

Session Id:      0
IP Address:Port  192.168.20.123:50001
Encoding:        GPB
Transport:       gRPC
Status:          Disconnected
Last Connected:  Fri Sep 02 11:45:57.505 UTC
Last Disconnected: Never
Tx Error Count:  224
Last Tx Error:   Fri Sep 02 12:23:49.555 UTC

```

```
switch# show telemetry transport 1
```

```

Session Id:      1
IP Address:Port  10.30.218.56:51235
Transport:       HTTP
Status:          Disconnected
Last Connected:  Never
Last Disconnected: Never
Tx Error Count:  3
Last Tx Error:   Wed Apr 19 15:56:51.617 PDT

```

The following example shows output from an IPv6 entry.

```
switch# show telemetry transport 0
```

```

Session Id: 0
IP Address:Port [10:10::1]:8000
Transport: GRPC
Status: Idle
Last Connected: Never
Last Disconnected: Never
Tx Error Count: 0

```

```

Last Tx Error: None
Event Retry Queue Bytes: 0
Event Retry Queue Size: 0
Timer Retry Queue Bytes: 0
Timer Retry Queue Size: 0
Sent Retry Messages: 0
Dropped Retry Messages: 0

```

show telemetry transport <session-id> stats

This command displays details of a specific transport session.

```
switch# show telemetry transport 0 stats
```

```

Session Id:          0
IP Address:Port      192.168.20.123:50001
Encoding:            GPB
Transport:           GRPC
Status:              Connected
Last Connected:      Mon May 01 11:29:46.912 PST
Last Disconnected:   Never
Tx Error Count:      0
Last Tx Error:       None

```

show telemetry transport <session-id> stats

This command displays details of a specific transport session.

```

Session Id:          0
Transmission Stats
  Compression:        disabled
  Source Interface:    not set()
  Transmit Count:      319297
  Last TX time:        Fri Aug 02 03:51:15.287 UTC
  Min Tx Time:         1 ms
  Max Tx Time:         3117 ms
  Avg Tx Time:         3 ms
  Cur Tx Time:         1 ms

```

show telemetry transport <session-id> errors

This command displays detailed error statistics for a specific transport session.

```

switch# show telemetry transport 0 errors
Session Id:          0
Connection Errors
Connection Error Count: 0
Transmission Errors
Tx Error Count:      30
Last Tx Error:        Thu Aug 01 04:39:47.083 UTC
Last Tx Return Code:   No error

```

show telemetry control databases sensor-paths

These following configuration steps result in the **show telemetry control databases sensor-paths** command output below.

```

feature telemetry

telemetry

```

```

destination-group 1
  ip address 172.25.238.13 port 50600 protocol gRPC encoding GPB
sensor-group 1
  path sys/cdp depth unbounded
  path sys/intf depth unbounded
  path sys/mac depth 0
subscription 1
  dst-grp 1
  snsr-grp 1 sample-interval 1000

```

Command output.

```
switch# show telemetry control databases sensor-paths
```

```

Sensor Path Database size = 3
-----
-----
Row ID      Subscribed Linked Groups  Sec Groups  Retrieve level  Path(GroupId) :
Query : Filter
-----
-----
1           No                1            0            Full          sys/cdp(1) : NA
: NA
GPB Encoded Data size in bytes (Cur/Min/Max): 30489/30489/30489
JSON Encoded Data size in bytes (Cur/Min/Max): 0/0/0
CGPB Encoded Data size in bytes (Cur/Min/Max): 0/0/0
Collection Time in ms (Cur/Min/Max): 6/5/54
Encoding Time in ms (Cur/Min/Max): 5/5/6
Transport Time in ms (Cur/Min/Max): 1027/55/1045
Streaming Time in ms (Cur/Min/Max): 48402/5/48402

2           No                1            0            Full          sys/intf(1) : N
A : NA
GPB Encoded Data size in bytes (Cur/Min/Max): 539466/539466/539466
JSON Encoded Data size in bytes (Cur/Min/Max): 0/0/0
CGPB Encoded Data size in bytes (Cur/Min/Max): 0/0/0
Collection Time in ms (Cur/Min/Max): 66/64/114
Encoding Time in ms (Cur/Min/Max): 91/90/92
Transport Time in ms (Cur/Min/Max): 4065/4014/5334
Streaming Time in ms (Cur/Min/Max): 48365/64/48365

3           No                1            0            Self          sys/mac(1) : NA
: NA
GPB Encoded Data size in bytes (Cur/Min/Max): 247/247/247
JSON Encoded Data size in bytes (Cur/Min/Max): 0/0/0
CGPB Encoded Data size in bytes (Cur/Min/Max): 0/0/0
Collection Time in ms (Cur/Min/Max): 1/1/47
Encoding Time in ms (Cur/Min/Max): 1/1/1
Transport Time in ms (Cur/Min/Max): 4/1/6
Streaming Time in ms (Cur/Min/Max): 47369/1/47369

```

show telemetry transport sessions

The following commands loop through all the transport sessions and prints the information in one command:

```

switch# show telemetry transport sessions
switch# show telemetry transport stats
switch# show telemetry transport errors
switch# show telemetry transport all

```

The following is an example for telemetry transport session:

```

switch# show telemetry transport sessions
Session Id:      0
IP Address:Port  172.27.254.13:50004

```

```

Transport:          GRPC
Status:             Transmit Error
SSL Certificate:    trustpoint1
Last Connected:     Never
Last Disconnected:  Never
Tx Error Count:     2
Last Tx Error:      Wed Aug 19 23:32:21.749 UTC
...
Session Id:         4
IP Address:Port     172.27.254.13:50006
Transport:          UDP

```

Telemetry Ephemeral Event

To support ephemeral event, a new sensor path query-condition is added. To enable accounting log ephemeral event streaming, use the following query condition:

```

sensor-group 1
path sys/accounting/log query-condition query-target=subtree&complete-mo=yes&notify-interval=1

```

The following are the other sensor paths that support ephemeral event:

```

sys/pim/inst/routedb-route, sys/pim/pimifdb-adj, sys/pim/pimifdb-prop
sys/igmp/igmpifdb-prop, sys/igmp/inst/routedb, sys/igmpsnoop/inst/dom/db-extrack,
sys/igmpsnoop/inst/dom/db-group, sys/igmpsnoop/inst/dom/db-mrouter
sys/igmpsnoop/inst/dom/db-querier, sys/igmpsnoop/inst/dom/db-snoop

```

Displaying Telemetry Log and Trace Information

Use the following NX-OS CLI commands to display the log and trace information.

show tech-support telemetry

This NX-OS CLI command collects the telemetry log contents from the tech-support log. In this example, the command output is redirected into a file in bootflash.

```
switch# show tech-support telemetry > bootflash:tmst.log
```

tmtrace.bin

This BASH shell command collects telemetry traces and prints them out.

```

switch# configure terminal
switch(config)# feature bash
switch(config)# run bash
bash-4.2$ tmtrace.bin -d tm-errors
bash-4.2$ tmtrace.bin -d tm-logs
bash-4.2$ tmtrace.bin -d tm-events

```

For example:

```

bash-4.2$ tmtrace.bin -d tm-logs
[01/25/17 22:52:24.563 UTC 1 29130] [3944724224][tm_ec_dme_auth.c:59] TM_EC: Authentication
refresh url http://127.0.0.1/api/aaaRefresh.json
[01/25/17 22:52:24.565 UTC 2 29130] [3944724224][tm_ec_dme_rest_util.c:382] TM_EC: Performed
POST request on http://127.0.0.1/api/aaaRefresh.json
[01/25/17 22:52:24.566 UTC 3 29130] [3944724224][tm_mgd_timers.c:114] TM_MGD_TIMER: Starting
leaf timer for leaf:0x11e17ea4 time_in_ms:540000

```



```
[01/25/17 22:52:45.317 UTC 4 29130] [3944724224][tm_ec_dme_event_subsc.c:790] TM_EC: Event
subscription database size 0
[01/25/17 22:52:45.317 UTC 5 29130] [3944724224][tm_mgd_timers.c:114] TM_MGD_TIMER: Starting
leaf timer for leaf:0x11e17e3c time_in_ms:50000
bash-4.2#
```



Note The **tm-logs** option is not enabled by default because it is verbose.

Enable **tm-logs** with the `tmtrace.bin -L D tm-logs` command.

Disable **tm-logs** with the `tmtrace.bin -L W tm-logs` command.

show system internal telemetry trace

The **show system internal telemetry trace** [**tm-events** | **tm-errors** | **tm-logs** | **all**] command displays system internal telemetry trace information.

```
switch# show system internal telemetry trace all
Telemetry All Traces:
Telemetry Error Traces:
[07/26/17 15:22:29.156 UTC 1 28577] [3960399872][tm_cfg_api.c:367] Not able to destroy dest
profile list for config node rc:-1610612714 reason:Invalid argument
[07/26/17 15:22:44.972 UTC 2 28577] [3960399872][tm_stream.c:248] No subscriptions for
destination group 1
[07/26/17 15:22:49.463 UTC 3 28577] [3960399872][tm_stream.c:576] TM_STREAM: Subscriptoin
1 does not have any sensor groups

3 entries printed
Telemetry Event Traces:
[07/26/17 15:19:40.610 UTC 1 28577] [3960399872][tm_debug.c:41] Telemetry xostrace buffers
initialized successfully!
[07/26/17 15:19:40.610 UTC 2 28577] [3960399872][tm.c:744] Telemetry statistics created
successfully!
[07/26/17 15:19:40.610 UTC 3 28577] [3960399872][tm_init_n9k.c:97] Platform intf:
grpc_traces:compression,channel
switch#

switch# show system internal telemetry trace tm-logs
Telemetry Log Traces:
0 entries printed
switch#

switch# show system internal telemetry trace tm-events
Telemetry Event Traces:
[07/26/17 15:19:40.610 UTC 1 28577] [3960399872][tm_debug.c:41] Telemetry xostrace buffers
initialized successfully!
[07/26/17 15:19:40.610 UTC 2 28577] [3960399872][tm.c:744] Telemetry statistics created
successfully!
[07/26/17 15:19:40.610 UTC 3 28577] [3960399872][tm_init_n9k.c:97] Platform intf:
grpc_traces:compression,channel
[07/26/17 15:19:40.610 UTC 4 28577] [3960399872][tm_init_n9k.c:207] Adding telemetry to
cgroup
[07/26/17 15:19:40.670 UTC 5 28577] [3960399872][tm_init_n9k.c:215] Added telemetry to
cgroup successfully!

switch# show system internal telemetry trace tm-errors
Telemetry Error Traces:
0 entries printed
switch#
```

Configuring Telemetry Using the NX-API

Configuring Telemetry Using the NX-API

In the object model of the switch DME, the configuration of the telemetry feature is defined in a hierarchical structure of objects as shown in the section "Telemetry Model in the DME." Following are the main objects to be configured:

- **fmEntity** — Contains the NX-API and Telemetry feature states.
 - **fmNxapi** — Contains the NX-API state.
 - **fmTelemetry** — Contains the Telemetry feature state.
- **telemetryEntity** — Contains the telemetry feature configuration.
 - **telemetrySensorGroup** — Contains the definitions of one or more sensor paths or nodes to be monitored for telemetry. The telemetry entity can contain one or more sensor groups.
 - **telemetryRtSensorGroupRel** — Associates the sensor group with a telemetry subscription.
 - **telemetrySensorPath** — A path to be monitored. The sensor group can contain multiple objects of this type.
 - **telemetryDestGroup** — Contains the definitions of one or more destinations to receive telemetry data. The telemetry entity can contain one or more destination groups.
 - **telemetryRtDestGroupRel** — Associates the destination group with a telemetry subscription.
 - **telemetryDest** — A destination address. The destination group can contain multiple objects of this type.
 - **telemetrySubscription** — Specifies how and when the telemetry data from one or more sensor groups is sent to one or more destination groups.
 - **telemetryRsDestGroupRel** — Associates the telemetry subscription with a destination group.
 - **telemetryRsSensorGroupRel** — Associates the telemetry subscription with a sensor group.
 - **telemetryCertificate** — Associates the telemetry subscription with a certificate and hostname.

To configure the telemetry feature using the NX-API, you must construct a JSON representation of the telemetry object structure and push it to the DME with an HTTP or HTTPS POST operation.



Note For detailed instructions on using the NX-API, see the *Cisco Nexus 3000 and 9000 Series NX-API REST SDK User Guide and API Reference*.

Before you begin

Your switch must be running Cisco NX-OS Release 7.3(0)I5(1) or a later release.

Your switch must be configured to run the NX-API from the CLI:

