bash

- Bash について （1 ページ）
- 注意事項と制約事項 （1 ページ）
- Bash へのアクセス （2 ページ）
- 権限をルートにエスカレーションする （4 ページ）
- Bash コマンドの例 （6 ページ）
- 機能 RPM の管理 （7 ページ）
- DME のモジュール性のサポート （10 ページ）
- パッチ RPM の管理 （19 ページ）
- SDK または ISO で構築されたサード パーティ プロセスの永続的なデーモン化 （26 ページ）
- ネイティブ Bash シェルからのアプリケーションの永続的な起動 （27 ページ）
- 現用系ブートフラッシュからスタンバイ ブートフラッシュへのファイルの同期（28 ページ）
- Kstack を介してコピー （29 ページ）
- ネイティブ Bash シェルのアプリケーション例 （30 ページ）

Bash について

Cisco NX-OS CLI に加えて、Cisco Nexus シリーズ 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。

注意事項と制約事項

Bash シェルには、次の注意事項と制約事項があります。

- インターフェイスのリンクローカルアドレスを定義すると、Netstack はカーネルのネットデバイスに /64 プレフィックスをインストールします。

  新しいリンクローカルアドレスがカーネルで設定されると、カーネルはカーネルルーティングテーブルに /64 ルートをインストールします。

ピア ボックスのインターフェイスに、同じ /64 サブネットに属するリンクローカル アドレスが設定されていない場合、 は bash プロンプトから成功しません。**ping** Cisco NX-OS は正常に動作します。**ping**

- /isan フォルダ内のバイナリは、 コマンドで入力するシェルの環境とは異なるように設定された環境で実行するためのものです。**run bash** この環境内の動作は予測できないため、Bash シェルからこれらのバイナリを使用しないことをお勧めします。
- デフォルトでは、Bash シェルは自動的にログアウトしません。ユーザーは、標準の Linux 環境変数 TMOUT を使用して、目的とするアイドルタイムアウトを設定できます。値は秒単位です。たとえば、「export TMOUT=300」です。
- Cisco Python モジュールをインポートする場合は、Bash シェルから Python を使用しないでください。代わりに、NX-OS VSH で最新の Python を使用します。
- 一部のプロセスおよびコマンドでは、大量の出力が発生する可能性があります。**show** スクリプトを実行していて、実行時間の長い出力を終了する必要がある場合は、Ctrl+C（Ctrl+Z ではなく）を使用してコマンド出力を終了します。Ctrl+Z を使用すると、このキー コマンドによって SIGCONT（信号継続）メッセージが生成され、スクリプトが停止する可能性があります。SIGCONT メッセージによって停止されたスクリプトは、動作を再開するためにユーザの介入が必要です。
- コマンドが実行中であり、強制終了する必要がある場合は、 コマンドを使用しないでください。**show tech supportclear tech-support lock** Ctrl+C を使用します。

  これは、テクニカルサポート情報の実際の収集が行われるバックグラウンド VSH セッションを強制終了しないためです。**clear tech-support lock** 代わりに、 コマンドは、CLI が呼び出されたフォアグラウンド VSH セッションのみを強制終了します。**clear tech-support lockshow tech support**

  show tech-support セッションを正しく強制終了するには、Ctrl+C を使用します。

  誤って を使用した場合は、次の手順を実行してバックグラウンド VSH プロセスを強制終了します。 **clear tech-support lock**

  1. Bash シェルに入ります。
  2. コマンドの VSH セッション（）を見つけます。**ps -l | moreshow tech support**
  3. セッションの VSH に関連付けられている PID（PID など）を強制終了します。**show tech supportkill -9**

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?
  run         Execute/run program
  run-script  Run shell scripts

switch# run bash?
  bash  Linux-bash

switch# run bash
bash-4.2$ whoami
admin
bash-4.2$ pwd
/bootflash/home/admin
bash-4.2$
```

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

たとえば、**run bash** コマンド を使用して **whoami** を実行することもできます。

```
run bash whoami
```

ユーザー **shelltype** を構成して Bash を実行することもできます。

```
username foo shelltype bash
```

このコマンドにより、ログイン時に Bash シェルを直接実行できるようになります。この場合、**feature bash-shell** を有効にする必要はありません。

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：

- 管理者権限ユーザー（network-admin/vdc-admin）は、NX-OS における、Linux の root 権限ユーザーに相当します。
- 認証された管理者ユーザーのみが権限を root に昇格できます。認証された管理者権限ユーザーにパスワードは必要ありません。*
- 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。
- 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。

* Cisco NX-OS リリース 9.2(3) 以降では、管理者（ネットワーク管理者ロールを持つユーザー）の特権ユーザーであっても、一部の使用例でパスワードプロンプトが必要な場合は、**system security hardening sudo prompt-password** コマンドを入力します。

NX-OS ネットワーク管理者ユーザーは、次の場合に root にエスカレーションして、構成コマンドを NX-OS VSH に渡す必要があります。

- NX-OS ユーザはシェルタイプの Bash を使用しており、シェルタイプの Bash を使用してスイッチにログインしています。
- Bash でスイッチにログインした NX-OS ユーザは、引き続きスイッチで Bash を使用します。

sudo su 'vsh -c "<configuration commands>"' または **sudo bash -c 'vsh -c "<configuration commands>"'** を実行します。

次の例は、デフォルトのシェルタイプが Bash であるネットワーク管理者ユーザー MyUser が、**sudo** を使用して構成コマンドを NX-OS に渡す方法を示しています。

```
ssh -l MyUser 1.2.3.4
-bash-4.2$ sudo vsh -c "configure terminal ; interface eth1/2 ; shutdown ; sleep 2 ; show
```

```
interface eth1/2 brief"

--------------------------------------------------------------------------------
Ethernet      VLAN    Type Mode   Status  Reason                   Speed     Port
Interface                                                                     Ch #
--------------------------------------------------------------------------------
Eth1/2        --      eth  routed down    Administratively down      auto(D) --
```

次の例は、デフォルトのシェルタイプが Bash であるネットワーク管理者ユーザー MyUser が、NX-OS に入り、NX-OS で Bash を実行する方法を示しています。

```
ssh -l MyUser 1.2.3.4
-bash-4.2$ vsh -h
Cisco NX-OS Software
Copyright (c) 2002-2016, Cisco Systems, Inc. All rights reserved.
Nexus 9000v software ("Nexus 9000v Software") and related documentation,
files or other reference materials ("Documentation") are
the proprietary property and confidential information of Cisco
Systems, Inc. ("Cisco") and are protected, without limitation,
pursuant to United States and International copyright and trademark
laws in the applicable jurisdiction which provide civil and criminal
penalties for copying or distribution without Cisco's authorization.

