NX-OS ソフトウェアのオプション

- Cisco NX-OS ソフトウェアのオプション （1 ページ）
- モジュラ パッケージ （3 ページ）
- NX-OS イメージ ブート モード （4 ページ）
- Red Hat パッケージ マネージャ （4 ページ）
- Dandified YUM コマンド （17 ページ）
- FTP サーバーの構成とローカル FTP YUM リポジトリのセットアップ （26 ページ）
- インストール操作用ユーザー ロールの作成 （30 ページ）

Cisco NX-OS ソフトウェアのオプション

選択性は、NX-OSソフトウェアの機能の一つで、

- モジュール式パッケージを使用して選択的な機能アップグレードを行い、
- 基本モードとフルモードの両方をサポートし、
- サービスの中断なしにオプションのRPMを個別にアップグレードまたは削除できます。

NX-OS リリース 9.2 (1) 以降では、NX-OS ソフトウェア イメージでモジュラ パッケージの管理をサポートします。これにより、NX-OS ソフトウェアは、基礎となる NX-OS ソフトウェアを変更することなく、機能を選択的に追加、削除、およびアップグレードする柔軟性を提供します。

モジュール型 NX-OSソフトウェアを使用すると、次のような利点があります：

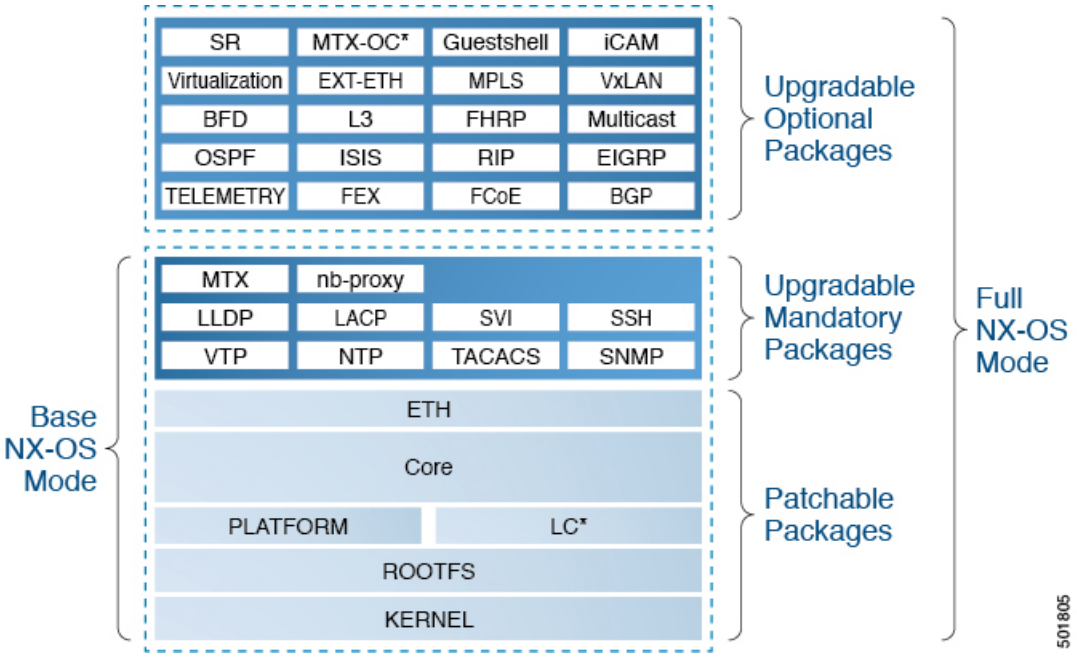
- よりスリムな NX-OS ソフトウェア
- 機能と修正の非同期配信：リリースとは独立した迅速な修正と新機能を提供します
- 実行時のバイナリとライブラリのフットプリントの削減

モード

NX-OS ソフトウェアは、図に示すように 2 つのモードで NX-OS ソフトウェアをブートするようにプロビジョニングされています：

- ベース NX-OS モード
- フル NX-OS モード

図 *1: NX-OS* ソフトウェアのオプション



- ベース NX-OS モードには次が含まれます。
  - アップグレード可能な必須パッケージ
  - パッチ適用可能パッケージ
- フル NX-OS モードには次が含まれます。
  - アップグレード可能なオプション パッケージ
  - アップグレード可能な必須パッケージ
  - パッチ適用可能パッケージ

（注） デフォルトのモードは、フル NX-OS モードです。

ベース NX-OS モードでは、レイヤ 2 およびレイヤ 3 の基本的な機能が提供されます。すべてのダイナミック ルーティング機能（BGP、OSPF、EIGRP、RIP、ISIS など）やその他のオプション機能RPMはデフォルトでは使用できません。オプションの機能RPMは、ベースイメージの上にインストールする必要があります。

フル NX-OS モードでは、ブート時にイーサネット プラグインがプラグイン マネージャによりアクティブ化されるときにすべての機能 RPM がインストールされます。以前のリリースと比較して、ユーザの動作に変更はありません。

モジュラ パッケージ

モジュラパ ッケージは、

- 同じリリース内で個別のアップグレードを可能にし、
- システムのスタートアップやその他の機能に影響を与えずにランタイムを削除できるようにし、そして
- パッケージのインストール後にのみ機能 API を使用する必要があるソフトウェア パッケージです。

NX-OS ソフトウェア イメージは、以前から Linux ディストリビューションを形成するパッケージングで構成されています。各パッケージのサイズが大きいため、特定のパッケージのアップグレードが困難になっています。このセクションでは、NX-OS ソフトウェア イメージの新しいパッケージの管理について説明します。NX-OS リリース 9.2（1）以降では、BGP、OSPF、VXLAN、MPLS、セグメント ルーティングなどの一部の NX-OS 機能はオプションと見なされます。

各モジュラ パッケージには、次の重要な特徴があります。

- アップグレード機能：モジュラ パッケージは個別にアップグレード可能です。モジュラ パッケージは、同じリリースのものを使用する必要があります。複数のリリースにまたがるパッケージでのアップグレードの実行はサポートされていません。
- オプション性：モジュラパッケージはオプションです。たとえば、これらのパッケージは実行時に削除またはアンインストールが可能です。モジュラパッケージの削除はシステムの稼働に影響を与えず、スイッチのその他の機能にも影響を与えません。

（注） モジュラ パッケージでエクスポートされたすべての API は、機能のインストール後にのみ使用する必要があります。

RPM と DNF

RPM（Red Hat Package Manager）は、Linux Standard Base（LSB）内のパッケージングに使用されるパッケージ管理システムです。RPM コマンド オプションは、次の 3 つのサブグループにまとめられます。

- パッケージのクエリと確認
- パッケージのインストール、アップグレードおよび削除
- その他の機能の実行

rpm は RPM で使用されるメイン コマンドのコマンド名です。一方、.rpm は RPM ファイルに使用される拡張子です。

洗練された YUM（Yellowdog Updater, Modified）または DNF は、RPM ベース Linux システム用のオープン ソース コマンド ライン ツールです。これにより、ユーザとシステム管理者はシステム上のソフトウェア パッケージのインストール、アップデート、削除、または検索を簡単に行うことができます。 DNF により、自動アップデートとパッケージ管理（ 依存関係管理を含む）の機能が RPM システムに追加されます。 DNF は、システムにインストールされたパッケージを把握するだけではなく、パッケージのコレクションであるリポジトリと連携します。通常、リポジトリにはネットワーク接続を介してアクセスできます。

NX-OS イメージ ブート モード

NX-OS イメージは、ベース モードまたはフル モードでブートできます。フル ブート モードでは、以前のリリースのソフトウェアと同様な完全な NX-OS ソフトウェアがインストールされます。これは、デフォルトのブート モードです。ベース ブート モードでは、オプションの RPM はインストールされません。

コマンド ライン オプションを使用するには、モードに応じて次の手順のいずれかを実行します。

- **install reset nxos base** オプションを VSH プロンプトで使用して、NX-OS イメージを基本ブート モードでインストールします。リロード後にスイッチはベース モードになり、オプション パッケージはインストールされません。
- **install reset nxos full** オプションを VSH プロンプトで使用して、NX-OS イメージをフル ブート モードでインストールします。リロード後にスイッチはフル モードになり、オプションのパッケージが自動的にインストールされます。

詳細については、「機能 *RPM* の操作にインストール コマンドを使用する」セクションを参照してください。

Red Hat パッケージ マネージャ

NX-OS インストール コマンドまたは、DNF コマンドを使用して Red Hat パッケージ マネージャ（RPM）を新しいソフトウェア バージョンへアップグレードまたは、ダウングレードできます。アップグレード可能な RPM には、オプションと必須があります。

（注） NX-OS の起動プロセス中、イメージ抽出段階が行われている間、署名された RPM はメモリに残ります。ただし、この方法はメモリ消費の点で最も効率的ではありません。NX-OS リリース 10.4（3）F では、システムが安定した状態になり、十分な SSD スペースにアクセスできるようになると、RPM はメモリから永続ストレージに転送されます。この機能は、N9K-C92348GC-X およびすべての Nexus 9300 TOR スイッチでサポートされています。