```
switch(config)# feature nxapi
```

NX-API sends telemetry data over management VRF:

```
switch(config)# nxapi use-vrf management
```

```
nxapi use-vrf vrf_name  
nxapi http port port_number
```

SUMMARY STEPS

1. Enable the telemetry feature.
2. Create the root level of the JSON payload to describe the telemetry configuration.
3. Create a sensor group to contain the defined sensor paths.
4. (Optional) Add an SSL/TLS certificate and a host.
5. Define a telemetry destination group.
6. Define a telemetry destination profile.
7. Define one or more telemetry destinations, consisting of an IP address and port number to which telemetry data will be sent.
8. Enable gRPC chunking and set the chunking size, between 64 and 4096 bytes.
9. Create a telemetry subscription to configure the telemetry behavior.
10. Add the sensor group object as a child object to the **telemetrySubscription** element under the root element (**telemetryEntity**).
11. Create a relation object as a child object of the subscription to associate the subscription to the telemetry sensor group and to specify the data sampling behavior.
12. Define one or more sensor paths or nodes to be monitored for telemetry.
13. Add sensor paths as child objects to the sensor group object (**telemetrySensorGroup**).
14. Add destinations as child objects to the destination group object (**telemetryDestGroup**).
15. Add the destination group object as a child object to the root element (**telemetryEntity**).
16. Create a relation object as a child object of the telemetry sensor group to associate the sensor group to the subscription.
17. Create a relation object as a child object of the telemetry destination group to associate the destination group to the subscription.
18. Create a relation object as a child object of the subscription to associate the subscription to the telemetry destination group.
19. Send the resulting JSON structure as an HTTP/HTTPS POST payload to the NX-API endpoint for telemetry configuration.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	Enable the telemetry feature. Example: <pre>{ "fmEntity" : { "children" : [{ "fmTelemetry" : { "attributes" : { "adminSt" : "enabled" } }] } }</pre>	The root element is fmTelemetry and the base path for this element is <code>sys/fm</code>. Configure the adminSt attribute as <code>enabled</code>.
Step 2	Create the root level of the JSON payload to describe the telemetry configuration. Example: <pre>{ "telemetryEntity": { "attributes": { "dn": "sys/tm" }, } }</pre>	The root element is telemetryEntity and the base path for this element is <code>sys/tm</code>. Configure the dn attribute as <code>sys/tm</code>.
Step 3	Create a sensor group to contain the defined sensor paths. Example: <pre>"telemetrySensorGroup": { "attributes": { "id": "10", "rn": "sensor-10" }, "children": [{ }] }</pre>	A telemetry sensor group is defined in an object of class telemetrySensorGroup. Configure the following attributes of the object: • id — An identifier for the sensor group. Currently only numeric ID values are supported. • rn — The relative name of the sensor group object in the format: sensor-id. • dataSrc — Selects the data source from DEFAULT, DME, YANG, or NX-API. Children of the sensor group object include sensor paths and one or more relation objects (telemetryRtSensorGroupRel) to associate the sensor group with a telemetry subscription.

	Command or Action	Purpose
Step 4	(Optional) Add an SSL/TLS certificate and a host. Example: <pre>{ "telemetryCertificate": { "attributes": { "filename": "root.pem" "hostname": "c.com" } } }</pre>	The telemetryCertificate defines the location of the SSL/TLS certificate with the telemetry subscription/destination.
Step 5	Define a telemetry destination group. Example: <pre>{ "telemetryDestGroup": { "attributes": { "id": "20" } } }</pre>	A telemetry destination group is defined in telemetryEntity . Configure the id attribute.
Step 6	Define a telemetry destination profile. Example: <pre>{ "telemetryDestProfile": { "attributes": { "adminSt": "enabled" }, "children": [{ "telemetryDestOptSourceInterface": { "attributes": { "name": "lo0" } }]] } }</pre>	A telemetry destination profile is defined in telemetryDestProfile. • Configure the adminSt attribute as <i>enabled</i>. • Under telemetryDestOptSourceInterface, configure the name attribute with an interface name to stream data from the configured interface to a destination with the source IP address.
Step 7	Define one or more telemetry destinations, consisting of an IP address and port number to which telemetry data will be sent. Example: <pre>{ "telemetryDest": { "attributes": { "addr": "1.2.3.4", "enc": "GPB", "port": "50001", "proto": "gRPC", "rn": "addr-[1.2.3.4]-port-50001" } } }</pre>	A telemetry destination is defined in an object of class telemetryDest. Configure the following attributes of the object: • addr — The IP address of the destination. • port — The port number of the destination. • rn — The relative name of the destination object in the format: path-[path]. • enc — The encoding type of the telemetry data to be sent. NX-OS supports:

	Command or Action	Purpose
	<pre> } } } </pre>	• Google protocol buffers (GPB) for gRPC. • JSON for C. • GPB or JSON for UDP and secure UDP (DTLS). • proto — The transport protocol type of the telemetry data to be sent. NX-OS supports: • gRPC • HTTP • VUDP and secure UDP (DTLS) • Supported encoded types are: • HTTP/JSON YES • HTTP/Form-data YES Only supported for Bin Logging. • GRPC/GPB-Compact YES Native Data Source Only. • GRPC/GPB YES • UDP/GPB YES • UDP/JSON YES
Step 8	Enable gRPC chunking and set the chunking size, between 64 and 4096 bytes. Example: <pre> { "telemetryDestGrpOptChunking": { "attributes": { "chunkSize": "2048", "dn": "sys/tm/dest-1/chunking" } } } </pre>	See Support for gRPC Chunking, on page 333 for more information.
Step 9	Create a telemetry subscription to configure the telemetry behavior. Example: <pre> "telemetrySubscription": { "attributes": { "id": "30", "rn": "subs-30" }, "children": [{ </pre>	A telemetry subscription is defined in an object of class telemetrySubscription. Configure the following attributes of the object: • id — An identifier for the subscription. Currently only numeric ID values are supported. • rn — The relative name of the subscription object in the format: subs-id.

	Command or Action	Purpose
	<pre>]] } </pre>	Children of the subscription object include relation objects for sensor groups (telemetryRsSensorGroupRel) and destination groups (telemetryRsDestGroupRel).
Step 10	Add the sensor group object as a child object to the telemetrySubscription element under the root element (telemetryEntity). Example: <pre> { "telemetrySubscription": { "attributes": { "id": "30" } "children": [{ "telemetryRsSensorGroupRel": { "attributes": { "sampleIntvl": "5000", "tDn": "sys/tm/sensor-10" } }] } } </pre>	
Step 11	Create a relation object as a child object of the subscription to associate the subscription to the telemetry sensor group and to specify the data sampling behavior. Example: <pre> "telemetryRsSensorGroupRel": { "attributes": { "rType": "mo", "rn": "rssensorGroupRel-[sys/tm/sensor-10]", "sampleIntvl": "5000", "tCl": "telemetrySensorGroup", "tDn": "sys/tm/sensor-10", "tType": "mo" } } </pre>	The relation object is of class telemetryRsSensorGroupRel and is a child object of telemetrySubscription. Configure the following attributes of the relation object: • rn — The relative name of the relation object in the format: rssensorGroupRel-[sys/tm/sensor-group-id]. • sampleIntvl — The data sampling period in milliseconds. An interval value of 0 creates an event-based subscription, in which telemetry data is sent only upon changes under the specified MO. An interval value greater than 0 creates a frequency-based subscription, in which telemetry data is sent periodically at the specified interval. For example, an interval value of 15000 results in the sending of telemetry data every 15 seconds. • tCl — The class of the target (sensor group) object, which is telemetrySensorGroup. • tDn — The distinguished name of the target (sensor group) object, which is sys/tm/sensor-group-id. • rType — The relation type, which is mo for managed object. • tType — The target type, which is mo for managed object.

	Command or Action	Purpose
Step 12	Define one or more sensor paths or nodes to be monitored for telemetry. Example: Single sensor path <pre>{ "telemetrySensorPath": { "attributes": { "path": "sys/cdp", "rn": "path-[sys/cdp]", "excludeFilter": "", "filterCondition": "", "path": "sys/fm/bgp", "secondaryGroup": "0", "secondaryPath": "", "depth": "0", "alias": "cdp_alias", } } }</pre> Example: Single sensor path for NX-API <pre>{ "telemetrySensorPath": { "attributes": { "path": "show interface", "path": "show bgp", "rn": "path-[sys/cdp]", "excludeFilter": "", "filterCondition": "", "path": "sys/fm/bgp", "secondaryGroup": "0", "secondaryPath": "", "depth": "0" } } }</pre> Example: Multiple sensor paths <pre>{ "telemetrySensorPath": { "attributes": { "path": "sys/cdp", "rn": "path-[sys/cdp]", "excludeFilter": "", "filterCondition": "", "path": "sys/fm/bgp", "secondaryGroup": "0", "secondaryPath": "", "depth": "0" } } }</pre>	A sensor path is defined in an object of class telemetrySensorPath. Configure the following attributes of the object: • path — The path to be monitored. • rn — The relative name of the path object in the format: path-[path] • depth — The retrieval level for the sensor path. A depth setting of 0 retrieves only the root MO properties. • filterCondition — (Optional) Creates a specific filter for event-based subscriptions. The DME provides the filter expressions. For more information about filtering, see the Cisco APIC REST API Usage Guidelines on composing queries: https://www.cisco.com/c/en/us/td/docs/switches/datacenter/aci/apic/sw/2-x/rest_cfg/2_1_x/b_Cisco_APIC_REST_API_Configuration_Guide/b_Cisco_APIC_REST_API_Configuration_Guide_chapter_01.html#d25e1534a1635 • alias - Specify an alias for this path.

	Command or Action	Purpose
	<pre> } }, { "telemetrySensorPath": { "attributes": { "excludeFilter": "", "filterCondition": "", "path": "sys/fm/dhcp", "secondaryGroup": "0", "secondaryPath": "", "depth": "0" } } } } </pre> Example: Single sensor path filtering for BGP disable events: <pre> { "telemetrySensorPath": { "attributes": { "path": "sys/cdp", "rn": "path-[sys/cdp]", "excludeFilter": "", "filterCondition": "eq(fmBgp.operSt.\"disabled\")", "path": "sys/fm/bgp", "secondaryGroup": "0", "secondaryPath": "", "depth": "0" } } } </pre>	
Step 13	Add sensor paths as child objects to the sensor group object (telemetrySensorGroup).	
Step 14	Add destinations as child objects to the destination group object (telemetryDestGroup).	
Step 15	Add the destination group object as a child object to the root element (telemetryEntity).	
Step 16	Create a relation object as a child object of the telemetry sensor group to associate the sensor group to the subscription. Example: <pre> "telemetryRtSensorGroupRel": { "attributes": { "rn": "rtsensorGroupRel-[sys/tm/subs-30]", "tCl": "telemetrySubscription", "tDn": "sys/tm/subs-30" } } </pre>	The relation object is of class telemetryRtSensorGroupRel and is a child object of telemetrySensorGroup. Configure the following attributes of the relation object: • rn — The relative name of the relation object in the format: rtsensorGroupRel-[sys/tm/subscription-id]. • tCl — The target class of the subscription object, which is telemetrySubscription. • tDn — The target distinguished name of the subscription object, which is sys/tm/subscription-id.

	Command or Action	Purpose
Step 17	Create a relation object as a child object of the telemetry destination group to associate the destination group to the subscription. Example: <pre>"telemetryRtDestGroupRel": { "attributes": { "rn": "rtdestGroupRel-[sys/tm/subs-30]", "tCl": "telemetrySubscription", "tDn": "sys/tm/subs-30" } }</pre>	The relation object is of class telemetryRtDestGroupRel and is a child object of telemetryDestGroup. Configure the following attributes of the relation object: • rn — The relative name of the relation object in the format: rtdestGroupRel-[sys/tm/subscription-id]. • tCl — The target class of the subscription object, which is telemetrySubscription. • tDn — The target distinguished name of the subscription object, which is sys/tm/subscription-id.
Step 18	Create a relation object as a child object of the subscription to associate the subscription to the telemetry destination group. Example: <pre>"telemetryRsDestGroupRel": { "attributes": { "rType": "mo", "rn": "rsdestGroupRel-[sys/tm/dest-20]", "tCl": "telemetryDestGroup", "tDn": "sys/tm/dest-20", "tType": "mo" } }</pre>	The relation object is of class telemetryRsDestGroupRel and is a child object of telemetrySubscription. Configure the following attributes of the relation object: • rn — The relative name of the relation object in the format: rsdestGroupRel-[sys/tm/destination-group-id]. • tCl — The class of the target (destination group) object, which is telemetryDestGroup. • tDn — The distinguished name of the target (destination group) object, which is sys/tm/destination-group-id. • rType — The relation type, which is mo for managed object. • tType — The target type, which is mo for managed object.
Step 19	Send the resulting JSON structure as an HTTP/HTTPS POST payload to the NX-API endpoint for telemetry configuration.	The base path for the telemetry entity is sys/tm and the NX-API endpoint is: <pre>{{URL}}/api/node/mo/sys/tm.json</pre>

Example

The following is an example of all the previous steps that are collected into one POST payload (note that some attributes may not match):