Any use or disclosure, in whole or in part, of the Nexus 9000v Software
or Documentation to any third party for any purposes is expressly
prohibited except as otherwise authorized by Cisco in writing.
The copyrights to certain works contained herein are owned by other
third parties and are used and distributed under license. Some parts
of this software may be covered under the GNU Public License or the
GNU Lesser General Public License. A copy of each such license is
available at
http://www.gnu.org/licenses/gpl.html and
http://www.gnu.org/licenses/lgpl.html
***************************************************************************
*  Nexus 9000v is strictly limited to use for evaluation, demonstration      *
*  and NX-OS education. Any use or disclosure, in whole or in part of     *
*  the Nexus 9000v Software or Documentation to any third party for any      *
*  purposes is expressly prohibited except as otherwise authorized by     *
*  Cisco in writing.                                                       *
***************************************************************************
switch# run bash
bash-4.2$ vsh -c "configure terminal ; interface eth1/2 ; shutdown ; sleep 2 ; show interface
 eth1/2 brief"

--------------------------------------------------------------------------------
Ethernet      VLAN    Type Mode   Status  Reason                   Speed     Port
Interface                                                                     Ch #
--------------------------------------------------------------------------------
Eth1/2        --      eth  routed down    Administratively down      auto(D) --
```

（注） **sudo su -** は使用しないでください。使用すると、システムがハングします。

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
bash-4.2$ sudo su root
bash-4.2# whoami
root
```

```
bash-4.2# 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

次の例は、システム統計情報の表示方法を示しています：

```
switch# run bash
bash-4.2$ cat /proc/meminfo
<snip>
MemTotal:       16402560 kB
MemFree:        14098136 kB
Buffers:           11492 kB
Cached:          1287880 kB
SwapCached:            0 kB
Active:          1109448 kB
Inactive:         717036 kB
Active(anon):     817856 kB
Inactive(anon):   702880 kB
Active(file):     291592 kB
Inactive(file):    14156 kB
Unevictable:           0 kB
Mlocked:               0 kB
SwapTotal:             0 kB
SwapFree:              0 kB
Dirty:                32 kB
Writeback:             0 kB
AnonPages:        527088 kB
Mapped:            97832 kB
<\snip>
```

CLI からの Bash の実行

次に、**run bash**コマンドを使用して Bash から **ps** を実行する例を示します：

```
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 -   528 poll_s ?        00:00:03 init
1 S     0     2     0  0  80   0 -     0 kthrea ?        00:00:00 kthreadd
1 S     0     3     2  0  80   0 -     0 run_ks ?        00:00:56 ksoftirqd/0
1 S     0     6     2  0 -40   - -     0 cpu_st ?        00:00:00 migration/0
1 S     0     7     2  0 -40   - -     0 watchd ?        00:00:00 watchdog/0
1 S     0     8     2  0 -40   - -     0 cpu_st ?        00:00:00 migration/1
1 S     0     9     2  0  80   0 -     0 worker ?        00:00:00 kworker/1:0
1 S     0    10     2  0  80   0 -     0 run_ks ?        00:00:00 ksoftirqd/1
```

機能 RPM の管理

RPM インストールの前提条件

RPM をインストールまたは追加する前に、次の手順によりシステムの準備ができていることを確認します。

手順の概要

1. switch# **show logging logfile | grep -i "System ready"**
2. switch# **run bash sudo su**

手順の詳細

手順

	コマンドまたはアクション	目的
Step 1	switch# **show logging logfile | grep -i "System ready"**	Bash を実行する前に、この手順によって、RPM をインストールまたは追加する前のシステムの準備ができていることを確認します。 以下のような出力が表示されれば、続行します。 **2018 Mar 27 17:24:22 switch %ASCII-CFG-2-CONF_CONTROL: System ready**

Step 2	switch# **run bash sudo su**
	例:
	`switch# run bash sudo su`
	`bash-4.2#`
		Bash をロードします。

Bash からの機能 RPM のインストール

手順

	コマンドまたはアクション	目的
Step 1	**sudo dnf installed | grep** *platform*	スイッチにインストールされている NX-OS 機能 RPM のリストを表示します。
Step 2	**dnf list available**	使用可能な RPM のリストを表示します。
Step 3	**sudo dnf -y install** *rpm*	使用可能な RPM をインストールします。

例

次に、**bfd** RPM をインストールする例を示します。