オプションおよび必須の RPM の詳細については、以降のセクションを参照してください。

RPM の形式

ここでは、NX-OS機能の RPM ファイルの一般的な形式と命名規則について説明します。

RPM の一般的な形式は、<name>-<version>-<release>.<arch>.rpm です。同じ形式が NX-OS 機能 RPM にも適用されます。

- name：パッケージ名（例：BGP）
- version（< X.y.x.b > 形式）：< major.minor.patch.build_number >（例：2.0.1.0）
- release：RPM 作成元のブランチ（例：9.2.1）
- arch：RPM のアーキテクチャ タイプ（例：lib32_n9000）

この表は、例えば fex-2.0.0.0-9.2.1.lib32_n9000.rpm のような命名規則に関する詳細情報を提供します。

表 *1 : RPM* 命名規則

RPM 命名規則 例：**fex-2.0.0.0-9.2.1.lib32_n9000.rpm**	**説明**
fex	コンポーネントの名前を示しています。
2	RPM に後方互換性がないことを示します。アップグレード中に設定の損失が発生します。
0	後方互換性がある増分API変更/コマンド変更/スキーマ変更を示します。既存の機能上の新しい機能が該当します。アップグレード中に失われる設定はありません。
0	機能の変更がないバグ修正を示します。アップグレード中に失われる設定はありません。
0	この番号は、リリースの開発サイクルの間にコンポーネントが変更された回数を追跡します。この値はすべてのリリース イメージで 0 となります。
9.2.1	RPMのリリース番号またはディストリビューション バージョンを示します。NVR 形式に沿っています。機能 RPM は NX-OS リリースにのみ適用可能であるため、このフィールドには存在する NX-OS リリース バージョンのみが指定されます。

RPM 命名規則 例：**fex-2.0.0.0-9.2.1.lib32_n9000.rpm**	説明
lib32_n9000	RPM のアーキテクチャ タイプを示します。

オプション RPM とその関連機能

オプション RPM をインストールすると、ネイティブの NXOS 動作に影響を与えずに機能を有効化できます。また、オプション RPM は、 **install deactivate** コマンドを使用してスイッチから削除できます。

EIGRP などのオプション RPM は、基本ソフトウェアの一部ではありません。これらの RPM は、**dnf** または **install** コマンドを使用して、必要に応じてスイッチに対して追加、アップグレード、削除が可能です。

このテーブルは、オプションの RPM およびそれの関連機能のリストを含みます。

表 *2:* オプションの *RPM* とその関連機能のリスト

パッケージ名	関連機能
アプリケーションのホスティング	feature app-hosting
BGP	feature bgp
BFD	feature bfd
Container-tracker	feature container-tracker
EIGRP	feature eigrp
Ext-Eth	• feature openflow • feature evb • feature imp • feature netflow • feature sla_sender • feature sla_responder • feature sla twamp-server • feature sflow
EXT_ETH_LOWMEM	• feature evb • feature netflow

パッケージ名	関連機能
FCoE	• feature-set fcoe • feature-set fcoe-npv
FEX	feature-set fex
FHRP	• feature hsrp • feature vrrpv3
HW TELEMETRY	feature hw telemetry
iCAM	feature icam
ISIS	feature isis
MPLS	• feature mpls segment-routing • feature mpls evpn
マルチキャスト	• feature pim • feature pim6 • feature msdp • feature ngmvpn
NIA	N/A
NXSDK	N/A
OSPF	• feature ospf • feature ospfv3
RIP	feature rip
SDAA	N/A
サービス	feature catena
SR	feature mpls segment-routing traffic-engineering
TELEMETRY	feature telemetry
仮想化	該当なし
VM トラッカー	機能 vmtracker
VXLAN	• feature nv overlay • feature fabric forwarding

NX-OS 機能 RPM インストールに関するガイドライン

NX-OS システム RPM レポジトリは、RPM 管理用に NX-OS シリーズ スイッチで存在します。

（注） RPM をシステムのリポジトリに手動でコピーをすることは避けてください。代わりに、install または DNF コマンドを使用してください。

表 *3:* スイッチで存在する *RPM* リポジトリ。

リポジトリ名	リポジトリのパス	説明
groups-repo	/rpms	バンドルされている NX-OS イメージの一部です。NX-OS イメージの一部としてバンドルされているすべての RPM を保持するために使用されます。このリポジトリに格納されているすべての RPM は、基本 RPM と呼ばれます。

リポジトリ名	リポジトリのパス	説明
localdb	/bootflash/.rpmstore/patching/localrepo	RPM の保持に使用されます。ユーザが NX- OS機能 RPM を **install add** コマンドの一部として追加すると、RPM がこの場所にコピーされ、リロード時にも持続します。ユーザは、リポジトリをクリーンアップする責任があります。 このリポジトリに RPM を追加するには、 **install add** コマンドを使用します。 このリポジトリから RPM を削除するには、 **install remove** コマンドを使用します。 DNF コマンドも、リポジトリに追加するために使用できます。 Nexus 3000 シリーズ スイッチを除き、リポジトリの最大領域は 200 Mb です（Nexus 9000 シリーズ スイッチのパッチ リポジトリを含む）。Nexus 3000 シリーズ スイッチでは、リポジトリの最大サイズは 20 Mb です。
patching	/bootflash/.rpmstore/patching/patchrepo	RPM の保持に使用されます。ユーザが NX-OS パッチ RPM をスイッチに追加すると、パッチ RPM がこのリポジトリにコピーされます。
Third_Party	/bootflash/.rpmstore/thirdparty	ユーザがサードパーティ RPM を追加したときに、RPM の保持に使用されます。

`groups-repo` と `localdb` リポジトリには、システム ブート時またはアクティベーション時にインストールする必要がある NX-OS 機能 RPM が保持されます。DNF コマンドまたは、**install** コマンドは、これらの RPM のインストールまたは削除に使用できます。

記載されているルールが、ブートまたはインストール時の機能 RPM のインストール手順に適用されます：

- 同じ NX-OS リリース番号の RPM のみをインストール用に選択する必要があります。
- ベース RPM は、localdb リポジトリに追加できません。

サードパーティ RPM のインストール向け注意事項

10.1（x）より前のリリースでは、Cisco によって提供または署名されていない場合でも、サードパーティのパッケージをデバイスにインストールできます。

リリース10.1（x）以降、シスコによって署名されていないサードパーティパッケージは、デバイスにインストールできません。ただし、これをバイパスしてソフトウェアをインストールする場合は、サードパーティ製ソフトウェアのインストールを有効にするようにデバイスを設定できます。構成は通常の構成として保持され、**running-config** コマンドを使用します。この設定に従って、既知のリスクがあるサードパーティ製ソフトウェアをインストールできます。

機能とサードパーティ RPM のコマンドオプションをインストールします

機能 RPM 操作におけるインストール コマンドの使用方法については、参照表を参照してください。

表 *4: RPM* 機能のインストールコマンドに関するリファレンス

コマンド	説明

install reset	この操作は、すべてのパッチ、保持されたコンフィグレーション、アップグレードされたパッケージ、-サードパーティのインストール済みパッケージ、未保存のコンフィグレーションを削除し、デフォルトのパッケージを使用してスイッチの以前のモード（フル/基本）をリロードします。
	install reset コマンドも write erase 操作を実行します。次のメッセージがプロンプトに表示されます。
	`switch(config)# install reset`
	警告!!この操作により、すべてのパッチ、アップグレードされたパッケージ、永続化された設定など、インストールされているサードパーティ製パッケージ、起動設定（書き込み消去）が削除され、スイッチがデフォルトのパッケージで再ロードされます。
	`Do you want to proceed with reset operation? (y/n)? [n]`

コマンド	説明
install reset nxos base	この操作は、すべてのパッチ、アップグレードされたパッケージ、保持された etc コンフィグレーション、-サードパーティのインストール済みパッケージ、スタートアップ設定（write erase）を削除して NX-OS をベース モードでインストールして、デフォルトのパッケージを使用してスイッチをリロードします。
install reset nxos full	この操作は、すべてのパッチ、アップグレードされたパッケージ、保持された etc コンフィグレーション、-サードパーティのインストール済みパッケージ、スタートアップ設定（write erase）を削除して NX-OS をフル モードでインストールして、デフォルトのパッケージ（必須およびオプションの RPM による）を使用してスイッチをリロードします。
install add <>	それぞれのリポジトリに RPM ファイルを追加して、次のリポジトリを更新します（**patch/feature/third-party**）。
install activate *<rpm name>*	リポジトリに存在する RPM をインストールします。
install commit *<rpm name>*	パッチ RPM に使用します。リロード時にパッチを保持します。
install deactivate *<rpm name>*	RPM をアンインストールします。 NX-OS リリース 10.1（1）以降では、このコマンドを使用して RPM を非アクティブ化すると、RPM の基本バージョンにダウングレードするオプションまたは RPM をアンインストールするオプションが表示されます。必要なオプションを選択すると、操作が続行されます。
install remove *<rpm name>*	リポジトリから RPM ファイルを削除してリポジトリを更新します。
sh install active	ベース rootfs RPM 以外の、システムにインストールされている RPM のリストを表示します（機能/パッチ/サードパーティ）。
sh install inactive	リポジトリに保持されていてインストールされていない RPM のリストを表示します。