```
{
  "telemetryEntity": {
    "children": [{
      "telemetrySensorGroup": {
        "attributes": {
          "id": "10"
        }
      },
      "children": [{
        "telemetrySensorPath": {
          "attributes": {

```

```

        "excludeFilter": "",
        "filterCondition": "",
        "path": "sys/fm/bgp",
        "secondaryGroup": "0",
        "secondaryPath": "",
        "depth": "0"
    }
}
]
},
{
    "telemetryDestGroup": {
        "attributes": {
            "id": "20"
        }
        "children": [{
            "telemetryDest": {
                "attributes": {
                    "addr": "10.30.217.80",
                    "port": "50051",
                    "enc": "GPB",
                    "proto": "gRPC"
                }
            }
        }
    ]
},
{
    "telemetrySubscription": {
        "attributes": {
            "id": "30"
        }
        "children": [{
            "telemetryRsSensorGroupRel": {
                "attributes": {
                    "sampleIntvl": "5000",
                    "tDn": "sys/tm/sensor-10"
                }
            }
        },
        {
            "telemetryRsDestGroupRel": {
                "attributes": {
                    "tDn": "sys/tm/dest-20"
                }
            }
        }
    ]
}
]
}
}

```

Configuration Example for Telemetry Using the NX-API

Streaming Paths to a Destination

This example creates a subscription that streams paths `sys/cdp` and `sys/ipv4` to a destination `1.2.3.4 port 50001` every five seconds.

POST `https://192.168.20.123/api/node/mo/sys/tm.json`

Payload:

```
{
  "telemetryEntity": {
    "attributes": {
      "dn": "sys/tm"
    },
    "children": [{
      "telemetrySensorGroup": {
        "attributes": {
          "id": "10",
          "rn": "sensor-10"
        },
        "children": [{
          "telemetryRtSensorGroupRel": {
            "attributes": {
              "rn": "rtsensorGroupRel-[sys/tm/subs-30]",
              "tCl": "telemetrySubscription",
              "tDn": "sys/tm/subs-30"
            }
          }
        ]
      }, {
        "telemetrySensorPath": {
          "attributes": {
            "path": "sys/cdp",
            "rn": "path-[sys/cdp]",
            "excludeFilter": "",
            "filterCondition": "",
            "secondaryGroup": "0",
            "secondaryPath": "",
            "depth": "0"
          }
        }
      }, {
        "telemetrySensorPath": {
          "attributes": {
            "path": "sys/ipv4",
            "rn": "path-[sys/ipv4]",
            "excludeFilter": "",
            "filterCondition": "",
            "secondaryGroup": "0",
            "secondaryPath": "",
            "depth": "0"
          }
        }
      }
    ]
  }, {
    "telemetryDestGroup": {
      "attributes": {
        "id": "20",
        "rn": "dest-20"
      },
      "children": [{
        "telemetryRtDestGroupRel": {
```

Filter Conditions on BGP Notifications

```
Payload:
{
  "telemetryEntity": {
```

```

"children": [{
  "telemetrySensorGroup": {
    "attributes": {
      "id": "10"
    }
  }
  "children": [{
    "telemetrySensorPath": {
      "attributes": {
        "excludeFilter": "",
        "filterCondition": "eq(fmBgp.operSt,\"disabled\")",
        "path": "sys/fm/bgp",
        "secondaryGroup": "0",
        "secondaryPath": "",
        "depth": "0"
      }
    }
  ]
}
],
{
  "telemetryDestGroup": {
    "attributes": {
      "id": "20"
    }
  }
  "children": [{
    "telemetryDest": {
      "attributes": {
        "addr": "10.30.217.80",
        "port": "50055",
        "enc": "GPB",
        "proto": "gRPC"
      }
    }
  ]
}
],
{
  "telemetrySubscription": {
    "attributes": {
      "id": "30"
    }
  }
  "children": [{
    "telemetryRsSensorGroupRel": {
      "attributes": {
        "sampleIntvl": "0",
        "tDn": "sys/tm/sensor-10"
      }
    }
  },
  {
    "telemetryRsDestGroupRel": {
      "attributes": {
        "tDn": "sys/tm/dest-20"
      }
    }
  }
]
}
]
}
}

```

Using Postman Collection for Telemetry Configuration

An [example Postman collection](#) is an easy way to start configuring the telemetry feature, and can run all telemetry CLI equivalents in a single payload. Modify the file in the preceding link using your preferred text editor to update the payload to your needs, then open the collection in Postman and run the collection.

Telemetry Model in the DME

The telemetry application is modeled in the DME with the following structure:

```
model
|----package [name:telemetry]
|   @name:telemetry
|   |----objects
|       |----mo [name:Entity]
|           |   @name:Entity
|           |   @label:Telemetry System
|           |--property
|           |   @name:adminSt
|           |   @type:AdminState
|           |
|           |----mo [name:SensorGroup]
|               |   @name:SensorGroup
|               |   @label:Sensor Group
|               |--property
|               |   @name:id [key]
|               |   @type:string:Basic
|               |   @name:dataSrc
|               |   @type:DataSource
|               |
|               |----mo [name:SensorPath]
|                   |   @name:SensorPath
|                   |   @label:Sensor Path
|                   |--property
|                   |   @name:path [key]
|                   |   @type:string:Basic
|                   |   @name:filterCondition
|                   |   @type:string:Basic
|                   |   @name:excludeFilter
|                   |   @type:string:Basic
|                   |   @name:depth
|                   |   @type:RetrieveDepth
|                   |
|                   |----mo [name:DestGroup]
|                       |   @name:DestGroup
|                       |   @label:Destination Group
|                       |--property
|                       |   @name:id
|                       |   @type:string:Basic
|                       |
|                       |----mo [name:Dest]
|                           |   @name:Dest
|                           |   @label:Destination
|                           |--property
|                           |   @name:addr [key]
|                           |   @type:address:Ip
|                           |   @name:port [key]
|                           |   @type:scalar:Uint16
|                           |   @name:proto
|                           |   @type:Protocol
|                           |   @name:enc
```