```
bash-4.2$ dnf list installed | grep n9000
base-files.n9000                        3.0.14-r74.2                    installed
bfd.lib32_n9000                         1.0.0-r0                        installed
core.lib32_n9000                        1.0.0-r0                        installed
eigrp.lib32_n9000                       1.0.0-r0                        installed
eth.lib32_n9000                         1.0.0-r0                        installed
isis.lib32_n9000                        1.0.0-r0                        installed
lacp.lib32_n9000                        1.0.0-r0                        installed
linecard.lib32_n9000                    1.0.0-r0                        installed
lldp.lib32_n9000                        1.0.0-r0                        installed
ntp.lib32_n9000                         1.0.0-r0                        installed
nxos-ssh.lib32_n9000                    1.0.0-r0                        installed
ospf.lib32_n9000                        1.0.0-r0                        installed
perf-cisco.n9000_gdb                    3.12-r0                         installed
platform.lib32_n9000                    1.0.0-r0                        installed
shadow-securetty.n9000_gdb              4.1.4.3-r1                      installed
snmp.lib32_n9000                        1.0.0-r0                        installed
svi.lib32_n9000                         1.0.0-r0                        installed
sysvinit-inittab.n9000_gdb              2.88dsf-r14                     installed
tacacs.lib32_n9000                      1.0.0-r0                        installed
task-nxos-base.n9000_gdb                1.0-r0                          installed
tor.lib32_n9000                         1.0.0-r0                        installed
vtp.lib32_n9000                         1.0.0-r0                        installed
bash-4.2$ dnf list available
bgp.lib32_n9000                         1.0.0-r0
bash-4.2$ sudo dnf -y install bfd
```

（注） 起動時のスイッチのリロード時に、永続的な RPM の代わりに コマンドを使用します。**rpmdnf**それ以外の場合、RPMは最初にインストールされたのではなく、リポジトリ名またはファイル名を使用してインストールされるか、または表示されます。**dnf bashinstall cli**

機能 RPM のアップグレード

始める前に

dnf リポジトリに RPM の上位バージョンが存在する必要があります。

手順の概要

1. **sudo dnf -y upgrade***rpm*

手順の詳細

手順

	コマンドまたはアクション	目的
Step 1	**sudo dnf -y upgrade***rpm*	インストールされている RPM をアップグレードします。

例

次に、**bfd** RPM のアップグレードの例を示します。

```
bash-4.2$ sudo dnf -y upgrade bfd
```

機能 RPM のダウングレード

手順の概要

1. **sudo dnf -y downgrade***rpm*

手順の詳細

手順

	コマンドまたはアクション	目的
Step 1	**sudo dnf -y downgrade***rpm*	いずれかの dnf リポジトリに下位バージョンの RPM がある場合に、RPM をダウングレードします。

例

次に、**bfd** RPM をダウングレードする例を示します。

```
bash-4.2$ sudo dnf -y downgrade bfd
```

機能 RPM の消去

（注） SNMP RPM および NTP RPM は保護されており、消去できません。

これらの RPM をアップグレードまたはダウングレードできます。アップグレードまたはダウングレードを有効にするには、システムのリロードが必要です。

保護された RPM のリストについては、/etc/yum/protected.d/protected_pkgs.conf /etc/dnf/protected.d/protected_pkgs.conf を参照してください。

手順の概要

1. **sudo dnf -y erase***rpm*

手順の詳細

手順

	コマンドまたはアクション	目的
Step 1	**sudo dnf -y erase***rpm*	RPM を消去します。

例

次の例は、**bfd** RPM を消去する方法を示しています：

```
bash-4.2$ sudo dnf -y erase bfd
```

DME のモジュール性のサポート

NX-OS リリース 9.3(1) 以降、Cisco NX-OS イメージは DME のモジュール性をサポートします。これは、スイッチの RPM マネージャと相互運用して、DME RPM の非侵入型アップグレードまたはダウングレードを可能にします。非侵入型のアップグレードまたはダウングレードにより、システムの再起動を実行せずに RPM をインストールでき、DME データベースに設定がある他のアプリケーションの妨害を防ぐことができます。DME のモジュール性を使用すると、ISSU やシステムのリロードを行わずに、モデルの変更をスイッチに適用できます。

（注） DME RPM をロードした後、VSH を再起動して新しい MO のクエリを有効にする必要があります。

DME RPM のインストール

デフォルトでは、NX-OS リリース 9.3(1) にアップグレードすると、必須のアップグレード可能 RPM パッケージであるベース DME RPM がインストールされ、アクティブになります。DME RPM は、RPM ファイルのデフォルト インストール ディレクトリ（`/rpms`）にインストールされます。

コードまたはモデルを変更する場合は、DME RPM をインストールする必要があります。インストールするには、**install** コマンドを使用する NX-OS RPM マネージャ、または **dnf** などの標準 RPM ツールを使用します。**dnf** を使用する場合は、スイッチの Bash シェルにアクセスする必要があります。

手順

Step 1 **copy** *path-to-dme-rpm* **bootflash:** [*//sup-#*][*/path*]

例：

```
switch-1# copy scp://test@10.1.1.1/dme-2.0.1.0-9.3.1.lib32_n9000.rpm bootflash://
switch-1#
```

SCP を介して DME RPM をブートフラッシュにコピーします。

Step 2 DME RPM をインストールまたはアップグレードするには、次のいずれかの方法を選択します。

NX-OS の **install** コマンドを使用するには、次のコマンドを実行します。

- **install add** *path-to-dme-rpm* **activate**

  例：

```
switch-1#install add dme-2.0.1.0-9.3.1.lib32_n9000.rpm activate
Adding the patch (/dme-2.0.1.0-9.3.1.lib32_n9000.rpm)
[####################] 100%
Install operation 90 completed successfully at Fri Jun  7 07:51:58 2019

Activating the patch (/dme-2.0.1.0-9.3.1.lib32_n9000.rpm)
[####################] 100%
Install operation 91 completed successfully at Fri Jun  7 07:52:35 2019
switch-1#
```

- **install add** *path-to-dme-rpm* **activate upgrade**

  例：

```
switch-1#install add dme-2.0.1.0-9.3.1.lib32_n9000.rpm activate upgrade
Adding the patch (/dme-2.0.1.0-9.3.1.lib32_n9000.rpm)
[####################] 100%
Install operation 87 completed successfully at Fri Jun  7 07:18:55 2019

Activating the patch (/dme-2.0.1.0-9.3.1.lib32_n9000.rpm)
[####################] 100%
Install operation 88 completed successfully at Fri Jun  7 07:19:35 2019
switch-1#
```

- **install add** *path-to-dme-rpm*　それから **install activate** *path-to-dme-rpm*

  例：