コマンド	説明
sh install packages	Rootfs RPM を含む、インストールされているすべての RPM をリストします。
[no] system software allow third-party	NX-OS リリース 10.1（1）以降では、デフォルトでは、サードパーティ製RPMをデバイスにインストールできません。このコマンドは、この制限をバイパスし、サードパーティ製ソフトウェアのインストールを有効にするようにデバイスを設定します。 次のコマンドは、サードパーティ コンフィギュレーションを適用せずにサードパーティ RPM をアクティブにした場合のエラー メッセージを示しています。 `switch(config)# **install activate pbwMonitor-1.0-1.5.0.x86_64.rpm** インストール操作193は、パッケージが Cisco によって署名されていないため失敗しました。「system software allow third-party」CLI を使用して TPS のインストールを有効にしてください（2020年11月17日火曜日 04:23:10）。` 次のコマンドは、設定適用後のサードパーティ製RPMインストールのアクティブ化を示しています。 `switch(config)# **system software allow third-party** switch(config)# 2020 Nov 17 04:25:41 switch %$ VDC-1 %$ %USER-2-SYSTEM_MSG: <<%PATCH-INSTALLER-2-TPS_FEATURE_ENABLED>> User has enabled TPS installation - patch_installer switch(config)# **install activate pbwMonitor-1.0-1.5.0.x86_64.rpm** [####################] 100% Install operation 194 completed successfully at Tue Nov 17 04:25:58 2020` 次のコマンドは、サードパーティ コンフィギュレーションの無効化を示しています。 `switch(config)# **no system software allow third-party** switch(config)# 2020 Nov 17 04:27:17 switch %$ VDC-1 %$ %USER-2-SYSTEM_MSG: <<%PATCH-INSTALLER-2-TPS_FEATURE_DISABLED>> User has disabled TPS installation - patch_installer`

（注） ISSU を使用している場合、または以前のバージョンから NX-OS リリース 10.1.1 リリースにアップグレードしている場合は、アップグレード後最初の 30 分以内にサードパーティの設定を手動で適用して、サードパーティの RPM をインストールする必要があります。

デジタル署名サポートのインストール コマンドを使用します

デジタル署名サポートのインストール コマンドを使用します

手順

ステップ 1 **install add bootflash**:<keyfile> **gpg-key** コマンドを使用して、ブートフラッシュにあるファイルから GPG（GNU Privacy Guard）キーをインポートして追加します。

例：

```
switch# install add bootflash:RPM-GPG-KEY-puppetlabs gpg-key [##########################] 100% インストール操作 304 が正常に完了しました（木 19 16 日） :40:28 2018
```

GPG（ NU Privacy Guard）キーを使用してリリース RPM に署名します。GPG 公開キーは **/etc/pki/rpm-gpg/arm-Nexus9k-rel.gpg**にあります。異なるソースからその他の公開キーを追加するには、このセクションの手順を使用してください。

ステップ 2 RPM ファイルが署名済みか未署名であるかどうかを確認するには、2 つのステップのいずれかを使用します。

a) **install verify package** <package-name> コマンドを使用して、パッケージが署名付きファイルであることを確認します。

b) **install verify bootflash**:<RPM file> コマンドを使用して、RPM ファイルが署名済みファイルであることを確認します。

例：

```
switch# install verify bootflash:vxlan-2.0.0.0-9.2.1.lib32_n9000.rpm RSA signed switch#
```

すべてのインストール済みの RPM のクエリ

インストール済みのすべての RPMをクエリには、次のステップを実行します

手順

次を使用して、インストール済みのすべての RPM をクエリします。 **show install packages** コマンドを使用します。

例：

```
switch# show install packages Boot Image: NXOS Image: bootflash:/nxos.9.2.1.bin
------------------------------------------------- Installed Packages attr.x86_64 2.4.47-r0.0
installed Unsigned aufs-util.x86_64 3.14+git0+b59a2167a1-r0.0 installed Unsigned base-files.n9000
3.0.14-r89.0 installed Unsigned base-passwd.lib32_x86 3.5.29-r0.1.0 installed Unsigned bash.lib32_x86
 4.3.30-r0.0 installed Unsigned bfd.lib32_n9000 2.0.0.0-9.2.1 installed Signed bgp.lib32_n9000
2.0.0.0-9.2.1 installed Signed binutils.x86_64 2.25.1-r0.0 installed Unsigned bridge-utils.x86_64
1.5-r0.0 installed Unsigned busybox.x86_64 1.23.2-r0.0 installed Unsigned busybox-udhcpc.x86_64
1.23.2-r0.0 installed Unsigned bzip2.x86_64 1.0.6-r5.0 installed Unsigned ca-certificates.all
20150426-r0.0 installed Unsigned cgroup-lite.x86_64 1.1-r0.0 installed Unsigned chkconfig.x86_64
1.3.58-r7.0 installed Unsigned container-tracker.lib32_n9000 2.0.0.0-9.2.1 installed Signed
containerd-docker.x86_64 0.2.3+gitaa8187dbd3b7ad67d8e5e3a15115d3eef43a7ed1-r0.0 installed Unsigned
 core.lib32_n9000 2.0.0.0-9.2.1 installed Signed coreutils.lib32_x86 8.24-r0.0 installed Unsigned
cpio.x86_64 2.12-r0.0 installed Unsigned cracklib.lib32_x86 2.9.5-r0.0 installed Unsigned
cracklib.x86_64 2.9.5-r0.0 installed Unsigned createrepo.x86_64 0.4.11-r9.0 installed Unsigned
cronie.x86_64 1.5.0-r0.0 installed Unsigned curl.lib32_x86 7.60.0-r0.0 installed Unsigned db.x86_64
 6.0.30-r0.0 installed Unsigned dbus-1.lib32_x86 1.8.20-r0.0 installed Unsigned dhcp-client.x86_64
 4.3.2-r0.0 installed Unsigned dhcp-server.x86_64 4.3.2-r0.0 installed Unsigned switch#
```

1 ステップ手順による RPM のインストール

RPM をインストールするコマンドとアップグレードする CLI は同じです。この 1 ステップ手順を活用 RPM をインストールします。

手順

ステップ 1 **install add** *<rpm>* **activate** コマンドを使用して、RPM をインストールしてアクティブ化します。

例：

```
switch# install add bootflash:chef.rpm activate Adding the patch (/chef.rpm) [####################]
 100% Install operation 868 completed successfully at Tue May 8 11:20:10 2018 Activating the patch
 (/chef.rpm) [####################] 100% Install operation 869 completed successfully at Tue May 8
 11:20:20 2018
```

ステップ 2 **show install active** コマンドの出力を確認します。

例：

```
switch# show install active Boot Image: NXOS Image: bootflash:/nxos.9.2.1.bin Active Packages:
bgp-2.0.1.0-9.2.1.lib32_n9000 chef-12.0.0alpha.2+20150319234423.git.1608.b6eb10f-1.el5.x86_64 Active
 Base Packages: lacp-2.0.0.0-9.2.1.lib32_n9000 lldp-2.0.0.0-9.2.1.lib32_n9000
mtx-device-2.0.0.0-9.2.1.lib32_n9000 mtx-grpc-agent-2.0.0.0-9.2.1.lib32_n9000
mtx-infra-2.0.0.0-9.2.1.lib32_n9000 mtx-netconf-agent-2.0.0.0-9.2.1.lib32_n9000
mtx-restconf-agent-2.0.0.0-9.2.1.lib32_n9000 mtx-telemetry-2.0.0.0-9.2.1.lib32_n9000
ntp-2.0.0.0-9.2.1.lib32_n9000 nxos-ssh-2.0.0.0-9.2.1.lib32_n9000 snmp-2.0.0.0-9.2.1.lib32_n9000
svi-2.0.0.0-9.2.1.lib32_n9000 tacacs-2.0.0.0-9.2.1.lib32_n9000 vtp-2.0.0.0-9.2.1.lib32_n9000
```

2 ステップ手順による RPM のインストール

RPM をインストールするコマンドとアップグレードする CLI は同じです。この 2 ステップ手順を活用 RPM をインストールします。

手順

ステップ 1 **install add** <*rpm*> コマンドを使用して RPM をインストールします。

例：

```
switch# install add bootflash:vxlan-2.0.1.0-9.2.1.lib32_n9000.rpm [####################] 100% Install
 operation 892 completed successfully at Thu Jun 7 13:56:38 2018
```

ステップ 2 **show install inactive** コマンドを使用して確認します。

例：

```
switch(config)# show install inactive | grep vxlan vxlan-2.0.1.0-9.2.1.lib32_n9000
```

ステップ 3 Activate the RPM using the **install activate** <*rpm*> command.