```

|           |           @type:Encoding
|
|-----mo [name:Subscription]
|           |           @name:Subscription
|           |           @label:Subscription
|           |--property
|           |           @name:id
|           |           @type:scalar:Uint64
|           |-----reldef
|           |           |           @name:SensorGroupRel
|           |           |           @to:SensorGroup
|           |           |           @cardinality:ntom
|           |           |           @label:Link to sensorGroup entry
|           |           |--property
|           |           |           @name:sampleIntvl
|           |           |           @type:scalar:Uint64
|           |
|           |-----reldef
|           |           |           @name:DestGroupRel
|           |           |           @to:DestGroup
|           |           |           @cardinality:ntom
|           |           |           @label:Link to destGroup entry

```

For a list of DNs available to the telemetry feature, see *Streaming Telemetry Sources*.

Telemetry Path Labels

About Telemetry Path Labels

Beginning with NX-OS release 9.3(1), model-driven telemetry supports path labels. Path labels provide an easy way to gather telemetry data from multiple sources at once. With this feature, you specify the type of telemetry data you want collected, and the telemetry feature gathers that data from multiple paths. The feature then returns the information to one consolidated place, the path label. This feature simplifies using telemetry because you no longer must:

- Have a deep and comprehensive knowledge of the Cisco DME model.
- Create multiple queries and add multiple paths to the subscription, while balancing the number of collected events and the cadence.
- Collect multiple chunks of telemetry information from the switch, which simplifies serviceability.

Path labels span across multiple instances of the same object type in the model, then gather and return counters or events. Path labels support the following telemetry groups:

- Environment, which monitors chassis information, including fan, temperature, power, storage, supervisors, and line cards.
- Interface, which monitors all the interface counters and status changes.

This label supports predefined keyword filters that can refine the returned data by using the **query-condition** command.

- Resources, which monitors system resources such as CPU utilization and memory utilization.

- VXLAN, which monitors VXLAN EVPNs including VXLAN peers, VXLAN counters, VLAN counters, and BGP Peer data.

Polling for Data or Receiving Events

The sample interval for a sensor group determines how and when telemetry data is transmitted to a path label. The sample interval can be configured either to periodically poll for telemetry data or gather telemetry data when events occur.

- When the sample interval for telemetry is configured as a non-zero value, telemetry periodically sends the data for the environment, interfaces, resources, and VXLAN labels during each sample interval.
- When the sample interval is set to zero, telemetry sends event notifications when the environment, interfaces, resources, and VXLAN labels experience operational state updates, as well as creation and deletion of MOs.

Polling for data or receiving events are mutually exclusive. You can configure polling or event-driven telemetry for each path label.

Guidelines and Limitations for Path Labels

The telemetry path labels feature has the following guidelines and limitations:

- The feature supports only Cisco DME data source only.
- You cannot mix and match usability paths with regular DME paths in the same sensor group. For example, you cannot configure `sys/intf` and `interface` in the same sensor group. Also, you cannot configure the same sensor group with `sys/intf` and `interface`. If this situation occurs, NX-OS rejects the configuration.
- User filter keywords, such as `oper-speed` and `counters=[detailed]`, are supported only for the `interface` path.
- The feature does not support other sensor path options, such as `depth` or `filter-condition`.

Configuring the Interface Path to Poll for Data or Events

The interface path label monitors all the interface counters and status changes. It supports the following interface types:

- Physical
- Subinterface
- Management
- Loopback
- VLAN
- Port Channel

You can configure the interface path label to either periodically poll for data or receive events. See [Polling for Data or Receiving Events](#), on page 373.



Note The model does not support counters for subinterface, loopback, or VLAN, so they are not streamed out.

SUMMARY STEPS

1. **configure terminal**
2. **telemetry**
3. **sensor-group** *sgrp_id*
4. **path interface**
5. **destination-group** *grp_id*
6. **ip address** *ip_addr* **port** *port*
7. **subscription** *sub_id*
8. **snsr-group** *sgrp_id* **sample-interval** *interval*
9. **dst-group** *dgrp_id*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	configure terminal Example: <pre>switch-1# configure terminal switch-1(config)#</pre>	Enter configuration mode.
Step 2	telemetry Example: <pre>switch-1(config)# telemetry switch-1(config-telemetry)#</pre>	Enter configuration mode for the telemetry features.
Step 3	sensor-group <i>sgrp_id</i> Example: <pre>switch-1(config-telemetry)# sensor-group 6 switch-1(conf-tm-sensor)#</pre>	Create a sensor group for telemetry data.
Step 4	path interface Example: <pre>switch-1(conf-tm-sensor)# path interface switch-1(conf-tm-sensor)#</pre>	Configure the interface path label, which enables sending one telemetry data query for multiple individual interfaces. The label consolidates the queries for multiple interfaces into one. Telemetry then telemetry gathers the data and returns it to the label. Depending on how the polling interval is configured, interface data is sent based on a periodic basis or whenever the interface state changes.

	Command or Action	Purpose
Step 5	destination-group <i>grp_id</i> Example: <pre>switch-1(conf-tm-sensor) # destination-group 33 switch-1(conf-tm-dest) #</pre>	Enter telemetry destination group submode and configure the destination group.
Step 6	ip address <i>ip_addr</i> port <i>port</i> Example: <pre>switch-1(conf-tm-dest) # ip address 1.2.3.4 port 50004 switch-1(conf-tm-dest) #</pre>	Configure the telemetry data for the subscription to stream to the specified IP address and port.
Step 7	subscription <i>sub_id</i> Example: <pre>switch-1(conf-tm-dest) # subscription 33 switch-1(conf-tm-sub) #</pre>	Enter telemetry subscription submode, and configure the telemetry subscription.
Step 8	snsr-group <i>sgrp_id</i> sample-interval <i>interval</i> Example: <pre>switch-1(conf-tm-sub) # snsr-grp 6 sample-interval 5000 switch-1(conf-tm-sub) #</pre>	Link the sensor group to the current subscription and set the data sampling interval in milliseconds. The sampling interval determines whether the switch sends telemetry data periodically, or when interface events occur.
Step 9	dst-group <i>dgrp_id</i> Example: <pre>switch-1(conf-tm-sub) # dst-grp 33 switch-1(conf-tm-sub) #</pre>	Link the destination group to the current subscription. The destination group that you specify must match the destination group that you configured in the destination-group command.

Configuring the Interface Path for Non-Zero Counters

You can configure the interface path label with a predefined keyword filter that returns only counters that have nonzero values. The filter is `counters=[detailed]`.

By using this filter, the interface path gathers all the available interface counters, filters the collected data, then forwards the results to the receiver. The filter is optional, and if you do not use it, all counters, including zero-value counters, are displayed for the interface path.



Note Using the filter is conceptually similar to issuing **show interface mgmt0 counters detailed**

SUMMARY STEPS

1. **configure terminal**
2. **telemetry**
3. **sensor-group** *sgrp_id*
4. **path interface query-condition** `counters=[detailed]`
5. **destination-group** *grp_id*

6. **ip address** *ip_addr* **port** *port*
7. **subscription** *sub_id*
8. **snsr-group** *sgrp_id* **sample-interval** *interval*
9. **dst-group** *dgrp_id*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	configure terminal Example: <pre>switch-1# configure terminal switch-1(config)#</pre>	Enter configuration mode.
Step 2	telemetry Example: <pre>switch-1(config)# telemetry switch-1(config-telemetry)#</pre>	Enter configuration mode for the telemetry features.
Step 3	sensor-group <i>sgrp_id</i> Example: <pre>switch-1(config-telemetry)# sensor-group 6 switch-1(conf-tm-sensor)#</pre>	Create a sensor group for telemetry data.
Step 4	path interface query-condition counters=[detailed] Example: <pre>switch-1(conf-tm-sensor)# path interface query-condition counters=[detailed] switch-1(conf-tm-sensor)#</pre>	Configure the interface path label and query for only the nonzero counters from all interfaces.
Step 5	destination-group <i>grp_id</i> Example: <pre>switch-1(conf-tm-sensor)# destination-group 33 switch-1(conf-tm-dest)#</pre>	Enter telemetry destination group submode and configure the destination group.
Step 6	ip address <i>ip_addr</i> port <i>port</i> Example: <pre>switch-1(conf-tm-dest)# ip address 1.2.3.4 port 50004 switch-1(conf-tm-dest)#</pre>	Configure the telemetry data for the subscription to stream to the specified IP address and port.
Step 7	subscription <i>sub_id</i> Example: <pre>switch-1(conf-tm-dest)# subscription 33 switch-1(conf-tm-sub)#</pre>	Enter telemetry subscription submode, and configure the telemetry subscription.

	Command or Action	Purpose
Step 8	snsr-group <i>sgrp_id</i> sample-interval <i>interval</i> Example: <pre>switch-1(conf-tm-sub) # snsr-grp 6 sample-interval 5000 switch-1(conf-tm-sub) #</pre>	Link the sensor group to the current subscription and set the data sampling interval in milliseconds. The sampling interval determines whether the switch sends telemetry data periodically, or when interface events occur.
Step 9	dst-group <i>dgrp_id</i> Example: <pre>switch-1(conf-tm-sub) # dst-grp 33 switch-1(conf-tm-sub) #</pre>	Link the destination group to the current subscription. The destination group that you specify must match the destination group that you configured in the destination-group command.

Configuring the Interface Path for Operational Speeds

You can configure the interface path label with a pre-defined keyword filter that returns counters for interfaces of specified operational speeds. The filter is `oper-speed=[ ]`. The following operational speeds are supported: auto, 10M, 100M, 1G, 10G, 40G, 200G, and 400G.

By using this filter, the interface path gathers the telemetry data for interfaces of the specified speed, then forwards the results to the receiver. The filter is optional. If you do not use it, counters for all interfaces are displayed, regardless of their operational speed.

The filter can accept multiple speeds as a comma-separated list, for example `oper-speed=[1G,10G]` to retrieve counters for interfaces that operate at 1 and 10 Gbps. Do not use a blank space as a delimiter.



Note Interface types subinterface, loopback, and VLAN do not have operational speed properties, so the filter does not support these interface types.

SUMMARY STEPS

1. **configure terminal**
2. **telemetry**
3. **snsr-group** *sgrp_id* **sample-interval** *interval*
4. **path interface query-condition** `oper-speed=[speed]`
5. **destination-group** *dgrp_id*
6. **ip address** *ip_addr* **port** *port*
7. **subscription** *sub_id*
8. **snsr-group** *sgrp_id* **sample-interval** *interval*
9. **dst-group** *dgrp_id*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	configure terminal Example: <pre>switch-1# configure terminal switch-1(config)#</pre>	Enter configuration mode.
Step 2	telemetry Example: <pre>switch-1(config)# telemetry switch-1(config-telemetry)#</pre>	Enter configuration mode for the telemetry features.
Step 3	snsr-group <i>sgrp_id</i> sample-interval <i>interval</i> Example: <pre>switch-1(conf-tm-sub)# snsr-grp 6 sample-interval 5000 switch-1(conf-tm-sub)#</pre>	Link the sensor group to the current subscription and set the data sampling interval in milliseconds. The sampling interval determines whether the switch sends telemetry data periodically, or when interface events occur.
Step 4	path interface query-condition oper-speed=[<i>speed</i>] Example: <pre>switch-1(conf-tm-sensor)# path interface query-condition oper-speed=[1G,40G] switch-1(conf-tm-sensor)#</pre>	Configure the interface path label and query for counters from interfaces running the specified speed, which in this example, is 1 and 40 Gbps only.
Step 5	destination-group <i>grp_id</i> Example: <pre>switch-1(conf-tm-sensor)# destination-group 33 switch-1(conf-tm-dest)#</pre>	Enter telemetry destination group submode and configure the destination group.
Step 6	ip address <i>ip_addr</i> port <i>port</i> Example: <pre>switch-1(conf-tm-dest)# ip address 1.2.3.4 port 50004 switch-1(conf-tm-dest)#</pre>	Configure the telemetry data for the subscription to stream to the specified IP address and port.
Step 7	subscription <i>sub_id</i> Example: <pre>switch-1(conf-tm-dest)# subscription 33 switch-1(conf-tm-sub)#</pre>	Enter telemetry subscription submode, and configure the telemetry subscription.
Step 8	snsr-group <i>sgrp_id</i> sample-interval <i>interval</i> Example: <pre>switch-1(conf-tm-sub)# snsr-grp 6 sample-interval 5000 switch-1(conf-tm-sub)#</pre>	Link the sensor group to the current subscription and set the data sampling interval in milliseconds. The sampling interval determines whether the switch sends telemetry data periodically, or when interface events occur.

	Command or Action	Purpose
Step 9	dst-group <i>dgrp_id</i> Example: switch-1(conf-tm-sub) # dst-grp 33 switch-1(conf-tm-sub) #	Link the destination group to the current subscription. The destination group that you specify must match the destination group that you configured in the destination-group command.