```
switch-1#install add bootflash:dme-2.0.1.0-9.3.1.lib32_n9000.rpm
[####################] 100%
Install operation 92 completed successfully at Fri Jun  7 09:31:04 2019
switch-1#install activate dme-2.0.1.0-9.3.1.lib32_n9000.rpm
[####################] 100%
Install operation 93 completed successfully at Fri Jun  7 09:31:55 2019
switch-1#
```

dnf install を使用するには、次のコマンドを実行します。

- **dnf install --add** *path-to-dme-rpm*

```
switch-1# dnf install --add bootflash:///dme-2.0.10.0-9.3.1.lib32_n9000.rpm
Loaded plugins: downloadonly, importpubkey, localrpmDB, patchaction, patching,
              : protect-packages
[##################  ] 90%Install operation 96 completed successfully at Fri Jun  7 22:58:50
2019.

[####################] 100%
switch-1#
```

- **dnf install --no-persist --nocommit***path-to-dme-rpm*

  このオプションには、次に示すようにユーザーの操作が必要です。

  例：

```
switch-1# dnf install --no-persist --nocommit dme-2.0.10.0-9.3.1.lib32_n9000
Loaded plugins: downloadonly, importpubkey, localrpmDB, patchaction, patching,
              : protect-packages
groups-repo                                              | 1.1 kB     00:00 ...
localdb                                                  |  951 B     00:00 ...
localdb/primary                                          | 6.2 kB     00:00 ...
localdb                                                                     2/2
patching                                                 |  951 B     00:00 ...
thirdparty                                               |  951 B     00:00 ...
wrl-repo                                                 |  951 B     00:00 ...
Setting up Install Process
Resolving Dependencies
--> Running transaction check
---> Package dme.lib32_n9000 0:2.0.1.0-9.3.1 will be updated
---> Package dme.lib32_n9000 0:2.0.10.0-9.3.1 will be an update
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package     Arch                Version                  Repository       Size
================================================================================
Updating:
 dme         lib32_n9000         2.0.10.0-9.3.1           localdb          45 M

Transaction Summary
================================================================================
Upgrade       1 Package

Total download size: 45 M
Is this ok [y/N]: y
Retrieving key from file:///etc/pki/rpm-gpg/arm-Nexus9k-dev.gpg
Downloading Packages:
Running Transaction Check
Running Transaction Test
Transaction Test Succeeded
Running Transaction
```

```
/bootflash/.rpmstore/config/etc/pki/rpm-gpg/arm-Nexus9k-dev.gpg
System at HA Standby, running transaction on Standby first
  Updating   : dme-2.0.10.0-9.3.1.lib32_n9000                              1/2
starting pre-install package version mgmt for dme
pre-install for dme complete
ln: failed to create symbolic link /var/run/mgmt/sharedmeta-hash: File exists
ln: failed to create symbolic link /var/run/mgmt/dme-objstores.conf: File exists
ln: failed to create symbolic link /var/run/mgmt/samlog.config: File exists
mgmt/
mgmt/shmetafiles/
mgmt/shmetafiles/sharedmeta-ArgMetaData
mgmt/shmetafiles/sharedmeta-RelsMetaData
mgmt/shmetafiles/sharedmeta-ClassRelMetaData
mgmt/shmetafiles/sharedmeta-ChunkMetaData
mgmt/shmetafiles/sharedmeta-ConstPropMetaData
mgmt/shmetafiles/sharedmeta-ConstIdMetaData
mgmt/shmetafiles/sharedmeta-ClassMetaData
mgmt/shmetafiles/sharedmeta-PropRefsMetaData
mgmt/shmetafiles/sharedmeta-SvcMetaData
mgmt/shmetafiles/sharedmeta-ActionContextMetaData
mgmt/shmetafiles/sharedmeta-ConstDefTypeMetaData
mgmt/shmetafiles/sharedmeta-ConstArgMetaData
mgmt/shmetafiles/sharedmeta-ClassNamingMetaData
mgmt/shmetafiles/sharedmeta-ConstMetaData
mgmt/shmetafiles/sharedmeta-PropMetaData
mgmt/shmetafiles/sharedmeta-DnMetaData
  Cleanup    : dme-2.0.1.0-9.3.1.lib32_n9000                               2/2

Updated:
  dme.lib32_n9000 0:2.0.10.0-9.3.1

Complete!
switch-1#
```

インストールされている RPM の確認

NX-OS **show install** コマンドまたは **dnf list** を使用して、DME RPM がインストールされているかどうかを確認できます。

手順

方法を選択します。

- NX-OS の場合：

  show install active

  例：

```
switch-1# show install active
Boot Image:
        NXOS Image: bootflash:///<boot_image.bin>

Active Packages:
```

```
        dme-2.0.1.0-9.3.1.lib32_n9000
switch-1#
```

- **dnf list** では、**dnf** コマンドを発行する前にスイッチの Bash シェル（**run bash**）にログインする必要があります。

  dnf list --patch-only installed | grep dme

  例：

```
switch-1# dnf list --patch-only installed | grep dme
dme.lib32_n9000                        2.0.1.0-9.3.1                   @localdb
```

ローカル リポジトリの RPM のクエリ

スイッチ上の（ローカル）リポジトリを照会して、RPM が存在するかどうかを確認できます。

手順

Step 1 **run bash**

例：

```
switch-1# run bash
bash-4.3$
```

スイッチの Bash シェルにログインします。

Step 2 **ls /bootflash/.rpmstore/patching/localrepo/dme-2.0.1.0-9.3.1.lib32_n9000.rpm**

例：