例：

```
switch# install activate vxlan [####################] 100% Install operation 891 completed successfully
 at Thu Jun 7 13:53:07 2018
```

ステップ 4 **show install active** コマンドを使用して確認します。

例：

```
switch# show install active | grep vxlan vxlan-2.0.0.0-9.2.1.lib32_n9000 switch# show install inactive
 | grep vxlan switch#
```

RPM のアップグレード

RPM をインストールするコマンドとアップグレードする CLI は同じです。RPM をアップグレードするには、次の手順を実行します：

手順

ステップ 1 **install add** <*rpm*>**activate upgrade**コマンドを使用して RPM をインストールします。

例：

```
switch(config)# install add bootflash:bgp-2.0.2.0-9.2.1.lib32_n9000.rpm activate upgrade Adding the
 patch (/bgp-2.0.2.0-9.2.1.lib32_n9000.rpm) [####################] 100% Install operation 870
completed successfully at Tue May 8 11:22:30 2018 Activating the patch
(/bgp-2.0.2.0-9.2.1.lib32_n9000.rpm) [####################] 100% Install operation 871 completed
successfully at Tue May 8 11:22:40 2018
```

ステップ 2 **show install active** コマンドを使用して出力を確認します。

例：

```
switch(config)# show install active Boot Image: NXOS Image: bootflash:/nxos.9.2.1.bin Active Packages:
 bgp-2.0.2.0-9.2.1.lib32_n9000 chef-12.0.0alpha.2+20150319234423.git.1608.b6eb10f-1.el5.x86_64 Active
 Base Packages: lacp-2.0.0.0-9.2.1.lib32_n9000 lldp-2.0.0.0-9.2.1.lib32_n9000
mtx-device-2.0.0.0-9.2.1.lib32_n9000 mtx-grpc-agent-2.0.0.0-9.2.1.lib32_n9000
mtx-infra-2.0.0.0-9.2.1.lib32_n9000 mtx-netconf-agent-2.0.0.0-9.2.1.lib32_n9000
mtx-restconf-agent-2.0.0.0-9.2.1.lib32_n9000 mtx-telemetry-2.0.0.0-9.2.1.lib32_n9000
ntp-2.0.0.0-9.2.1.lib32_n9000 nxos-ssh-2.0.0.0-9.2.1.lib32_n9000 snmp-2.0.0.0-9.2.1.lib32_n9000
svi-2.0.0.0-9.2.1.lib32_n9000 tacacs-2.0.0.0-9.2.1.lib32_n9000 vtp-2.0.0.0-9.2.1.lib32_n9000
```

RPM のダウングレード

ダウングレード手順では、特別なコマンド属性を必要とします。1 ステップ手順を使用して RPM をダウングレードします。

手順

ステップ 1 **install add** *<rpm>***activate ダウングレード** コマンドを使用して、RPM をダウングレードします。

例：

```
switch(config)# install add bootflash:bgp-2.0.1.0-9.2.1.lib32_n9000.rpm activate ダウングレード パッチ
の追加（/bgp-2.0.1.0-9.2.1.lib32_n9000.rpm）[###### ####################] インストール操作 872 が 100% 正
常に完了しました：2018 年 5 月 8 日 11:24:43 火曜日 11:24:43 パッチのアクティブ化
(/bgp-2.0.1.0-9.2.1.lib32_n9000.rpm) ） [########################] 2018 年 5 月 8 日（火曜日） 11:24:52
 に 100% インストール操作 873 が正常に完了しました
```

ステップ 2 **show install active** コマンドを使用して出力を確認します。

例：

```
switch(config)# show install active Boot Image: NXOS Image: /nxos.9.2.1.bin Active Packages:
bgp-2.0.1.0-9.2.1.lib32_n9000 Chef-12.0.0alpha.2+20150319234423 ギット.1608.b6eb10f-1.el5.x86_64 ア
クティブベースパッケージ：lacp-2.0.0.0-9.2.1.lib32_n9000 lldp-2.0.0.0-9.2.1.lib32_n9000
mtx-device-2.0.0.0-9.2.1.lib32_n9000 mtx-grpc-agent-2.0.0.0-9.2.1.lib32_n9000
mtx-infra-2.0.0.0-9.2.1.lib32_n9000 mtx-netconf-agent-2.0.0.0-9.2.1.lib32_n9000 mtx-restconf-agent-2.0
 .0.0-9.2.1.lib32_n9000 mtx-telemetry-2.0.0.0-9.2.1.lib32_n9000 ntp-2.0.0.0-9.2.1.lib32_n9000
nxos-ssh-2.0.0.0-9.2.1.lib32_n9000 snmp-2.0.0.0 -9.2.1.lib32_n9000 svi-2.0.0.0-9.2.1.lib32_n9000
tacacs-2.0.0.0-9.2.1.lib32_n9000 vtp-2.0.0.0-9.2.1.lib32_n9000 switch(config)#
```

RPM のアンインストール

RPM をアンインストールするには、次の手順を実行します。

手順

groups-repo（/ rpms）に RPM がある場合は、RPM の基本バージョンにダウングレードします。または、**install deactivate** *<rpm>* コマンドを使用して RPM をスイッチから完全にアンインストールします。

- 基本バージョンにダウングレードするには、**y** と入力します。
- RPM を完全にアンインストールするには、コマンド プロンプトで **n** と入力します。

例：

```
switch(config)# install deactivate bgp Base RPM found. Do you want to downgrade to base version(y/n)
 [n] y Downgrading to the base version [####################] 100% Install operation 190 completed
 successfully at Tue Nov 17 04:10:40 2020
```

例：

```
switch(config)# install deactivate bgp Base RPM が見つかりました。Do you want to downgrade to base
version(y/n) [n] n ================================================================================
 WARNING!! この操作により、正常に完了すると、実行コンフィギュレーションから「bgp-3.0.0.0-9.4.1.lib32_n9000」関
連の構成が削除されます。適宜、startup-configuration を更新してください。
================================================================================
[####################] 100% Install operation 9 completed successfully at Tue Nov 17 05:05:59 2020
```

RPM の削除

RPM を削除するには、次の手順を実行します。

手順

install remove *<rpm>* コマンドを使用して、リポジトリから RPM を削除します。

例：

```
switch(config)# show install inactive | grep vxlan vxlan-2.0.0.0-9.2.1.lib32_n9000 switch(config)#
 install remove vxlan Proceed with removing vxlan? (y/n)? [n] y [####################] 100% Install
 operation 890 Removal of base rpm package is not permitted at Thu Jun 7 13:52:15 2018
```

Dandified YUM コマンド

Nexus 9000 スイッチでは、DNF（Dandified YUM）は、NX-OS環境内のモジュラ ソフトウェア コンポーネント（RPM）を管理するために使用されるパッケージマネージャです。NX-OSリリース 10.1（x）以降、DNF はオプション機能の主要なツールとして YUM を置き換え、完全なシステムを実行せずに特定の機能（レイヤー3、BGP、またはOSPFなど）をインストール、

アップグレード、または削除できるようになりました。バイナリ アップグレード、またはスイッチのリロード。

（注） DNF コマンドは CTRL+C をサポートしていません。インストール コマンドは CTRL+C をサポートしています。 DNF コマンドを CTRL+C を使用して中断した場合は、**/isan/bin/patching_utils.py --unlock**であることを確認します。

DNF コマンドを使用したパッケージ操作

このセクションでは、 DNF コマンドを使用してパッケージの操作を実行する方法について説明します。

ここでは、 DNF コマンドを使用してパッケージの操作を実行する方法について説明します：

（注）
- DNF コマンドは、ボックスの BASH シェルからのみアクセスできます。NX-OS VSH ターミナルからはアクセスできません。
- sudo ユーザとして、スーパー ユーザ権限にアクセスできることを確認してください。