Configuring the Interface Path with Multiple Queries

You can configure multiple filters for the same query condition in the interface path label. When you do so, the individual filters you use are ANDed.

Separate each filter in the query condition by using a comma. You can specify any number of filters for the query-condition, but the more filters you add, the more focused the results become.

SUMMARY STEPS

1. **configure terminal**
2. **telemetry**
3. **sensor-group** *sgrp_id*
4. **path interface query-condition** counters=[detailed],oper-speed=[1G,40G]
5. **destination-group** *grp_id*
6. **ip address** *ip_addr* **port** *port*
7. **subscription** *sub_id*
8. **snsr-group** *sgrp_id* **sample-interval** *interval*
9. **dst-group** *dgrp_id*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	configure terminal Example: switch-1# configure terminal switch-1(config)#	Enter configuration mode.
Step 2	telemetry Example: switch-1(config)# telemetry switch-1(config-telemetry)#	Enter configuration mode for the telemetry features.
Step 3	sensor-group <i>sgrp_id</i> Example: switch-1(config-telemetry)# sensor-group 6 switch-1(conf-tm-sensor)#	Create a sensor group for telemetry data.

	Command or Action	Purpose
Step 4	path interface query-condition counters=[detailed],oper-speed=[1G,40G] Example: <pre>switch-1(conf-tm-sensor)# path interface query-condition counters=[detailed],oper-speed=[1G,40G] switch-1(conf-tm-sensor)#</pre>	Configures multiple conditions in the same query. In this example, the query does both of the following: • Gathers and returns non-zero counters on interfaces running at 1 Gbps. • Gathers and returns non-zero counters on interfaces running at 40 Gbps.
Step 5	destination-group grp_id Example: <pre>switch-1(conf-tm-sensor)# destination-group 33 switch-1(conf-tm-dest)#</pre>	Enter telemetry destination group submode and configure the destination group.
Step 6	ip address ip_addr port port Example: <pre>switch-1(conf-tm-dest)# ip address 1.2.3.4 port 50004 switch-1(conf-tm-dest)#</pre>	Configure the telemetry data for the subscription to stream to the specified IP address and port.
Step 7	subscription sub_id Example: <pre>switch-1(conf-tm-dest)# subscription 33 switch-1(conf-tm-sub)#</pre>	Enter telemetry subscription submode, and configure the telemetry subscription.
Step 8	snsr-group sgrp_id sample-interval interval Example: <pre>switch-1(conf-tm-sub)# snsr-grp 6 sample-interval 5000 switch-1(conf-tm-sub)#</pre>	Link the sensor group to the current subscription and set the data sampling interval in milliseconds. The sampling interval determines whether the switch sends telemetry data periodically, or when interface events occur.
Step 9	dst-group dgrp_id Example: <pre>switch-1(conf-tm-sub)# dst-grp 33 switch-1(conf-tm-sub)#</pre>	Link the destination group to the current subscription. The destination group that you specify must match the destination group that you configured in the destination-group command.

Configuring the Environment Path to Poll for Data or Events

The environment path label monitors chassis information, including fan, temperature, power, storage, supervisors, and line cards. You can configure the environment path to either periodically poll for telemetry data or get the data when events occur. For information, see [Polling for Data or Receiving Events, on page 373](#).

You can set the resources path to return system resource information through either periodic polling or based on events. This path does not support filtering.

SUMMARY STEPS

1. **configure terminal**
2. **telemetry**
3. **sensor-group** *sgrp_id*
4. **path environment**
5. **destination-group** *grp_id*
6. **ip address** *ip_addr* **port** *port*
7. **subscription** *sub_id*
8. **snsr-group** *sgrp_id* **sample-interval** *interval*
9. **dst-group** *dgrp_id*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	configure terminal Example: <pre>switch-1# configure terminal switch-1(config)#</pre>	Enter configuration mode.
Step 2	telemetry Example: <pre>switch-1(config)# telemetry switch-1(config-telemetry)#</pre>	Enter configuration mode for the telemetry features.
Step 3	sensor-group <i>sgrp_id</i> Example: <pre>switch-1(config-telemetry)# sensor-group 6 switch-1(conf-tm-sensor)#</pre>	Create a sensor group for telemetry data.
Step 4	path environment Example: <pre>switch-1(conf-tm-sensor)# path environment switch-1(conf-tm-sensor)#</pre>	Configures the environment path label, which enables telemetry data for multiple individual environment objects to be sent to the label. The label consolidates the multiple data inputs into one output. Depending on the sample interval, the environment data is either streaming based on the polling interval, or sent when events occur.
Step 5	destination-group <i>grp_id</i> Example: <pre>switch-1(conf-tm-sensor)# destination-group 33 switch-1(conf-tm-dest)#</pre>	Enter telemetry destination group submode and configure the destination group.
Step 6	ip address <i>ip_addr</i> port <i>port</i> Example:	Configure the telemetry data for the subscription to stream to the specified IP address and port.

	Command or Action	Purpose
	<pre>switch-1(conf-tm-dest)# ip address 1.2.3.4 port 50004 switch-1(conf-tm-dest)#</pre>	
Step 7	subscription <i>sub_id</i> Example: <pre>switch-1(conf-tm-dest)# subscription 33 switch-1(conf-tm-sub)#</pre>	Enter telemetry subscription submode, and configure the telemetry subscription.
Step 8	snsr-group <i>sgrp_id</i> sample-interval <i>interval</i> Example: <pre>switch-1(conf-tm-sub)# snsr-grp 6 sample-interval 5000 switch-1(conf-tm-sub)#</pre>	Link the sensor group to the current subscription and set the data sampling interval in milliseconds. The sampling interval determines whether the switch sends telemetry data periodically, or when environment events occur.
Step 9	dst-group <i>dgrp_id</i> Example: <pre>switch-1(conf-tm-sub)# dst-grp 33 switch-1(conf-tm-sub)#</pre>	Link the destination group to the current subscription. The destination group that you specify must match the destination group that you configured in the destination-group command.

Configuring the Resources Path for Poll for Events or Data

The resources path monitors system resources such as CPU utilization and memory utilization. You can configure this path to either periodically gather telemetry data, or when events occur. See [Polling for Data or Receiving Events, on page 373](#).

This path does not support filtering.

SUMMARY STEPS

1. **configure terminal**
2. **telemetry**
3. **sensor-group** *sgrp_id*
4. **path resources**
5. **destination-group** *grp_id*
6. **ip address** *ip_addr* **port** *port*
7. **subscription** *sub_id*
8. **snsr-group** *sgrp_id* **sample-interval** *interval*
9. **dst-group** *dgrp_id*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	configure terminal Example: <pre>switch-1# configure terminal switch-1(config)#</pre>	Enter configuration mode.
Step 2	telemetry Example: <pre>switch-1(config)# telemetry switch-1(config-telemetry)#</pre>	Enter configuration mode for the telemetry features.
Step 3	sensor-group sgrp_id Example: <pre>switch-1(config-telemetry)# sensor-group 6 switch-1(conf-tm-sensor)#</pre>	Create a sensor group for telemetry data.
Step 4	path resources Example: <pre>switch-1(conf-tm-sensor)# path resources switch-1(conf-tm-sensor)#</pre>	Configure the resources path label, which enables telemetry data for multiple individual system resources to be sent to the label. The label consolidates the multiple data inputs into one output. Depending on the sample interval, the resource data is either streaming based on the polling interval, or sent when system memory changes to Not OK.
Step 5	destination-group grp_id Example: <pre>switch-1(conf-tm-sensor)# destination-group 33 switch-1(conf-tm-dest)#</pre>	Enter telemetry destination group submode and configure the destination group.
Step 6	ip address ip_addr port port Example: <pre>switch-1(conf-tm-dest)# ip address 1.2.3.4 port 50004 switch-1(conf-tm-dest)#</pre>	Configure the telemetry data for the subscription to stream to the specified IP address and port.
Step 7	subscription sub_id Example: <pre>switch-1(conf-tm-dest)# subscription 33 switch-1(conf-tm-sub)#</pre>	Enter telemetry subscription submode, and configure the telemetry subscription.
Step 8	snsr-group sgrp_id sample-interval interval Example:	Link the sensor group to the current subscription and set the data sampling interval in milliseconds. The sampling interval determines whether the switch sends telemetry data periodically, or when resource events occur.

	Command or Action	Purpose
	<pre>switch-1 (conf-tm-sub) # snsrgroup 6 sample-interval 5000 switch-1 (conf-tm-sub) #</pre>	
Step 9	dst-group <i>dgrp_id</i> Example: <pre>switch-1 (conf-tm-sub) # dstgrp 33 switch-1 (conf-tm-sub) #</pre>	Link the destination group to the current subscription. The destination group that you specify must match the destination group that you configured in the destination-group command.

Configuring the VXLAN Path to Poll for Events or Data

The VXLAN path label provides information about the switch's Virtual Extensible LAN EVPNs, including VXLAN peers, VXLAN counters, VLAN counters, and BGP Peer data. You can configure this path label to gather telemetry information either periodically, or when events occur. See [Polling for Data or Receiving Events](#), on page 373.

This path does not support filtering.

SUMMARY STEPS

1. **configure terminal**
2. **telemetry**
3. **sensor-group** *sgrp_id*
4. **vxlan environment**
5. **destination-group** *grp_id*
6. **ip address** *ip_addr* **port** *port*
7. **subscription** *sub_id*
8. **snsr-group** *sgrp_id* **sample-interval** *interval*
9. **dst-group** *dgrp_id*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	configure terminal Example: <pre>switch-1# configure terminal switch-1 (config) #</pre>	Enter configuration mode.
Step 2	telemetry Example: <pre>switch-1 (config) # telemetry switch-1 (config-telemetry) #</pre>	Enter configuration mode for the telemetry features.

	Command or Action	Purpose
Step 3	sensor-group <i>sgrp_id</i> Example: <pre>switch-1(config-telemetry) # sensor-group 6 switch-1(conf-tm-sensor) #</pre>	Create a sensor group for telemetry data.
Step 4	vxlan environment Example: <pre>switch-1(conf-tm-sensor) # vxlan environment switch-1(conf-tm-sensor) #</pre>	Configure the VXLAN path label, which enables telemetry data for multiple individual VXLAN objects to be sent to the label. The label consolidates the multiple data inputs into one output. Depending on the sample interval, the VXLAN data is either streaming based on the polling interval, or sent when events occur.
Step 5	destination-group <i>grp_id</i> Example: <pre>switch-1(conf-tm-sensor) # destination-group 33 switch-1(conf-tm-dest) #</pre>	Enter telemetry destination group submode and configure the destination group.
Step 6	ip address <i>ip_addr</i> port <i>port</i> Example: <pre>switch-1(conf-tm-dest) # ip address 1.2.3.4 port 50004 switch-1(conf-tm-dest) #</pre>	Configure the telemetry data for the subscription to stream to the specified IP address and port.
Step 7	subscription <i>sub_id</i> Example: <pre>switch-1(conf-tm-dest) # subscription 33 switch-1(conf-tm-sub) #</pre>	Enter telemetry subscription submode, and configure the telemetry subscription.
Step 8	snsr-group <i>sgrp_id</i> sample-interval <i>interval</i> Example: <pre>switch-1(conf-tm-sub) # snsrgroup 6 sample-interval 5000 switch-1(conf-tm-sub) #</pre>	Link the sensor group to the current subscription and set the data sampling interval in milliseconds. The sampling interval determines whether the switch sends telemetry data periodically, or when VXLAN events occur.
Step 9	dst-group <i>dgrp_id</i> Example: <pre>switch-1(conf-tm-sub) # dst-grp 33 switch-1(conf-tm-sub) #</pre>	Link the destination group to the current subscription. The destination group that you specify must match the destination group that you configured in the destination-group command.

Verifying the Path Label Configuration

At any time, you can verify that path labels are configured, and check their values by displaying the running telemetry configuration.

SUMMARY STEPS

1. **show running-config-telemetry**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	show running-config-telemetry Example: <pre>switch-1(conf-tm-sensor)# show running-config telemetry !Command: show running-config telemetry !Running configuration last done at: Mon Jun 10 08:10:17 2019 !Time: Mon Jun 10 08:10:17 2019 version 9.3(1) Bios:version feature telemetry telemetry destination-profile use-nodeid tester sensor-group 4 path interface query-condition and(counters=[detailed],oper-speed=[1G,10G]) sensor-group 6 path interface query-condition oper-speed=[1G,40G] subscription 6 snsr-grp 6 sample-interval 6000 nxosv2(conf-tm-sensor)#</pre>	Displays the current running config for telemetry, In this example, sensor group 4 is configured to gather non-zero counters from interfaces running at 1 and 10 Gbps. Sensor group 6 is configured to gather all counters from interfaces running at 1 and 40 Gbps.

Displaying Path Label Information

Path Label Show Commands

Through the **show telemetry usability** commands, you can display the individual paths that the path label walks when you issue a query.

Command	Shows
show telemetry usability {all environment interface resources vxlan}	Either all telemetry paths for all path labels, or all telemetry paths for a specified path label. Also, the output shows whether each path reports telemetry data based on periodic polling or events. For the interfaces path label, also any keyword filters or query conditions you configured.
show running-config telemetry	The running configuration for telemetry and selected path information.