```
bash-4.3$ ls /bootflash/.rpmstore/patching/localrepo/dme-2.0.1.0-9.3.1.lib32_n9000.rpm
inactive_feature_rpms.inf
repodata

bash-4.3$
```

ベース DME RPM がインストールされている場合は、`/rpms` にあります。

DME RPM のバージョン間ダウングレード

NX-OS コマンドまたは を使用して、DME RPM の上位バージョンから下位バージョンにダウングレードできます。**installdnf** ダウングレードすることで、DME のモジュラリティ機能が保持されます。

DME RPM は保護されているため、サポートされていません。**install deactivateinstall remove**

手順

ダウングレード方法を選択します。

NX-OS の場合：

- **install add** *dme-rpm*へのパス **activate downgrade**

  例：

```
switch-1# install add bootflash:dme-2.0.1.0-9.3.1.lib32_n9000.rpm activate downgrade
Adding the patch (/dme-2.0.1.0-9.3.1.lib32_n9000.rpm)
[####################] 100%
Install operation 94 completed successfully at Fri Jun  7 22:48:34 2019

Activating the patch (/dme-2.0.1.0-9.3.1.lib32_n9000.rpm)
[####################] 100%
Install operation 95 completed successfully at Fri Jun  7 22:49:12 2019
switch-1#
```

- **show install active | include dme**

  例：

```
switch-1# show install active | include dme
        dme-2.0.1.0-9.3.1.lib32_n9000
switch-1#
```

この例では、DME RPM がバージョン 2.0.1.0 ～ 9.3.1 にダウングレードされています。

dnf の場合、root ユーザー（**run bash sudo su**）として Bash シェルでコマンドを実行する必要があります。

- Bash で、**dnf downgrade dme** *dme-rpm* を実行します。

  このオプションを使用すると、リポジトリ内の下位バージョンの DME RPM に直接ダウンロードできます。

  次のコマンド出力で強調表示されているように、このオプションオプションを完了するには、ユーザーの介入が必要です。

  例：

```
bash-4.3# dnf downgrade dme 2.0.1.0-9.3.1
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 ...
patching                                                 |  951 B     00:00 ...
thirdparty                                               |  951 B     00:00 ...
wrl-repo                                                 |  951 B     00:00 ...
Resolving Dependencies
--> Running transaction check
---> Package dme.lib32_n9000 0:2.0.1.0-9.3.1 will be a downgrade
---> Package dme.lib32_n9000 0:2.0.10.0-9.3.1 will be erased
--> Finished Dependency Resolution

Dependencies Resolved
================================================================================
 Package     Arch                Version                  Repository       Size
```

```
================================================================================
Downgrading:
 dme         lib32_n9000         2.0.10.0-9.3.1            localdb          45 M

Transaction Summary
================================================================================
Downgrade     1 Package

Total download size: 45 M
Is this ok [y/N]: y
Retrieving key from file:///etc/pki/rpm-gpg/arm-Nexus9k-dev.gpg
Downloading Packages:
Running Transaction Check
Running Transaction Test
Transaction Test Succeeded
Running Transaction
/bootflash/.rpmstore/config/etc/pki/rpm-gpg/arm-Nexus9k-dev.gpg
System at HA Standby, running transaction on Standby first
  Installing : dme-2.0.1.0-9.3.1.lib32_n9000                                1/2
starting pre-install package version mgmt for dme
pre-install for dme complete
ln: failed to create symbolic link /var/run/mgmt/sharedmeta-hash: File exists
ln: failed to create symbolic link /var/run/mgmt/dme-objstores.conf: File exists
ln: failed to create symbolic link /var/run/mgmt/samlog.config: File exists
mgmt/
mgmt/shmetafiles/
mgmt/shmetafiles/sharedmeta-ArgMetaData
mgmt/shmetafiles/sharedmeta-RelsMetaData
mgmt/shmetafiles/sharedmeta-ClassRelMetaData
mgmt/shmetafiles/sharedmeta-ChunkMetaData
mgmt/shmetafiles/sharedmeta-ConstPropMetaData
mgmt/shmetafiles/sharedmeta-ConstIdMetaData
mgmt/shmetafiles/sharedmeta-ClassMetaData
mgmt/shmetafiles/sharedmeta-PropRefsMetaData
mgmt/shmetafiles/sharedmeta-SvcMetaData
mgmt/shmetafiles/sharedmeta-ActionContextMetaData
mgmt/shmetafiles/sharedmeta-ConstDefTypeMetaData
mgmt/shmetafiles/sharedmeta-ConstArgMetaData
mgmt/shmetafiles/sharedmeta-ClassNamingMetaData
mgmt/shmetafiles/sharedmeta-ConstMetaData
mgmt/shmetafiles/sharedmeta-PropMetaData
mgmt/shmetafiles/sharedmeta-DnMetaData
  Cleanup    : dme-2.0.10.0-9.3.1.lib32_n9000                               2/2

Removed:
  dme.lib32_n9000 0:2.0.10.0-9.3.1

Installed:
  dme.lib32_n9000 0:2.0.1.0-9.3.1

Complete!
```

DME RPM の 1 つのバージョンから下位のバージョンにダウングレードします。この例では、バージョン 2.0.10.0 〜 9.3.1 がバージョン 2.0.1.0 〜 9.3.1 にダウングレードされます。

- **dnf list --patch-only installed | grep dme**

  例：

```
bash-4.3# dnf list --patch-only installed | grep dme
dme.lib32_n9000                        2.0.1.0-9.3.1                @groups-repo
bash-4.3#
```

インストールされている DME RPM のバージョンを表示します。

ベース RPM へのダウングレード

NX-OS の **install** コマンドを使用してベース DME RPM をインストールするか、または **dnf downgrade** を使用して、上位バージョンの DME RPM からベース DME RPM にダウングレードできます。

手順

ダウングレード方法を選択します。

NX-OS の場合：

- **install activate** *dme-rpm*

  例：

```
switch-1# install activate dme-2.0.0.0-9.2.1.lib32_n9000.rpm
[####################] 100%
Install operation 89 completed successfully at Fri Jun  7 07:21:45 2019
switch-1#
```

- **show install active | dme**

  例：