イメージのベース バージョン RPM を特定する

ベース RPM バージョンは、システム イメージにアーカイブされた、事前インストール済みの RPM です。

ls /rpms コマンドを使用して、イメージのベース バージョン RPM を特定します。

```
switch# ls /rpms bfd-2.0.0.0-9.2.1.lib32_n9000.rpm
ins_tor_sdk_t2-1.0.0.0-9.2.0.77.lib32_n9000.rpm
mtx-netconf-agent-2.0.0.0-9.2.1.lib32_n9000.rpm snmp-2.0.0.0-9.2.1.lib32_n9000.rpm
bgp-2.0.0.0-9.2.1.lib32_n9000.rpm ins_tor_sdk_t3-1.0.0.0-9.2.0.77.lib32_n9000.rpm
mtx-restconf-agent-2.0.0.0-9.2.1.lib32_n9000.rpm sr-2.0.0.0-9.2.1.lib32_n9000.rpm
container-tracker-2.0.0.0-9.2.1.lib32_n9000.rpm isis-2.0.0.0-9.2.1.lib32_n9000.rpm
mtx-telemetry-2.0.0.0-9.2.1.lib32_n9000.rpm svi-2.0.0.0-9.2.1.lib32_n9000.rpm
eigrp-2.0.0.0-9.2.1.lib32_n9000.rpm lacp-2.0.0.0-9.2.1.lib32_n9000.rpm
nbproxy-2.0.0.0-9.2.1.lib32_n9000.rpm tacacs-2.0.0.0-9.2.1.lib32_n9000.rpm
ext-eth-2.0.0.0-9.2.1.lib32_n9000.rpm lldp-2.0.0.0-9.2.1.lib32_n9000.rpm
ntp-2.0.0.0-9.2.1.lib32_n9000.rpm telemetry-2.3.4.0-9.2.1.lib32_n9000.rpm
fcoe-2.0.0.0-9.2.1.lib32_n9000.rpm mcast-2.0.0.0-9.2.1.lib32_n9000.rpm
nxos-ssh-2.0.0.0-9.2.1.lib32_n9000.rpm virtualization-2.0.0.0-9.2.1.lib32_n9000.rpm
fex-2.0.0.0-9.2.1.lib32_n9000.rpm mpls-2.0.0.0-9.2.1.lib32_n9000.rpm
ospf-2.0.0.0-9.2.1.lib32_n9000.rpm vtp-2.0.0.0-9.2.1.lib32_n9000.rpm
fhrp-2.0.0.0-9.2.1.lib32_n9000.rpm mtx-device-2.0.0.0-9.2.1.lib32_n9000.rpm repodata
vxlan-2.0.0.0-9.2.1.lib32_n9000.rpm guestshell-2.0.0.0-9.2.1.lib32_n9000.rpm
mtx-grpc-agent-2.0.0.0-9.2.1.lib32_n9000.rpm rip-2.0.0.0-9.2.1.lib32_n9000.rpm
icam-2.0.0.0-9.2.1.lib32_n9000.rpm mtx-infra-2.0.0.0-9.2.1.lib32_n9000.rpm
services-2.0.0.0-9.2.1.lib32_n9000.rpm
```

インストール済み RPM のリストをチェックする

dnf list installed コマンドを使用して機能 RPM とサードパーティ RPM をクエリして、 特定の RPM を grep 検索します。

機能 RPM の例を次に示します。

```
bash-4.2# dnf list installed | grep lib32_n9000  bfd.lib32_n9000 2.0.0.0-9.2.1
 @groups-repo core.lib32_n9000 2.0.0.0-9.2.1 installed eth.lib32_n9000 2.0.0.0-9.2.1
installed guestshell.lib32_n9000 2.0.0.0-9.2.1 @groups-repo lacp.lib32_n9000 2.0.0.0-9.2.1
 installed linecard2.lib32_n9000 2.0.0.0-9.2.1 installed lldp.lib32_n9000 2.0.0.0-9.2.1
 installed mcast.lib32_n9000 2.0.0.0-9.2.1 @groups-repo mtx-device.lib32_n9000
2.0.0.0-9.2.1 installed mtx-grpc-agent.lib32_n9000 2.0.0.0-9.2.1 installed
mtx-infra.lib32_n9000 2.0.0.0-9.2.1 installed mtx-netconf-agent.lib32_n9000 2.0.0.0-9.2.1
 installed mtx-restconf-agent.lib32_n9000 2.0.0.0-9.2.1 installed mtx-telemetry.lib32_n9000
 2.0.0.0-9.2.1 installed nbproxy.lib32_n9000 2.0.0.0-9.2.1 installed ntp.lib32_n9000
2.0.0.0-9.2.1 installed nxos-ssh.lib32_n9000 2.0.0.0-9.2.1 installed ospf.lib32_n9000
2.0.0.0-9.2.1 @groups-repo platform.lib32_n9000 2.0.0.0-9.2.1 installed snmp.lib32_n9000
 2.0.0.0-9.2.1 installed svi.lib32_n9000 2.0.0.0-9.2.1 installed tacacs.lib32_n9000
2.0.0.0-9.2.1 installed tor.lib32_n9000 2.0.0.0-9.2.0.77 installed
virtualization.lib32_n9000 2.0.1.0-9.2.1 @localdb vtp.lib32_n9000 2.0.0.0-9.2.1 installed
 vxlan.lib32_n9000 2.0.0.0-9.2.1 @groups-repo ...
```

インストール済み RPM の詳細を取得する

dnf info <*rpmname*> コマンドは、インストール済み RPM の詳細情報リストを出力します。

```
dnf info vxlan Loaded plugins: downloadonly, importpubkey, localrpmDB, patchaction,
 patching, protect-packages groups-repo | 1.1 kB 00:00 ... localdb | 951 B 00:00 ...
patching | 951 B 00:00 ... thirdparty | 951 B 00:00 ... Installed Packages Name : vxlan
 Arch : lib32_n9000 Version : 2.0.0.0 Release : 9.2.1 Size : 6.4 M Repo : installed From
 repo : groups-repo Summary : Cisco NXOS VxLAN URL : http://cisco.com/ License :
Proprietary Description : Provides VxLAN support
```

RPM のインストール

RPM をインストールすると、RPM がダウンロードされ、それぞれのプログラムがスイッチにコピーされます。これは、RPM をリモート サーバー（ネットワークで到達可能）からインストールする例を示しています。

```
bash 4.3 # dnf install
http://10.0.0.2/modularity/rpms/vxlan-2.0.1.0-9.2.1.lib32_n9000.rpm
Loaded plugins: downloadonly, importpubkey, localrpmDB, patchaction, patching,
protect-packages groups-repo | 1.1 kB 00:00 ... localdb | 951 B 00:00 ... localdb/primary
 | 886 B 00:00 ... localdb 1/1 patching | 951 B 00:00 ... thirdparty | 951 B 00:00 ...
インストール プロセス vxlan-2.0.1.0-9.2.1.lib32_n9000.rpm の設定 | 1.6 MB 00:00
/var/tmp/yum-root-RaANgb/vxlan-2.0.1.0-9.2.1.lib32_n9000.rpm を検査中:
vxlan-2.0.1.0-9.2.1.lib32_n9000
/var/tmp/yum-root-RaANgb/vxlan-2.0.1.0-9.2.1.lib32_n9000.rpm をインストール対象としてマーク
依存関係の解決 --> トランザクション チェックを実行中 ---> パッケージ vxlan.lib32_n9000
0:2.0.1.0-9.2.1 がインストールされます --> 依存関係の解決が完了しました 依存関係が解決されました
================================================================================
パッケージ アーキテクチャ バージョン リポジトリ サイズ
================================================================================
インストール中: vxlan lib32_n9000 2.0.1.0-9.2.1 /vxlan-2.0.1.0-9.2.1.lib32_n9000 6.4 M トランザクションの概要
================================================================================
パッケージを 1 つインストールします。合計サイズ: 6.4 M インストール後のサイズ: 6.4 M よろしいですか
```

```
[y/N]: y パッケージをダウンロードしています: トランザクションチェックを実行しています トランザクション
テストを実行しています トランザクションテストが成功しました トランザクションを実行しています インストー
ル中: vxlan-2.0.1.0-9.2.1.lib32_n9000 1/1 vxlan のプリインストールパッケージ バージョン mgmt を
開始 vxlan のプリインストールが完了しました vxlan のポストインストールパッケージ バージョン mgmt を開
始 vxlan のポストインストールが完了しました インストール済み: vxlan.lib32_n9000 0:2.0.1.0-9.2.1
完了!
```

この例は、RPMをローカルブートフラッシュからインストールする場合の例を示しています。

```
sudo dnf install /bootflash/vxlan-2.0.1.0-9.2.1.lib32_n9000.rpm  Loaded
 plugins: downloadonly, importpubkey, localrpmDB, patchaction, patching, protect-packages
 groups-repo | 1.1 kB 00:00 ... localdb | 951 B 00:00 ... patching | 951 B 00:00 ...
thirdparty | 951 B 00:00 ... Setting up Install Process Examining
/bootflash/vxlan-2.0.1.0-9.2.1.lib32_n9000.rpm: vxlan-2.0.1.0-9.2.1.lib32_n9000 Marking
 /bootflash/vxlan-2.0.1.0-9.2.1.lib32_n9000.rpm as an update to
vxlan-2.0.0.0-9.2.1.lib32_n9000 Resolving Dependencies --> Running transaction check
---> Package vxlan.lib32_n9000 0:2.0.0.0-9.2.1 will be updated ---> Package
vxlan.lib32_n9000 0:2.0.1.0-9.2.1 will be an update --> Finished Dependency Resolution
Dependencies Resolved
================================================================================
 Package Arch Version Repository Size
================================================================================
 Updating: vxlan lib32_n9000 2.0.1.0-9.2.1 /vxlan-2.0.1.0-9.2.1.lib32_n9000 6.4 M
Transaction Summary
================================================================================
 Upgrade 1 Package Total size: 6.4 M Is this ok [y/N]: y Downloading Packages: Running
Transaction Check Running Transaction Test Transaction Test Succeeded Running Transaction
 Updating : vxlan-2.0.1.0-9.2.1.lib32_n9000 1/2 starting pre-install package version
mgmt for vxlan pre-install for vxlan complete starting post-install package version mgmt
 for vxlan post-install for vxlan complete Cleanup : vxlan-2.0.0.0-9.2.1.lib32_n9000 2/2
 Updated: vxlan.lib32_n9000 0:2.0.1.0-9.2.1 Complete!
```