```

switch-1# show telemetry usability resources
1) label_name      : resources

   path_name       : sys/proc
   query_type      : poll
   query_condition  : rsp-subtree=full&rsp-foreign-subtree=ephemeral

2) label_name      : resources

   path_name       : sys/procsys
   query_type      : poll
   query_condition  :
query-target-filter=and(updated(procSysMem.memstatus),ne(procSysMem.memstatus,"OK"))

3) label_name      : resources

   path_name       : sys/procsys/systemem
   query_type      : event
   query_condition  :
query-target-filter=and(updated(procSysMem.memstatus),ne(procSysMem.memstatus,"OK"))

switch-1#

```

The following shows an example of the **show telemetry usability vxlan** command.

```

switch-1# show telemetry usability vxlan
1) label_name      : vxlan

   path_name       : sys/bd
   query_type      : poll
   query_condition  : query-target=subtree&target-subtree-class=l2VlanStats

2) label_name      : vxlan

   path_name       : sys/eps
   query_type      : poll
   query_condition  : rsp-subtree=full&rsp-foreign-subtree=ephemeral

3) label_name      : vxlan

   path_name       : sys/eps
   query_type      : event
   query_condition  : query-target=subtree&target-subtree-class=nvoDyPeer

4) label_name      : vxlan

   path_name       : sys/bgp
   query_type      : event
   query_condition  : query-target=subtree&query-target-filter=or(deleted(),created())

5) label_name      : vxlan

   path_name       : sys/bgp
   query_type      : event
   query_condition  :
query-target=subtree&target-subtree-class=bgpDn,bgpPeer,bgpPeerAf,bgpDnAf,bgpPeerAfEntry,bgpOperRtrCt1L3,bgpOperRtrP,bgpOperRtrEntry,bgpOperAfCt1

switch-1#

```


Native Data Source Paths

About Native Data Source Paths

NX-OS Telemetry supports the native data source, which is a neutral data source that is not restricted to a specific infrastructure or database. Instead, the native data source enables components or applications to hook into and inject relevant information into the outgoing telemetry stream. This feature provides flexibility because the path for the native data source does not belong to any infrastructure, so any native applications can interact with NX-OS Telemetry.

The native data source path enables you to subscribe to specific sensor paths to receive selected telemetry data. The feature works with the NX-SDK to support streaming telemetry data from the following paths:

- RIB path, which sends telemetry data for the IP routes.
- MAC path, which sends telemetry data for static and dynamic MAC entries.
- Adjacency path, which sends telemetry data for IPv4 and IPv6 adjacencies.

When you create a subscription, all telemetry data for the selected path streams to the receiver as a baseline. After the baseline, only event notifications stream to the receiver.

Streaming of native data source paths supports the following encoding types:

- Google Protobuf (GPB)
- JavaScript Object Notation (JSON)
- Compact Google Protobuf (compact GPB)

Telemetry Data Streamed for Native Data Source Paths

For each source path, the following table shows the information that is streamed when the subscription is first created (the baseline) and when event notifications occur.

Path Type	Subscription Baseline	Event Notifications
RIB	Sends all routes	Sends event notifications for create, update, and delete events. The following values are exported through telemetry for the RIB path: • Next-hop routing information: • Address of the next hop • Outgoing interface for the next hop • VRF name for the next hop • Owner of the next hop • Preference for the next hop • Metric for the next hop • Tag for the next hop • Segment ID for the next hop • Tunnel ID for the next hop • Encapsulation type for the next hop • Bitwise OR of flags for the Next Hop Type • For Layer-3 routing information: • VRF name of the route • Route prefix address • Mask length for the route • Number of next hops for the route • Event type • Next hops

Path Type	Subscription Baseline	Event Notifications
MAC	Executes a <code>GETALL</code> from DME for static and dynamic MAC entries	Sends event notifications for add, update, and delete events. The following values are exported through telemetry for the MAC path: • MAC address • MAC address type • VLAN number • Interface name • Event types Both static and dynamic entries are supported in event notifications.
Adjacency	Sends the IPv4 and IPv6 adjacencies	Sends event notifications for add, update, and delete events. The following values are exported through telemetry for the Adjacency path: • IP address • MAC address • Interface name • Physical interface name • VRF name • Preference • Source for the adjacency • Address family for the adjacency • Adjacency event type

For additional information, refer to Github <https://github.com/CiscoDevNet/nx-telemetry-proto>.

Guidelines and Limitations for Native Data Source Path

The native data source path feature has the following guidelines and limitations:

- For streaming from the RIB, MAC, and Adjacency native data source paths, sensor-path property updates do not support custom criteria like **depth**, **query-condition**, or **filter-condition**.
- Beginning with Cisco NX-OS Release 10.4(3)F, new query conditions are introduced to support sample-based subscription or updates-only support for RIB native path.

Configuring the Native Data Source Path for Routing Information

You can configure the native data source path for routing information, which sends information about all routes that are contained in the URIB. When you subscribe, the baseline sends all the route information. After the baseline, notifications are sent for route update and delete operations for the routing protocols that the switch supports. For the data sent in the RIB notifications, see [Telemetry Data Streamed for Native Data Source Paths](#), on page 389.

Before you begin

If you have not enabled the telemetry feature, enable it now (**feature telemetry**).

SUMMARY STEPS

1. **configure terminal**
2. **telemetry**
3. **sensor-group** *sgrp_id*
4. **data-source native**
5. **path rib query-condition** [**data=ephemeral** | **updates_only**]
6. **destination-group** *grp_id*
7. **ip address** *ip_addr* **port** *port* **protocol** { **HTTP** | **gRPC** } **encoding** { **JSON** | **GPB** | **GPB-compact** }
8. **subscription** *sub_id*
9. **snsr-group** *sgrp_id* **sample-interval** *interval*
10. **dst-group** *dgrp_id*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	configure terminal Example: switch-1# configure terminal switch-1(config)#	Enter configuration mode.
Step 2	telemetry Example: switch-1(config)# telemetry switch-1(config-telemetry)#	Enter configuration mode for the telemetry features.
Step 3	sensor-group <i>sgrp_id</i> Example: switch-1(conf-tm-sub)# sensor-grp 6 switch-1(conf-tm-sub)#	Create a sensor group.

	Command or Action	Purpose
Step 4	data-source native Example: <pre>switch-1(conf-tm-sensor)# data-source native switch-1(conf-tm-sensor)#</pre>	Set the data source to native so that any native application can use the streamed data without requiring a specific model or database.
Step 5	path rib query-condition [data=ephemeral updates_only] Example: <pre>nxosv2(conf-tm-sensor)# path rib nxosv2(conf-tm-sensor)#</pre> Example: <pre>nxosv2(conf-tm-sensor)# path rib query condition data=ephemeral nxosv2(conf-tm-sensor)#</pre> Example: <pre>nxosv2(conf-tm-sensor)# path rib query condition updates_only nxosv2(conf-tm-sensor)#</pre>	Configure the RIB path which streams routes and route update information. query condition data=ephemeral (optional) - You can configure sample interval 0 or other than 0. This sample interval will determine how frequently the route information is sent to the destination periodically (at the configured sample interval). query condition updates-only (optional) - Supported only for sample interval 0. With this query condition, the initial snapshot data will not be sent, only the route information updates will be sent to the destination.
Step 6	destination-group grp_id Example: <pre>switch-1(conf-tm-sensor)# destination-group 33 switch-1(conf-tm-dest)#</pre>	Enter telemetry destination group submode and configure the destination group.
Step 7	ip address ip_addr port port protocol { HTTP gRPC } encoding { JSON GPB GPB-compact } Example: <pre>switch-1(conf-tm-dest)# ip address 192.0.2.11 port 50001 protocol http encoding json switch-1(conf-tm-dest)#</pre> Example: <pre>switch-1(conf-tm-dest)# ip address 192.0.2.11 port 50001 protocol grpc encoding gpb switch-1(conf-tm-dest)#</pre> Example: <pre>switch-1(conf-tm-dest)# ip address 192.0.2.11 port 50001 protocol grpc encoding gpb-compact switch-1(conf-tm-dest)#</pre>	Configure the telemetry data for the subscription to stream to the specified IP address and port and set the protocol and encoding for the data stream.
Step 8	subscription sub_id Example: <pre>switch-1(conf-tm-dest)# subscription 33 switch-1(conf-tm-sub)#</pre>	Enter telemetry subscription submode, and configure the telemetry subscription.
Step 9	snsr-group sgrp_id sample-interval interval Example:	Link the sensor group to the current subscription and set the data sampling interval in milliseconds. The sampling

	Command or Action	Purpose
	<pre>switch-1 (conf-tm-sub) # snsr-grp 6 sample-interval 5000 switch-1 (conf-tm-sub) #</pre>	interval determines whether the switch sends telemetry data periodically, or when rib events occur. Note Depending on the sample interval, the rib sensor path streams based on the polling interval.
Step 10	dst-group <i>dgrp_id</i> Example: <pre>switch-1 (conf-tm-sub) # dst-grp 33 switch-1 (conf-tm-sub) #</pre>	Link the destination group to the current subscription. The destination group that you specify must match the destination group that you configured in the destination-group command.

Configuring the Native Data Source Path for MAC Information

You can configure the native data source path for MAC information, which sends information about all entries in the MAC table. When you subscribe, the baseline sends all the MAC information. After the baseline, notifications are sent for add, update, and delete MAC address operations. For the data sent in the MAC notifications, see [Telemetry Data Streamed for Native Data Source Paths, on page 389](#).



Note For update or delete events, MAC notifications are sent only for the MAC addresses that have IP adjacencies.

Before you begin

If you have not enabled the telemetry feature, enable it now (**feature telemetry**).

SUMMARY STEPS

1. **configure terminal**
2. **telemetry**
3. **sensor-group** *sgrp_id*
4. **data-source native**
5. **path mac**