```
switch-1# show install active | include dme
        dme-2.0.0.0-9.2.1.lib32_n9000
switch-1#
```

dnf の場合、root ユーザー（**run bash sudo su**）として Bash シェルでコマンドを実行する必要があります。

- Bash で、**dnf downgrade dme** *dme-rpm* を実行します。

  このオプションにより、ベース DME RPM に直接ダウングレードできます。

  次のコマンド出力で強調表示されているように、このオプションを完了するには、ユーザーの介入が必要です。

  例：

```
bash-4.3# dnf downgrade dme-2.0.0.0-9.3.1.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 ...
patching                                                 |  951 B     00:00 ...
thirdparty                                               |  951 B     00:00 ...
wrl-repo                                                 |  951 B     00:00 ...
Resolving Dependencies
--> Running transaction check
---> Package dme.lib32_n9000 0:2.0.0.0-9.3.1 will be a downgrade
---> Package dme.lib32_n9000 0:2.0.10.0-9.3.1 will be erased
--> Finished Dependency Resolution
```

```
Dependencies Resolved

================================================================================
Package    Arch               Version                 Repository          Size
================================================================================
Downgrading:
dme        lib32_n9000        2.0.0.0-9.3.1           groups-repo         44 M

Transaction Summary
================================================================================
Downgrade     1 Package

Total download size: 44 M
Is this ok [y/N]: y
Downloading Packages:
Running Transaction Check
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing : dme-2.0.0.0-9.3.1.lib32_n9000                                1/2
starting pre-install package version mgmt for dme
pre-install for dme complete
mgmt/
mgmt/shmetafiles/
mgmt/shmetafiles/sharedmeta-ChunkMetaData
mgmt/shmetafiles/sharedmeta-ClassMetaData
mgmt/shmetafiles/sharedmeta-ArgMetaData
mgmt/shmetafiles/sharedmeta-ConstMetaData
mgmt/shmetafiles/sharedmeta-ConstIdMetaData
mgmt/shmetafiles/sharedmeta-ConstDefTypeMetaData
mgmt/shmetafiles/sharedmeta-ConstPropMetaData
mgmt/shmetafiles/sharedmeta-ConstArgMetaData
mgmt/shmetafiles/sharedmeta-ClassRelMetaData
mgmt/shmetafiles/sharedmeta-DnMetaData
mgmt/shmetafiles/sharedmeta-PropRefsMetaData
mgmt/shmetafiles/sharedmeta-PropMetaData
mgmt/shmetafiles/sharedmeta-RelsMetaData
mgmt/shmetafiles/sharedmeta-ActionContextMetaData
mgmt/shmetafiles/sharedmeta-SvcMetaData
mgmt/shmetafiles/sharedmeta-ClassNamingMetaData
  Cleanup    : dme-2.0.10.0-9.3.1.lib32_n9000                               2/2

Removed:
  dme.lib32_n9000 0:2.0.10.0-9.3.1

Installed:
  dme.lib32_n9000 0:2.0.0.0-9.3.1

Complete!
bash-4.3#
```

ベース DME RPM をインストールします。

- **dnf list --patch-only installed | grep dme**

例:

```
bash-4.3# dnf list --patch-only installed | grep dme
dme.lib32_n9000                        2.0.0.0-9.3.1                @groups-repo
bash-4.3#
```

インストールされているベース DME RPM を表示します。

パッチ RPM の管理

RPM インストールの前提条件

RPM をインストールまたは追加する前に、次の手順によりシステムの準備ができていることを確認します。

手順の概要

1. switch# **show logging logfile | grep -i "System ready"**
2. switch# **run bash sudo su**

手順の詳細

手順

	コマンドまたはアクション	目的
Step 1	switch# **show logging logfile | grep -i "System ready"**	Bash を実行する前に、この手順によって、RPM をインストールまたは追加する前のシステムの準備ができていることを確認します。 以下のような出力が表示されれば、続行します。 **2018 Mar 27 17:24:22 switch %ASCII-CFG-2-CONF_CONTROL: System ready**

Step 2	switch# **run bash sudo su**
	例:
	`switch# run bash sudo su`
	`bash-4.2#`
		Bash をロードします。

Bash からパッチ RPM を追加する

手順

	コマンドまたはアクション	目的
Step 1	**dnf list --patch-only**	スイッチに存在するパッチRPMのリストを表示します。
Step 2	**sudo dnf install --add** *URL_of_patch*	リポジトリにパッチを追加します。ここで、URL_of_patch は、/bootflash/ patch などの標準的な Linux 形式ではなく、bootflash:/ patch などの明確に定義された形式です。
Step 3	**dnf list --patch-only available**	リポジトリに追加されているが非アクティブ状態のパッチのリストを表示します。

例

次に、RPM をインストールする例を示します。
nxos.CSCab00001-n9k_ALL-1.0.0-7.0.3.I7.3.lib32_n9000

```
bash-4.2# dnf list --patch-only
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 ...
bash-4.2#
bash-4.2# sudo dnf install --add
bootflash:/nxos.CSCab00001-n9k_ALL-1.0.0-7.0.3.I7.3.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 ...
[##############     ] 70%Install operation 135 completed successfully at Tue Mar 27 17:45:34
 2018.

[####################] 100%
bash-4.2#
```

パッチ RPM がインストールされたら、正しくインストールされたことを確認します。次のコマンドは、リポジトリに追加され、非アクティブ状態のパッチを一覧表示します。

```
bash-4.2# dnf list --patch-only available
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 ...
```

```
nxos.CSCab00001-n9k_ALL.lib32_n9000    1.0.0-7.0.3.I7.3    patching
bash-4.2#
```

RPM が tar ファイルにバンドルされている tar ファイルからリポジトリにパッチを追加することもできます。次に、nxos.CSCab00002_CSCab00003-n9k_ALL-1.0.0-7.0.3.I7.3.lib32_n9000 tar ファイルに含まれる 2 つの RPM をパッチリポジトリに追加する例を示します。

```
bash-4.2# sudo dnf install --add
bootflash:/nxos.CSCab00002_CSCab00003-n9k_ALL-1.0.0-7.0.3.I7.3.lib32_n9000.tar
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 ...
[##############     ] 70%Install operation 146 completed successfully at Tue Mar 27 21:17:39
 2018.