次には、RPM がリポジトリ内で利用可能な場合の RPM のインストールを示しています。

```
dnf install eigrp
```

RPM のアップグレード

このトピックは、RPM をリモート サーバー（ネットワークで到達可能）からアップグレードする場合の例を提供します。

```
bash 4.3 # dnf upgrade
http://10.0.0.2/modularity/rpms/vxlan-2.0.1.0-9.2.1.lib32_n9000.rpm
Loaded plugins：downloadonly、importpubkey, localrpmDB、patchaction、patching、
protect-packages groups-repo | 1.1 kB 00:00 ... localdb | 951 B 00:00 ... patching | 951
 B 00:00 ... thirdparty | 951 B 00:00 ... アップグレードプロセスの設定
vxlan-2.0.1.0-9.2.1.lib32_n9000.rpm | 1.6 MB 00:00
/var/tmp/yum-root-RaANgb/vxlan-2.0.1.0-9.2.1.lib32_n9000.rpm を検査しています:
vxlan-2.0.1.0-9.2.1.lib32_n9000
/var/tmp/yum-root-RaANgb/vxlan-2.0.1.0-9.2.1.lib32_n9000.rpm を
vxlan-2.0.0.0-9.2.1.lib32_n9000 への更新としてマークします 依存関係を解決しています --> トランザ
クション チェックを実行しています ---> パッケージ vxlan.lib32_n9000 0:2.0.0.0-9.2.1 が更新されま
す ---> パッケージ vxlan.lib32_n9000 0:2.0.1.0-9.2.1 は更新 --> 依存関係の解決が完了しました。依
存関係が解決されました。
================================================================================
 パッケージアーキテクチャ バージョン リポジトリ サイズ
================================================================================
 更新中: vxlan lib32_n9000 2.0.1.0-9.2.1 /vxlan-2.0.1.0-9.2.1.lib32_n9000 6.4 M トランザク
ション概要
```

```
アップグレード 1 パッケージ 合計サイズ: 6.4 M よろしいですか [y/N]: y パッケージをダウンロード中:
トランザクションチェックを実行中 トランザクションテストを実行中 トランザクションテスト成功 トランザクショ
ンを実行中 ** 既存の rpmdb の問題が 1 つ見つかりました。「yum check」の出力は次のとおりです:
busybox-1.23.2-r0.0.x86_64 には busybox-syslog の要件が不足しています 更新中:
vxlan-2.0.1.0-9.2.1.lib32_n9000 1/2 vxlan のプリインストールパッケージ バージョン mgmt を開始
vxlan のプリインストール完了 vxlan のポストインストールパッケージ バージョン mgmt を開始 vxlan のポ
ストインストール完了 クリーンアップ: vxlan-2.0.0.0-9.2.1.lib32_n9000 2/2 更新済み:
vxlan.lib32_n9000 0:2.0.1.0-9.2.1完了！
```

この例は、RPM をローカル ブートフラッシュからアップグレードする場合の例を示しています。

```
sudo dnf upgrade /bootflash/vxlan-2.0.1.0-9.2.1.lib32_n9000.rpm Loaded
plugins：downloadonly、importpubkey, localrpmDB、patchaction、patching、protect-packages
groups-repo | 1.1 kB 00:00 ... localdb | 951 B 00:00 ... patching | 951 B 00:00 ...
thirdparty | 951 B 00:00 ... アップグレード プロセスの設定
/bootflash/vxlan-2.0.1.0-9.2.1.lib32_n9000.rpm を検査中: vxlan-2.0.1.0-9.2.1.lib32_n9000
/bootflash/vxlan-2.0.1.0-9.2.1.lib32_n9000.rpm を vxlan-2.0.0.0-9.2.1.lib32_n9000 への更
新としてマーク 依存関係の解決 --> トランザクション チェックを実行中 ---> パッケージ vxlan.lib32_n9000
0:2.0.0.0-9.2.1 が更新されます ---> パッケージ vxlan.lib32_n9000 0:2.0.1.0-9.2.1 は更新にな
ります --> 依存関係の解決が完了しました 依存関係が解決されました

パッケージアーキテクチャ バージョン リポジトリ サイズ

更新中: vxlan lib32_n9000 2.0.1.0-9.2.1 /vxlan-2.0.1.0-9.2.1.lib32_n9000 6.4 M トランザク
ション概要

アップグレード 1 パッケージ 合計サイズ: 6.4 M よろしいですか [y/N]: y パッケージをダウンロードして
います: トランザクションチェックを実行しています トランザクションテストを実行しています トランザクション
テストが成功しました トランザクションを実行しています 更新中: vxlan-2.0.1.0-9.2.1.lib32_n9000 1/2
vxlan のプリインストールパッケージ バージョン mgmt を開始 vxlan のプリインストールが完了しました
vxlan のポストインストールパッケージ バージョン mgmt を開始 vxlan のポストインストールが完了しました
クリーンアップ: vxlan-2.0.0.0-9.2.1.lib32_n9000 2/2 更新済み: vxlan.lib32_n9000
0:2.0.1.0-9.2.1 完了!
```

これは、RPM がいずれかのリポジトリ内で利用可能な場合の RPM のアップグレードの例を示しています。

```
dnf upgrade eigrp
```

RPM のダウングレード

このトピックは、RPM をリモート サーバー（ネットワークで到達可能）からダウングレードする場合の例を提供します。

```
sudo dnf  downgrade vxlan-2.0.0.0-9.2.1.lib32_n9000 Loaded plugins：
downloadonly、importpubkey、localrpmDB、patchaction、patching, protect-packages Setting
up Downgrade Process groups-repo | 1.1 kB 00:00 ... localdb | 951 B 00:00 ...
localdb/primary | 1.3 kB 00:00 ... localdb 2/2 patching | 951 B 00:00 ... thirdparty |
951 B 00:00 ... 依存関係の解決 --> トランザクション チェックを実行中 ---> パッケージ
vxlan.lib32_n9000 0:2.0.0.0-9.2.1 はダウングレードされます ---> パッケージ vxlan.lib32_n9000
0:2.0.1.0-9.2.1 は削除されます --> 依存関係の解決が完了しました 依存関係が解決されました

パッケージ アーキテクチャ バージョン リポジトリ サイズ
```

```
ダウングレード：vxlan lib32_n9000 2.0.0.0-9.2.1 groups-repo 1.6 M トランザクション概要
```

```
ダウングレード 1 パッケージ 合計ダウンロード サイズ：1.6 M これでよろしいですか [y/N]：y パッケージ
をダウンロード中：トランザクション チェックを実行中 トランザクション テストを実行中 トランザクション テ
スト成功 トランザクションを実行中 インストール中：vxlan-2.0.0.0-9.2.1.lib32_n9000 1/2 vxlan のプ
リインストール パッケージ バージョン mgmt を開始 vxlan のプリインストール完了 vxlan のポストインス
トール パッケージ バージョン mgmt を開始 vxlan のポストインストール完了 クリーンアップ：
vxlan-2.0.1.0-9.2.1.lib32_n9000 2/2 削除済み：vxlan.lib32_n9000 0:2.0.1.0-9.2.1 インストー
ル済み：vxlan.lib32_n9000 0:2.0.0.0-9.2.1 完了！
```

この例は、RPM をローカル ブートフラッシュからダウングレードする場合の例を示しています。

```
dnf downgrade /bootflash/eigrp-2.0.0-9.2.1.lib32_n9000.rpm
```

これは、RPM がいずれかのリポジトリ内で利用可能な場合の RPM のダウングレードの例を示しています。

dnf eigrp のダウングレード

RPM を削除

RPM を削除すると、RPM がアンインストールされ、機能のコンフィグレーション コマンドがすべて削除されます。RPM を削除するには、**dnf erase** <*rpm*> コマンドを使用します。

```
bash-4.2# sudo dnf erase vxlan Loaded plugins: downloadonly, importpubkey, localrpmDB,
 patchaction, patching, protect-packages Setting up Remove Process Resolving Dependencies
 --> Running transaction check ---> Package vxlan.lib32_n9000 0:2.0.1.0-9.2.1 will be
erased --> Finished Dependency Resolution Dependencies Resolved
```

```
 Package Arch Version RepositorySize
```

```
 Removing: vxlan lib32_n9000 2.0.1.0-9.2.1 @/vxlan-2.0.1.0-9.2.1.lib32_n9000 6.4 M
Transaction Summary
```

```
 Remove1 Package Installed size: 6.4 M Is this ok [y/N]: y Downloading Packages: Running
 Transaction Check Running Transaction Test Transaction Test Succeeded Running Transaction
 Erasing : vxlan-2.0.1.0-9.2.1.lib32_n9000 1/1 starting pre-remove package version mgmt
 for vxlan pre-remove for vxlan complete Removed: vxlan.lib32_n9000 0:2.0.1.0-9.2.1
Complete!
```