6. **destination-group** *grp_id*
7. **ip address** *ip_addr* **port** *port* **protocol** { HTTP | gRPC } **encoding** { JSON | GPB | GPB-compact }
8. **subscription** *sub_id*
9. **snsr-group** *sgrp_id* **sample-interval** *interval*
10. **dst-group** *dgrp_id*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	configure terminal Example: <pre>switch-1# configure terminal switch-1(config)#</pre>	Enter configuration mode.
Step 2	telemetry Example: <pre>switch-1(config)# telemetry switch-1(config-telemetry)#</pre>	Enter configuration mode for the telemetry features.
Step 3	sensor-group <i>sgrp_id</i> Example: <pre>switch-1(conf-tm-sub)# sensor-grp 6 switch-1(conf-tm-sub)#</pre>	Create a sensor group.
Step 4	data-source native Example: <pre>switch-1(conf-tm-sensor)# data-source native switch-1(conf-tm-sensor)#</pre>	Set the data source to native so that any native application can use the streamed data without requiring a specific model or database.
Step 5	path mac Example: <pre>nxosv2(conf-tm-sensor)# path mac nxosv2(conf-tm-sensor)#</pre>	Configure the MAC path which streams information about MAC entries and MAC notifications.
Step 6	destination-group <i>grp_id</i> Example: <pre>switch-1(conf-tm-sensor)# destination-group 33 switch-1(conf-tm-dest)#</pre>	Enter telemetry destination group submode and configure the destination group.
Step 7	ip address <i>ip_addr</i> port <i>port</i> protocol { HTTP gRPC } encoding { JSON GPB GPB-compact } Example: <pre>switch-1(conf-tm-dest)# ip address 192.0.2.11 port 50001 protocol http encoding json switch-1(conf-tm-dest)#</pre> Example: <pre>switch-1(conf-tm-dest)# ip address 192.0.2.11 port 50001 protocol grpc encoding gpb switch-1(conf-tm-dest)#</pre> Example:	Configure the telemetry data for the subscription to stream to the specified IP address and port and set the protocol and encoding for the data stream.

	Command or Action	Purpose
	<pre>switch-1(conf-tm-dest)# ip address 192.0.2.11 port 50001 protocol grpc encoding gpb-compact switch-1(conf-tm-dest)#</pre>	
Step 8	subscription <i>sub_id</i> Example: <pre>switch-1(conf-tm-dest)# subscription 33 switch-1(conf-tm-sub)#</pre>	Enter telemetry subscription submode, and configure the telemetry subscription.
Step 9	snsr-group <i>sgrp_id</i> sample-interval <i>interval</i> Example: <pre>switch-1(conf-tm-sub)# snsr-grp 6 sample-interval 5000 switch-1(conf-tm-sub)#</pre>	Link the sensor group to the current subscription and set the data sampling interval in milliseconds. The sampling interval determines whether the switch sends telemetry data periodically, or when interface events occur.
Step 10	dst-group <i>dgrp_id</i> Example: <pre>switch-1(conf-tm-sub)# dst-grp 33 switch-1(conf-tm-sub)#</pre>	Link the destination group to the current subscription. The destination group that you specify must match the destination group that you configured in the destination-group command.

Configuring the Native Data Source Path for All MAC Information

You can configure the native data source path for MAC information, which sends information about all entries in the MAC table from Layer 3 and Layer 2. When you subscribe, the baseline sends all the MAC information. After the baseline, notifications are sent for add, update, and delete MAC address operations. For the data sent in the MAC notifications, see [Telemetry Data Streamed for Native Data Source Paths, on page 389](#).



Note For update or delete events, MAC notifications are sent only for the MAC addresses that have IP adjacencies.

Before you begin

If you have not enabled the telemetry feature, enable it now (**feature telemetry**).

SUMMARY STEPS

1. **configure terminal**
2. **telemetry**
3. **sensor-group** *sgrp_id*
4. **data-source native**
5. **path mac-all**
6. **destination-group** *dgrp_id*
7. **ip address** *ip_addr* **port** *port* **protocol** { HTTP | gRPC } **encoding** { JSON | GPB | GPB-compact }
8. **subscription** *sub_id*
9. **snsr-group** *sgrp_id* **sample-interval** *interval*

10. `dst-group dgrp_id`

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	configure terminal Example: <pre>switch-1# configure terminal switch-1(config)#</pre>	Enter configuration mode.
Step 2	telemetry Example: <pre>switch-1(config)# telemetry switch-1(config-telemetry)#</pre>	Enter configuration mode for the telemetry features.
Step 3	sensor-group sgrp_id Example: <pre>switch-1(conf-tm-sub)# sensor-grp 6 switch-1(conf-tm-sub)#</pre>	Create a sensor group.
Step 4	data-source native Example: <pre>switch-1(conf-tm-sensor)# data-source native switch-1(conf-tm-sensor)#</pre>	Set the data source to native so that any native application can use the streamed data without requiring a specific model or database.
Step 5	path mac-all Example: <pre>nxosv2(conf-tm-sensor)# path mac-all nxosv2(conf-tm-sensor)#</pre>	Configure the MAC path which streams information about all MAC entries and MAC notifications.
Step 6	destination-group grp_id Example: <pre>switch-1(conf-tm-sensor)# destination-group 33 switch-1(conf-tm-dest)#</pre>	Enter telemetry destination group submode and configure the destination group.
Step 7	ip address ip_addr port port protocol { HTTP gRPC } encoding { JSON GPB GPB-compact } Example: <pre>switch-1(conf-tm-dest)# ip address 192.0.2.11 port 50001 protocol http encoding json switch-1(conf-tm-dest)#</pre> Example: <pre>switch-1(conf-tm-dest)# ip address 192.0.2.11 port 50001 protocol grpc encoding gpb switch-1(conf-tm-dest)#</pre>	Configure the telemetry data for the subscription to stream to the specified IP address and port and set the protocol and encoding for the data stream.

	Command or Action	Purpose
	Example: <pre>switch-1 (conf-tm-dest) # ip address 192.0.2.11 port 50001 protocol grpc encoding gpb-compact switch-1 (conf-tm-dest) #</pre>	
Step 8	subscription <i>sub_id</i> Example: <pre>switch-1 (conf-tm-dest) # subscription 33 switch-1 (conf-tm-sub) #</pre>	Enter telemetry subscription submode, and configure the telemetry subscription.
Step 9	snsr-group <i>sgrp_id</i> sample-interval <i>interval</i> Example: <pre>switch-1 (conf-tm-sub) # snsr-grp 6 sample-interval 5000 switch-1 (conf-tm-sub) #</pre>	Link the sensor group to the current subscription and set the data sampling interval in milliseconds. The sampling interval determines whether the switch sends telemetry data periodically, or when interface events occur.
Step 10	dst-group <i>dgrp_id</i> Example: <pre>switch-1 (conf-tm-sub) # dst-grp 33 switch-1 (conf-tm-sub) #</pre>	Link the destination group to the current subscription. The destination group that you specify must match the destination group that you configured in the destination-group command.

Configuring the Native Data Path for IP Adjacencies

You can configure the native data source path for IP adjacency information, which sends information about all IPv4 and IPv6 adjacencies for the switch. When you subscribe, the baseline sends all the adjacencies. After the baseline, notifications are sent for add, update, and delete adjacency operations. For the data sent in the adjacency notifications, see [Telemetry Data Streamed for Native Data Source Paths, on page 389](#).

Before you begin

If you have not enabled the telemetry feature, enable it now (**feature telemetry**).

SUMMARY STEPS

1. **configure terminal**
2. **telemetry**
3. **sensor-group** *sgrp_id*
4. **data-source native**
5. **path adjacency**
6. **destination-group** *dgrp_id*
7. **ip address** *ip_addr* **port** *port* **protocol** { HTTP | gRPC } **encoding** { JSON | GPB | GPB-compact }
8. **subscription** *sub_id*
9. **snsr-group** *sgrp_id* **sample-interval** *interval*
10. **dst-group** *dgrp_id*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	configure terminal Example: <pre>switch-1# configure terminal switch-1(config)#</pre>	Enter configuration mode.
Step 2	telemetry Example: <pre>switch-1(config)# telemetry switch-1(config-telemetry)#</pre>	Enter configuration mode for the telemetry features.
Step 3	sensor-group <i>sgrp_id</i> Example: <pre>switch-1(conf-tm-sub)# sensor-grp 6 switch-1(conf-tm-sub)#</pre>	Create a sensor group.
Step 4	data-source native Example: <pre>switch-1(conf-tm-sensor)# data-source native switch-1(conf-tm-sensor)#</pre>	Set the data source to native so that any native application can use the streamed data.
Step 5	path adjacency Example: <pre>nxosv2(conf-tm-sensor)# path adjacency nxosv2(conf-tm-sensor)#</pre>	Configure the Adjacency path which streams information about the IPv4 and IPv6 adjacencies.
Step 6	destination-group <i>grp_id</i> Example: <pre>switch-1(conf-tm-sensor)# destination-group 33 switch-1(conf-tm-dest)#</pre>	Enter telemetry destination group submode and configure the destination group.
Step 7	ip address <i>ip_addr</i> port <i>port</i> protocol { HTTP gRPC } encoding { JSON GPB GPB-compact } Example: <pre>switch-1(conf-tm-dest)# ip address 192.0.2.11 port 50001 protocol http encoding json switch-1(conf-tm-dest)#</pre> Example: <pre>switch-1(conf-tm-dest)# ip address 192.0.2.11 port 50001 protocol grpc encoding gpb switch-1(conf-tm-dest)#</pre> Example:	Configure the telemetry data for the subscription to stream to the specified IP address and port and set the protocol and encoding for the data stream.

	Command or Action	Purpose
	<pre>switch-1(conf-tm-dest)# ip address 192.0.2.11 port 50001 protocol grpc encoding gpb-compact switch-1(conf-tm-dest)#</pre>	
Step 8	subscription <i>sub_id</i> Example: <pre>switch-1(conf-tm-dest)# subscription 33 switch-1(conf-tm-sub)#</pre>	Enter telemetry subscription submode, and configure the telemetry subscription.
Step 9	snsr-group <i>sgrp_id</i> sample-interval <i>interval</i> Example: <pre>switch-1(conf-tm-sub)# snsr-grp 6 sample-interval 5000 switch-1(conf-tm-sub)#</pre>	Link the sensor group to the current subscription and set the data sampling interval in milliseconds. The sampling interval determines whether the switch sends telemetry data periodically, or when interface events occur.
Step 10	dst-group <i>dgrp_id</i> Example: <pre>switch-1(conf-tm-sub)# dst-grp 33 switch-1(conf-tm-sub)#</pre>	Link the destination group to the current subscription. The destination group that you specify must match the destination group that you configured in the destination-group command.

Displaying Native Data Source Path Information

Use the NX-OS **show telemetry event collector** commands to display statistics and counters, or errors for the native data source path.

Displaying Statistics

You can issue **show telemetry event collector stats** command to display the statistics and counters for each native data source path.

An example of statistics for the RIB path:

```
switch-1# show telemetry event collector stats
```

```
-----
Row ID      Collection Count  Latest Collection Time  Sensor Path(GroupId)
-----
1           4                Mon Jul 01 13:53:42.384 PST rib(1)
switch-1#
```

An example of the statistics for the MAC path:

```
switch-1# show telemetry event collector stats
```

```
-----
Row ID      Collection Count  Latest Collection Time  Sensor Path(GroupId)
-----
1           3                Mon Jul 01 14:01:32.161 PST mac(1)
switch-1#
```

An example of the statistics for the Adjacency path:

```
switch-1# show telemetry event collector stats
```

```
-----
```

Row ID	Collection Count	Latest Collection Time	Sensor Path(GroupId)
1	7	Mon Jul 01 14:47:32.260 PST	adjacency(1)

switch-1#

Displaying Error Counters

You can use the **show telemetry event collector stats** command to display the error totals for all the native data source paths.

```
switch-1# show telemetry event collector errors
```

```
-----
-
Error Description                                     Error Count
-----
-
Dme Event Subscription Init Failures                  - 0
Event Data Enqueue Failures                          - 0
Event Subscription Failures                          - 0
Pending Subscription List Create Failures             - 0
Subscription Hash Table Create Failures              - 0
Subscription Hash Table Destroy Failures             - 0
Subscription Hash Table Insert Failures              - 0
Subscription Hash Table Remove Failures              - 0
switch-1#
```

Streaming Syslog

About Streaming Syslog for Telemetry

Beginning with Cisco NX-OS release 9.3(3), model-driven telemetry supports streaming of syslogs using YANG as a data source. When you create a subscription, all the syslogs are streamed to the receiver as a baseline. This feature works with the NX-SDK to support streaming syslog data from the following syslog paths:

- Cisco-NX-OS-Syslog-oper:syslog
- Cisco-NX-OS-Syslog-oper:syslog/messages

After the baseline, only syslog event notifications stream to the receiver. Streaming of syslog paths supports the following encoding types:

- Google Protobuf (GPB)
- JavaScript Object Notation (JSON)

Configuring the Native Data Source Path for Routing Information

You can configure the native data source path for routing information, which sends information about all routes that are contained in the URIB. When you subscribe, the baseline sends all the route information. After the baseline, notifications are sent for route update and delete operations for the routing protocols that the switch supports. For the data sent in the RIB notifications, see [Telemetry Data Streamed for Native Data Source Paths, on page 389](#).

Before you begin

If you have not enabled the telemetry feature, enable it now (**feature telemetry**).

SUMMARY STEPS

1. **configure terminal**
2. **telemetry**
3. **sensor-group** *sgrp_id*
4. **data-source native**
5. **path rib query-condition** [**data=ephemeral** | **updates_only**]
6. **destination-group** *grp_id*
7. **ip address** *ip_addr* **port** *port* **protocol** { **HTTP** | **gRPC** } **encoding** { **JSON** | **GPB** | **GPB-compact** }
8. **subscription** *sub_id*
9. **snsr-group** *sgrp_id* **sample-interval** *interval*
10. **dst-group** *dgrp_id*

DETAILED STEPS**Procedure**

	Command or Action	Purpose
Step 1	configure terminal Example: <pre>switch-1# configure terminal switch-1(config)#</pre>	Enter configuration mode.
Step 2	telemetry Example: <pre>switch-1(config)# telemetry switch-1(config-telemetry)#</pre>	Enter configuration mode for the telemetry features.
Step 3	sensor-group <i>sgrp_id</i> Example: <pre>switch-1(conf-tm-sub)# sensor-grp 6 switch-1(conf-tm-sub)#</pre>	Create a sensor group.
Step 4	data-source native Example: <pre>switch-1(conf-tm-sensor)# data-source native switch-1(conf-tm-sensor)#</pre>	Set the data source to native so that any native application can use the streamed data without requiring a specific model or database.
Step 5	path rib query-condition [data=ephemeral updates_only] Example: <pre>nxosv2(conf-tm-sensor)# path rib nxosv2(conf-tm-sensor)#</pre>	Configure the RIB path which streams routes and route update information. query condition data=ephemeral (optional) - You can configure sample interval 0 or other than 0. This sample interval will determine how frequently the route

	Command or Action	Purpose
	Example: <pre>nxosv2(conf-tm-sensor)# path rib query condition data=ephemeral nxosv2(conf-tm-sensor)#</pre> Example: <pre>nxosv2(conf-tm-sensor)# path rib query condition updates_only nxosv2(conf-tm-sensor)#</pre>	information is sent to the destination periodically (at the configured sample interval). query condition updates-only (optional) - Supported only for sample interval 0. With this query condition, the initial snapshot data will not be sent, only the route information updates will be sent to the destination.
Step 6	destination-group <i>grp_id</i> Example: <pre>switch-1(conf-tm-sensor)# destination-group 33 switch-1(conf-tm-dest)#</pre>	Enter telemetry destination group submode and configure the destination group.
Step 7	ip address <i>ip_addr</i> port <i>port</i> protocol { HTTP gRPC } encoding { JSON GPB GPB-compact } Example: <pre>switch-1(conf-tm-dest)# ip address 192.0.2.11 port 50001 protocol http encoding json switch-1(conf-tm-dest)#</pre> Example: <pre>switch-1(conf-tm-dest)# ip address 192.0.2.11 port 50001 protocol grpc encoding gpb switch-1(conf-tm-dest)#</pre> Example: <pre>switch-1(conf-tm-dest)# ip address 192.0.2.11 port 50001 protocol grpc encoding gpb-compact switch-1(conf-tm-dest)#</pre>	Configure the telemetry data for the subscription to stream to the specified IP address and port and set the protocol and encoding for the data stream.
Step 8	subscription <i>sub_id</i> Example: <pre>switch-1(conf-tm-dest)# subscription 33 switch-1(conf-tm-sub)#</pre>	Enter telemetry subscription submode, and configure the telemetry subscription.
Step 9	snsr-group <i>sgrp_id</i> sample-interval <i>interval</i> Example: <pre>switch-1(conf-tm-sub)# snsr-grp 6 sample-interval 5000 switch-1(conf-tm-sub)#</pre>	Link the sensor group to the current subscription and set the data sampling interval in milliseconds. The sampling interval determines whether the switch sends telemetry data periodically, or when rib events occur. Note Depending on the sample interval, the rib sensor path streams based on the polling interval.
Step 10	dst-group <i>dgrp_id</i> Example: <pre>switch-1(conf-tm-sub)# dst-grp 33 switch-1(conf-tm-sub)#</pre>	Link the destination group to the current subscription. The destination group that you specify must match the destination group that you configured in the destination-group command.

Telemetry Data Streamed for Syslog Path

For each source path, the following table shows the information that is streamed when the subscription is first created "the baseline" and when event notifications occur.

Path	Subscription Baseline	Event Notification
Cisco-NX-OS-Syslog-oper:syslog/messages	Stream all the existing syslogs from the switch.	Sends event notification for syslog occurred on the switch: • message-id • node-name • time-stamp • time-of-day • time-zone • category • message-name • severity • text

Displaying Syslog Path Information

Use the Cisco NX-OS **show telemetry event collector** commands to display statistics and counters, or errors for the syslog path.

Displaying Statistics

You can enter the **show telemetry event collector stats** command to display the statistics and counters for each syslog path.

The following is an example of statistics for the syslog path:

```
switch# show telemetry event collector stats
```

```
-----
Row ID      Collection Count  Latest Collection Time  Sensor Path(GroupId)
-----
1           138          Tue Dec 03 11:20:08.200 PST Cisco-NX-OS-Syslog-oper:syslog(1)

2           138          Tue Dec 03 11:20:08.200 PST
Cisco-NX-OS-Syslog-oper:syslog/messages(1)
```

Displaying Error Counters

You can use the **show telemetry event collector errors** command to display the error totals for all the syslog paths.

```
switch(config-if)# show telemetry event collector errors
```

```
-----
Error Description                                Error Count
```



```

-----
Dme Event Subscription Init Failures           - 0
Event Data Enqueue Failures                   - 0
Event Subscription Failures                    - 0
Pending Subscription List Create Failures      - 0
Subscription Hash Table Create Failures        - 0
Subscription Hash Table Destroy Failures       - 0
Subscription Hash Table Insert Failures        - 0
Subscription Hash Table Remove Failures        - 0

```

Sample JSON Output

The following is a sample of JSON output:

```

172.19.216.13 - - [03/Dec/2019 19:38:50] "POST
/network/Cisco-NX-OS-Syslog-oper%3Asyslog%2Fmessages HTTP/1.0" 200 -
172.19.216.13 - - [03/Dec/2019 19:38:50] "POST
/network/Cisco-NX-OS-Syslog-oper%3Asyslog%2Fmessages HTTP/1.0" 200 -
>>> URL           : /network/Cisco-NX-OS-Syslog-oper%3Asyslog%2Fmessages
>>> TM-HTTP-VER    : 1.0.0
>>> TM-HTTP-CNT    : 1
>>> Content-Type   : application/json
>>> Content-Length : 578
      Path => Cisco-NX-OS-Syslog-oper:syslog/messages
              node_id_str   : task-n9k-1
              collection_id : 40
              data_source    : YANG
              data           :
[
  [
    {
      "message-id": 420
    },
    {
      "category": "ETHPORT",
      "group": "ETHPORT",
      "message-name": "IF_UP",
      "node-name": "task-n9k-1",
      "severity": 5,
      "text": "Interface loopback10 is up ",
      "time-of-day": "Dec 3 2019 11:38:51",
      "time-stamp": "1575401931000",
      "time-zone": ""
    }
  ]
]

```

.

Sample KVGPB Output

The following is a sample KVGPB output.

```

KVGPB Output:
---Telemetry msg received @ 18:22:04 UTC

```

```
Read frag:1 size:339 continue to block on read..
All the fragments:1 read successfully total size read:339
node_id_str: "task-n9k-1"
subscription_id_str: "1"
collection_id: 374
data_gpbkv {
  fields {
    name: "keys"
    fields {
      name: "message-id"
      uint32_value: 374
    }
  }
  fields {
    name: "content"
    fields {
      fields {
        name: "node-name"
        string_value: "task-n9k-1"
      }
      fields {
        name: "time-of-day"
        string_value: "Jun 26 2019 18:20:21"
      }
      fields {
        name: "time-stamp"
        uint64_value: 1574293838000
      }
      fields {
        name: "time-zone"
        string_value: "UTC"
      }
    }
  }
}
```

```
fields {  
    name: "process-name"  
    string_value: ""  
}  
fields {  
    name: "category"  
    string_value: "VSHD"  
}  
fields {  
    name: "group"  
    string_value: "VSHD"  
}  
fields {  
    name: "message-name"  
    string_value: "VSHD_SYSLOG_CONFIG_I"  
}  
fields {  
    name: "severity"  
    uint32_value: 5  
}  
fields {  
    name: "text"  
    string_value: "Configured from vty by admin on console0"  
}  
}  
}
```

•

Additional References

Related Documents

Related Topic	Document Title
Example configurations of telemetry deployment for VXLAN EVPN.	Telemetry Deployment for VXLAN EVPN Solution



付録 **A**

ストリーミング テレメトリの送信元

- [ストリーミング テレメトリについて \(409 ページ\)](#)
- [注意事項と制約事項 \(409 ページ\)](#)
- [テレメトリで利用可能なデータ \(409 ページ\)](#)

ストリーミング テレメトリについて

Cisco Nexus スイッチのストリーミングテレメトリ機能は、ネットワークからデータを継続的にストリーミングしてクライアントに通知し、モニタリング データへのほぼリアルタイムのアクセスを提供します。

注意事項と制約事項

ストリーミング テレメトリのガイドラインと制限は次のとおりです。

- テレメトリ機能は、Cisco Nexus スイッチで使用できます。
- メモリが 8 GB 未満のスイッチは、テレメトリをサポートしていません。[Cisco Nexus 3500-XL スイッチは 16 GB のメモリを備えているため、テレメトリをサポートしています。(The Cisco Nexus 3500-XL switch has 16 GB of memory and therefore supports telemetry.)]

テレメトリで利用可能なデータ

コンポーネント グループごとに、[NX-API DME モデル リファレンス](#)の付録にある識別名 (DN) は、一覧表示されたプロパティをテレメトリのデータとして提供できます。



INDEX

B

bash [5, 7](#)
 アクセス [5](#)
 機能 bash シェル [5](#)
 例 [7](#)
Bourne-Again シェル () , *See* Bash

F

feature grpc [281](#)

G

grpc certificate [281](#)
grpc gnmi max-concurrent-call [281–282](#)
grpc port [281](#)

N

NX-API [145–148, 151, 153, 163, 179, 183, 190](#)
 CLI [148](#)
 Cookie [147](#)
 応答コード [179](#)
 応答要素 [163](#)
 セキュリティ [147](#)
 トランスポート層 [146](#)
 管理コマンド [151](#)
 メッセージ形式 [146](#)
 ユーザ インターフェイス [183, 190](#)
 リクエスト要素 [153](#)

P

Python [81–83, 85, 87–90](#)
 Cisco パッケージ [82](#)
 APIについて [81](#)
 CLI コマンド API [83](#)
 embedded event manager (EEM) [88](#)
 NX-OS セキュリティ [90](#)
 NX-OS ネットワーク インターフェイス [89](#)
 スクリプト [82](#)
 non-interactive [87](#)
 表示フォーマットの例 [85](#)
 呼び出し [85](#)

T

tcl [95–97, 100](#)
 cli コマンド [96](#)
 tclquit コマンド [97](#)
 オプション [97](#)
 コマンドの区切り [97](#)
 references [100](#)
 サンドボックス [97](#)
 セキュリティ [97](#)
 対話型ヘルプがない [95](#)
 タブ補完 [96](#)
 変数 [97](#)
 history [96](#)
Tool Command Language, *See* tcl

翻訳について

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