[####################] 100%
bash-4.2#
bash-4.2# dnf list --patch-only
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 ...
patching/primary                                             |  942 B     00:00 ...
patching                                                                      2/2
thirdparty                                                   |  951 B     00:00 ...
nxos.CSCab00003-n9k_ALL.lib32_n9000    1.0.0-7.0.3.I7.3    patching
nxos.CSCab00002-n9k_ALL.lib32_n9000    1.0.0-7.0.3.I7.3    patching
bash-4.2#
```

パッチ RPM のアクティブ化

始める前に

#unique_61 の手順に従って、必要なパッチ RPM がリポジトリに追加されていることを確認します。

手順

	コマンドまたはアクション	目的
Step 1	**sudo dnf install** *patch_RPM* **--nocommit**	パッチ RPM をアクティブにします。ここで、*patch_RPM* はリポジトリにあるパッチです。この手順では、パッチの場所を指定しないでください。 （注） --nocommit フラグをコマンドに追加すると、パッチ RPM がこの手順でアクティブになりますが、コミットされません。パッチ RPM をアクティブ化した後

	コマンドまたはアクション	目的
		にコミットする手順については、パッチ RPM のコミット （23 ページ） を参照してください。

例

次に、**nxos.CSCab00001-n9k_ALL-1.0.0-7.0.3.I7.3.lib32_n9000** パッチ RPM をアクティブにする例を示します。

```
bash-4.2# sudo dnf install nxos.CSCab00001-n9k_ALL-1.0.0-7.0.3.I7.3.lib32_n9000 --nocommit
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
Resolving Dependencies
--> Running transaction check
---> Package nxos.CSCab00001-n9k_ALL.lib32_n9000 0:1.0.0-7.0.3.I7.3 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package                    Arch           Version               Repository   Size
================================================================================
Installing:
 nxos.CSCab00001-n9k_ALL    lib32_n9000    1.0.0-7.0.3.I7.3      patching     28 k

Transaction Summary
================================================================================
Install       1 Package

Total download size: 28 k
Installed size: 82 k
Is this ok [y/N]: y
Downloading Packages:
Running Transaction Check
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing : nxos.CSCab00001-n9k_ALL-1.0.0-7.0.3.I7.3.lib32_n9000            1/1
[##################  ] 90%error: reading
/var/sysmgr/tmp/patches/CSCab00001-n9k_ALL/isan/bin/sysinfo manifest, non-printable characters
 found

Installed:
  nxos.CSCab00001-n9k_ALL.lib32_n9000 0:1.0.0-7.0.3.I7.3

Complete!
Install operation 140 completed successfully at Tue Mar 27 18:07:40 2018.

[####################] 100%
bash-4.2#
```

次のコマンドを入力して、パッチ RPM が正常にアクティブ化されたことを確認します。

```
bash-4.2# dnf list --patch-only
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 ...
nxos.CSCab00001-n9k_ALL.lib32_n9000    1.0.0-7.0.3.I7.3    installed
bash-4.2#
```

パッチ RPM のコミット

手順

	コマンドまたはアクション	目的
Step 1	**sudo dnf install** *patch_RPM* **--commit**	パッチ RPM をコミットします。パッチ RPM は、リロード後もアクティブな状態を維持するためにコミットする必要があります。

例

次に、パッチ RPM をコミットする例を示します。
nxos.CSCab00001-n9k_ALL-1.0.0-7.0.3.I7.3.lib32_n9000

```
bash-4.2# sudo dnf install nxos.CSCab00001-n9k_ALL-1.0.0-7.0.3.I7.3.lib32_n9000 --commit
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 ...
Install operation 142 completed successfully at Tue Mar 27 18:13:16 2018.

[####################] 100%
bash-4.2#
```

次のコマンドを入力して、パッチ RPM が正常にコミットされたことを確認します。

```
bash-4.2# dnf list --patch-only committed
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 ...
nxos.CSCab00001-n9k_ALL.lib32_n9000    1.0.0-7.0.3.I7.3    installed
bash-4.2#
```

パッチ RPM の非アクティブ化

手順

	コマンドまたはアクション	目的
Step 1	**sudo dnf erase** *patch_RPM* **--nocommit**	パッチ RPM を非アクティブ化します。 （注） コマンドに --nocommit フラグを追加すると、パッチ RPM はこの手順でのみ非アクティブ化されます。
Step 2	**sudo dnf install** *patch_RPM* **--commit**	パッチ RPM をコミットします。パッチ RPM をコミットしないまま削除しようとすると、エラーメッセージが表示されます。

例

次に、**nxos.CSCab00001-n9k_ALL-1.0.0-7.0.3.I7.3.lib32_n9000** パッチ RPM を非アクティブにする例を示します。

```
bash-4.2# sudo dnf erase nxos.CSCab00001-n9k_ALL-1.0.0-7.0.3.I7.3.lib32_n9000 --nocommit
Loaded plugins: downloadonly, importpubkey, localrpmDB, patchaction, patching,
              : protect-packages
Setting up Remove Process
Resolving Dependencies
--> Running transaction check
---> Package nxos.CSCab00001-n9k_ALL.lib32_n9000 0:1.0.0-7.0.3.I7.3 will be erased
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package                     Arch           Version              Repository    Size
================================================================================
Removing:
 nxos.CSCab00001-n9k_ALL    lib32_n9000    1.0.0-7.0.3.I7.3     @patching     82 k

Transaction Summary
================================================================================
Remove        1 Package

Installed size: 82 k
Is this ok [y/N]: y
Downloading Packages:
Running Transaction Check
Running Transaction Test
Transaction Test Succeeded
Running Transaction
[######              ] 30%error: reading
/var/sysmgr/tmp/patches/CSCab00001-n9k_ALL/isan/bin/sysinfo manifest, non-printable characters
 found
  Erasing    : nxos.CSCab00001-n9k_ALL-1.0.0-7.0.3.I7.3.lib32_n9000            1/1
[##################  ] 90%
Removed:
```

```
  nxos.CSCab00001-n9k_ALL.lib32_n9000 0:1.0.0-7.0.3.I7.3

Complete!
Install operation 143 completed successfully at Tue Mar 27 21:03:47 2018.