DNF グループのサポート

DNF グループは、

- 管理者によるパッケージの管理を簡素化、より高度な柔軟性を提供、
- 柔軟性を向上し、
- 管理者がパッケージのコレクションを論理的なグループとして管理することを可能にします。

DNF のグループのサポートは、パッケージ管理の一部です。

管理者は、パッケージ（RPM）のリストを論理グループにグループ化して、さまざまな操作を実行することができます。DNF がサポートするグループ コマンドは、

- **grouplist**
- **groupinfo**
- **groupinstall**
- **groupremove**、および
- **groupupdate**を含みます。

DNF のグループは、大きく分けて、レイヤ 2、レイヤ 3、ルーティング、および管理として分類できます。

利用可能のパッケージ グループの表示のグループリスト コマンド

Linux では、複数のパッケージを特定のグループにまとめることができます。dnf でパッケージを個別にインストールするのではなく、特定のグループをインストールして、そのグループに属するすべての関連パッケージをインストールできます。たとえば、使用可能なすべてのグループを一覧表示するには、次を使用します： **dnf grouplist** コマンドに適用されます。

次のコマンドを使用して Linux で使用可能なすべてのパッケージ グループを一覧表示します。**dnfgrouplist** コマンドを使用します。

```
bash-4.4# dnf grouplist 最後のメタデータ有効期限チェック: 2024 年 3 月 8 日（金曜日）12:26:33
 PM UTC の 0:00:00 分前。] --- B/s | 0 B --:-- ETA 利用可能なグループ：管理ルーティング L2 L3
bash-4.4#
```

パッケージ グループの内容を表示する Groupmembers コマンド

次のコマンドを使用して説明とパッケージ グループのコンテンツを表示します。 **dnf groupinfo** コマンドを使用します。

このコマンドは、グループの機能メンバのリストを出力します。

```
bash-4.4# dnf groupinfo l2 最後のメタデータ有効期限チェック: 2024 年 3 月 8 日（金）12:27:44
 PM UTC の 0:00:00 分前。] --- B/s | 0 B --:-- ETA Group: L2 Mandatory Packages: lacp lldp
 svi vtp bash-4.4#
```

メンバー RPM のインストールおよびアップグレード用の groupinstall コマンド

このコマンドは、メンバRPMのインストールとアップグレードの両方に使用します。メンバーがインストールされていない場合は、使用可能な最も高いバージョンがインストールされます。メンバーがすでにインストールされていてより高いバージョンの RPM が使用可能である場合、このコマンドでそのメンバーがアップグレードされます。

```
bash-4.4# dnf groupinstall l3 最後のメタデータ有効期限チェック: 0:00:00 ago on Fri 08 Mar
 2024 12:38:05 PM UTC. ] --- B/s | 0 B --:--  ETA 冗長システムではありません。何もする必要はありません。依存関係が解決されました。
================================================================================
 Group Packages
================================================================================

 Marking packages as installed by the group: @L3 bfd Is this ok [y/N]: y Complete! インストール操作 10 は 2024 年 3 月 8 日（12:38:08）に正常に完了しました。[####################] 100%
```

パッケージ グループ アップデートの Groupupdate コマンド

dnf groupupdate コマンドを使用して、既存のインストール済みグループ パッケージを更新します。

```
bash-4.4# dnfdnf groupupdate l3 l3

 最後のメタデータ有効期限チェック: 2024 年 3 月 13 日（水）12 時 30 分 11 秒（UTC）の 0:00:00 分前] --- B/s | 0 B --:- ETA の依存関係が解決されました。
=================================================================================================$
 Group Packages
=================================================================================================$
 Marking packages as installed by the group: @L3 bfd
=================================================================================================$
 Package Arch Version Repository Size
=================================================================================================$
 Installing group packages: bfd lib32_64_n9000 2.0.0.0-10.4.3 groups-repo 562 k Transaction Summary
=================================================================================================$
 Install 1 Package Total size: 562 k Installed size: 2.3 M Is this ok [y/N]: y Downloading Packages: Running transaction check Transaction check succeeded. トランザクション テスト トランザクション テスト成功を実行します。Running transaction Preparing : 1/1 Running scriptlet: bfd-2.0.0.0-10.4.3.lib32_64_n9000 1/1 starting pre-install package version mgmt for bfd pre-install for bfd complete Installing : bfd-2.0.0.0-10.4.3.lib32_64_n9000 1/1 Running scriptlet: bfd-2.0.0.0-10.4.3.lib32_64_n9000 1/1 starting post-install package version mgmt for bfd post-install for bfd complete Verifying : bfd-2.0.0.0-10.4.3.lib32_64_n9000 1/1 Installed: bfd.lib32_64_n9000 2.0.0.0-10.4.3 Complete! インストール操作 14 は 2024 年 3 月 13 日（12:30:23）水曜日に正常に完了しました。[####################] 100% bash-4.4#
```

グループを削除するための Grouperase コマンド

すべての RPM メンバーまたは、グループを削除するために **dnf grouperase** コマンドを使用します。

```
bash-4.4# dnf grouperase l3 Dependencies resolved.
=================================================================================================
 Group Packages
=================================================================================================
 Marking packages as removed by the group: @L3 bfd
=================================================================================================
 Package Arch Version Repository Size
=================================================================================================
 Removing: bfd lib32_64_n9000 2.0.0.0-10.4.3 @System 2.3 M Transaction Summary
=================================================================================================
 Remove 1 Package Freed space: 2.3 M Is this ok [y/N]: y Running transaction check Transaction check succeeded. トランザクション テスト トランザクション テスト成功を実行します。Running transaction Preparing : 1/1 Running scriptlet: bfd-2.0.0.0-10.4.3.lib32_64_n9000 starting pre-remove package version mgmt for bfd pre-remove for bfd complete Erasing : bfd-2.0.0.0-10.4.3.lib32_64_n9000 1/1 Running scriptlet: bfd-2.0.0.0-10.4.3.lib32_64_n9000 1/1 starting post-remove package version mgmt for bfd post-remove for bfd complete Verifying : bfd-2.0.0.0-10.4.3.lib32_64_n9000 1/1 Removed: bfd.lib32_64_n9000 2.0.0.0-10.4.3 Complete! インストール操作 11 は 2024 年 3 月 8 日（12:38:41）に正常に完了しました。[####################] 100% bash-4.4#
```

リポジトリを特定する

この **dnf repolist all** コマンドは、スイッチに存在するリポジトリに加え、これらのリポジトリに含まれる RPM の数をリストします。

```
bash 4.3 # dnf repolist all  Loaded plugins: downloadonly, importpubkey, localrpmDB,
 patchaction, patching, protect-packages groups-repo | 1.1 kB 00:00 ... localdb | 951 B
 00:00 ... patching | 951 B 00:00 ... thirdparty | 951 B 00:00 ... repo id repo name
status groups-repo Groups-RPM Database enabled: 37 localdb Local RPM Database enabled:
6 patching Patch-RPM Database enabled: 0 thirdparty Thirdparty RPM Database enabled: 0
open-nxos open-nxos disabled repolist: 43
```

インストールされている DNF バージョン

インストールされている DNF のバージョンを表示するには、 **dnf --version** コマンドを使用します。

```
dnf --version  3.4.3 Installed: rpm-5.4.14-r0.0.x86_64 at 2018-06-02 13:04 Built :
Wind River <info@windriver.com> at 2018-04-27 08:36 Committed: Wind River
<info@windriver.com> at 2018-04-27 Installed: yum-3.4.3-r9.0.x86_64 at 2018-06-02 13:05
 Built : Wind River <info@windriver.com> at 2018-04-27 08:36 Committed: Wind River
<info@windriver.com> at 2018-04-27
```

NX-OS コマンドの DNF コマンドへのマッピング

表は、NX-OS コマンドとそれに対応する DNF コマンドを示しています。

表 *5:* パッチ適用コマンド リファレンス

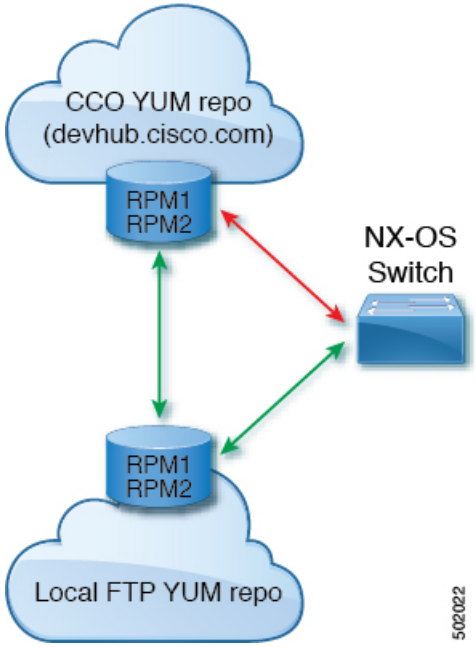
NX-OS コマンド	DNF コマンド
show install inactive	**dnf list --patch-only available**
show install active	**dnf list --patch-only installed**
show install committed	**dnf list --patch-only committed**
show install packages	**dnf list --patch-only**
show install pkg-info	**dnf info --patch-only**
show install log	**dnf history --show-patch-log** ここで log_cmd は次のとおりです。 • opid：ある操作 ID に固有なログ • last：最新の操作のログを示します。 • reverse：逆の順序でログを表示します。 • detail：詳細ログを表示します。 • from：特定の操作 ID 以降のログを示します。

NX-OS コマンド	DNF コマンド
clear install log	**dnf history --clear-patch-log=** ここで clear_log_cmd は次のとおりです。 • all：ログを完全にクリアします。 • ：この操作 ID 以降のログをクリアします。
install add	**dnf install --add bootflash:/**
install remove	**dnf install --remove**
install remove inactive	**dnf install --remove all**
install activate	**dnf install --no-persist --nocommit** （注） デフォルトでは、すべてのパッケージがアクティブ化され、コミットされます。
install deactivate	**dnf erase --nocommit** （注） デフォルトでは、すべてのパッケージが非アクティブ化され、コミットされます。
install commit	**dnf install --commit**
Install commit	**dnf install --commit all**

FTP サーバーの構成とローカル FTP YUM リポジトリのセットアップ

ローカル FTP YUM リポジトリをセットアップするには、この図に示すように、初めに FTP サーバーを作成して、ローカル FTP YUM リポジトリを作成し、FTP サーバーに到達するように NX-OS スイッチを設定します。

図 *2 : FTP* サーバーの構成とローカル *FTP YUM* リポジトリのセットアップ



（注） NX-OS リリース 10.1（1）の場合は、次の https://devhub.cisco.com/artifactory/open-nxos/10.1.1/ をアクセスします。 **open-nxos** リポジトリ。

Red Hat Enterprise Linux 7（RHEL7）仮想マシン上に FTP サーバーを作成する

Red Hat Enterprise Linux 7（RHEL7）仮想マシン上に FTP サーバを作成するには、次の手順を実行します。

手順

ステップ 1 **dnfinstall vsftpd** コマンドを使用して、vsftpd （FTP サーバー）をインストールします。

ステップ 2 **systemctl start vsftpd** コマンドを使用して FTP サーバーを起動します。

ステップ 3 **systemctl status vsftpd**コマンドを使用して、FTP サーバーのステータスを確認します。

ステップ 4 外部システムから FTP サービスへのアクセスを提供し、**firewall-cmd --zone=public --permanent --add-port=21/tcp** コマンドを使用してポート 21 を開きます。

ステップ 5 **firewall-cmd --zone=public --permanent --add-service=ftp** コマンドを使用して、 FTP サービスを追加します。

ステップ 6 **firewall-cmd --reload** コマンドを使用してサーバーをリロードします。

ステップ 7 FTP サーバーでファイル（test.txt など）をホストし、**wget ftp://** < *ip of FTP server* > / *test.txt*コマンドを使用してそのファイルの Wget を試みます。

（注）
/var/ftp/ ディレクトリは、FTP サーバーのデフォルト ホーム ディレクトリです。

ローカル FTP YUM リポジトリを作成する

外部リポジトリ RPM と FTP サーバーを同期し、ローカル FTP YUM リポジトリを作成するには、この手順で提供されているステップを実行します。

手順

ステップ 1 **touch/etc/yum.repos.d/local.repo** コマンドを使用してリポジトリ ファイルを作成します。

/etc/yum.repos.d/ 下にリポジトリ ファイルを作成します。たとえば、**local.repo** リポジトリを作成してベース URL を追加します。

例：

```
bash-4.3# touch /etc/yum.repos.d/local.repo
```

ステップ 2 リポジトリ ファイルを編集、**vim /etc/yum.repos.d/local.repo** コマンドを使用して localrepo の詳細をコピーします。

（注）
ベース URL を必要なリポジトリ URL に変更します。

例：

```
bash-4.3# vim /etc/yum.repos.d/local.repo [localrepo] name=localrepo baseurl=
https://devhub.cisco.com/artifactory/open-nxos/7.0-3-I2-1/x86_64/ enabled=1 gpgcheck=0 sslverify=0
```

ステップ 3 続行するには、**cat /etc/yum.repos.d/local.repo** コマンドを使用してローカル リポジトリのデータを確認します。

例：

```
bash-4.3# cat /etc/yum.repos.d/local.repo [localrepo] name=localrepo baseurl=
https://devhub.cisco.com/artifactory/open-nxos/7.0-3-I2-1/x86_64/ enabled=1 gpgcheck=0 sslverify=0
```

ステップ 4 **dnf repolist** コマンドを使用して、リポジトリの到達可能性を確認する。

例：

```
bash-4.3# dnf repolist Loaded plugins: fastestmirror, langpacks Loading mirror speeds from cached
hostfile * base: mirror.dhakacom.com * extras: mirror.dhakacom.com * updates: mirror.dhakacom.com
repo id repo name status base/7/x86_64 CentOS-7 - Base 9,911 extras/7/x86_64 CentOS-7 - Extras 313
 localrepo localrepo 687 updates/7/x86_64 CentOS-7 - Updates 711 repolist: 11,622
```

ステップ 5 **nohup reposync -r** < *repo-name mentioned in the local.repo* > **-p** < *directory path to sync* > **&** コマンドを使用して、外部リポジトリから FTP サーバーのホーム ディレクトリにすべてのパッケージを同期します。

例：

```
nohup reposync -r localrepo -p /var/ftp/ &
```

このコマンドは、**/var/ftp/** 内部に名前 **local.repo** でディレクトリを作成し、すべてのパッケージを **devhub.cisco.com** からこのディレクトリにダウンロードします。

ステップ 6 **tail -f nouhup.out** コマンドを使用して、同期のステータスを確認します。

a) **ls /var/ftp/localrepo**
b) **cd /var/ftp/localrepo/ && createrepo .**

FTP サーバーに到達するようにスイッチを構成する

FTP サーバーに到達するようにスイッチを設定するには、次の手順を実行します。

手順

ステップ 1 **run bash sudo su** コマンドを使用して、sudo ユーザーとしてログインします。

ステップ 2 **ip netns exec management ping** < *ip_address* > コマンドを使用して、 FTP サーバーに到達できることを確認します。

このコマンドは、スイッチから **ping** コマンドを使用して、FTP サーバー アドレスの到達可能性を確認します。

ステップ 3 **Touch/etc/yum/repos.d/ftp.repo** コマンドを使用してリポジトリ ファイルを作成します。

このコマンドは、**/etc/yum/**repos.d/ の下にリポジトリファイルを作成します。たとえば、**ftp.repo** リポジトリを作成します。

例：

```
bash-4.3# touch /etc/yum/repos.d/ftp.repo
```

ステップ 4 リポジトリ ファイルを編集、 **vim /etc/yum/repos.d/ftp.repo** コマンドを使用して ftp リポジトリの詳細をコピーします。

（注）
ベース URL を必要な ftp サーバー IP に変更します。

例：

```
bash-4.3# vim /etc/yum/repos.d/ftp.repo [ftp] name=ftp baseurl= ftp://198.51.100.1/localrepo/
 enabled=1 gpgcheck=0 sslverify=0
```

ステップ 5 **cat /etc/yum/repos.d/ftp.repo** コマンドを使用して FTP サーバー アドレスを URL としてスイッチにレポジトリ ファイルを作成します。

例：

```
bash-4.3# cat /etc/yum/repos.d/ftp.repo [ftp] name=ftp baseurl=ftp://198.51.100.1/localrepo/ enabled=1
 gpgcheck=0 sslverify=0
```

ステップ 6 Bash シェル プロンプトを使用するには、 **ip netns exec management bash** コマンドを実行します。

ステップ 7 **dnfrepolist** コマンドを使用して、新しく作成されたリポジトリの到達可能性を確認する。

例：

```
bash-4.3# dnf repolist Loaded plugins: downloadonly, importpubkey, localrpmDB, patchaction, patching,
 : protect-packages groups-repo | 1.1 kB 00:00 ... localdb | 951 B 00:00 ... patching | 951 B 00:00
 ... thirdparty | 951 B 00:00 ... thirdparty/primary | 758 B 00:00 ... thirdparty 1/1 repo id repo
 name status groups-repo Groups-RPM Database 37 localdb Local RPM Database 0 patching Patch-RPM
Database 0 thirdparty Thirdparty RPM Database 1 ftp ftp 686 repolist: 724
```

ステップ 8 **dnflist available** コマンドを使用して、新しいリポジトリで使用可能なパッケージをリストします。

インストール操作用ユーザー ロールの作成

この **install** コマンドは、admin ロールのユーザーのみが使用できます。値は、 **install** コマンドは、RBAC によりユーザーが利用できるようになります。RBAC 構成時の注意事項については、 「ユーザー アカウントおよび RBAC のガイドラインと制限事項」を参照してください。

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。