[####################] 100%
bash-4.2#
```

パッチ RPM は、非アクティブ化した後にコミットする必要があります。パッチ RPM を非アクティブ化した後にコミットしなかった場合に、パッチ RPM の削除 （25 ページ）の手順を使用してパッチ RPM を削除しようとすると、エラー メッセージが表示されます。

```
bash-4.2# sudo dnf install nxos.CSCab00001-n9k_ALL-1.0.0-7.0.3.I7.3.lib32_n9000 --commit
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 ...
Install operation 144 completed successfully at Tue Mar 27 21:09:28 2018.

[####################] 100%
bash-4.2#
```

次のコマンドを入力して、パッチ RPM が正常にコミットされたことを確認します。

```
bash-4.2# dnf list --patch-only
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 ...
nxos.CSCab00001-n9k_ALL.lib32_n9000    1.0.0-7.0.3.I7.3    patching
bash-4.2#
```

パッチ RPM の削除

手順

	コマンドまたはアクション	目的
Step 1	**sudo dnf install --remove** *patch_RPM*	非アクティブなパッチ RPM を削除します。

例

次に、パッチ RPM を削除する例を示します。
nxos.CSCab00001-n9k_ALL-1.0.0-7.0.3.I7.3.lib32_n9000

```
bash-4.2# sudo dnf install --remove nxos.CSCab00001-n9k_ALL-1.0.0-7.0.3.I7.3.lib32_n9000
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 ...
[##########        ] 50%Install operation 145 completed successfully at Tue Mar 27 21:11:05
 2018.

[####################] 100%
bash-4.2#
```

（注） パッチ RPM を削除しようとした後、次のエラーメッセージが表示されます。

Install operation 11 "failed because patch was not committed". at Wed Mar 28 22:14:05 2018

その後、削除を試みる前にパッチ RPM をコミットしませんでした。パッチ RPM を削除する前にコミットする手順については、を参照してください。パッチ RPM の非アクティブ化 （24 ページ）

次のコマンドを入力して、非アクティブなパッチ RPM が正常に削除されたことを確認します。

```
bash-4.2# dnf list --patch-only
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 ...
patching/primary                                          |  197 B     00:00 ...
thirdparty                                                |  951 B     00:00 ...
bash-4.2#
```

SDK または ISO で構築されたサード パーティ プロセスの永続的なデーモン化

アプリケーションには、/etc/init.d/*application_name* にインストールされる起動 Bash スクリプトが必要です。この起動Bashスクリプトは、次の一般的なフォーマットにする必要があります（このフォーマットの詳細については、http://linux.die.net/man/8/chkconfigを参照してください）。

```
#!/bin/bash
#
# <application_name> Short description of your application
#
# chkconfig: 2345 15 85
# description: Short description of your application
#
### BEGIN INIT INFO
# Provides: <application_name>
# Required-Start: $local_fs $remote_fs $network $named
# Required-Stop: $local_fs $remote_fs $network
# Description: Short description of your application
### END INIT INFO
# See how we were called.
case "$1" in
```

```
start)
# Put your startup commands here
# Set RETVAL to 0 for success, non-0 for failure
;;
stop)
# Put your stop commands here
# Set RETVAL to 0 for success, non-0 for failure
;;
status)
# Put your status commands here
# Set RETVAL to 0 for success, non-0 for failure
;;
restart|force-reload|reload)
# Put your restart commands here
# Set RETVAL to 0 for success, non-0 for failure
;;
*)
echo $"Usage: $prog {start|stop|status|restart|force-reload}"
RETVAL=2
esac

exit $RETVAL
```

ネイティブBashシェルからのアプリケーションの永続的な起動

手順

Step 1 作成したアプリケーション起動 Bash スクリプトを /etc/init.d/application_name にインストールします。

Step 2 `/etc/init.d/`*application_name* start でアプリケーションを開始します

Step 3 **chkconfig --add** *application_name*を入力します

Step 4 **chkconfig --level 3** *application_name* **on**を入力します

実行レベル 3 は、標準のマルチユーザ実行レベルであり、スイッチが通常実行されるレベルです。

Step 5 -- application_name を実行して、アプリケーションがレベル 3 で実行されるようにスケジュールされていることを確認し、レベル 3 が on に設定されていることを確認します。**chkconfiglist**

Step 6 アプリケーションが /etc/rc3.d にリストされていることを確認します。「S」の後に数字が続き、アプリケーション名（この例では tcollector）が続き、../init.d/application_name に Bash 起動スクリプトへのリンクが表示されます。

bash-4.2# ls -l /etc/rc3.d/tcollector

lrwxrwxrwx 1 root root 20 Sep 25 22:56 /etc/rc3.d/S15tcollector ->../init.d/tcollector

bash-4.2#

現用系ブートフラッシュからスタンバイ ブートフラッシュへのファイルの同期

Cisco Nexus 9500 プラットフォーム スイッチは、通常、高可用性を提供するために 2 つのスーパーバイザ モジュール（1 つの現用系スーパーバイザ モジュールと 1 つのスタンバイ スーパーバイザ モジュール）で構成されています。各スーパーバイザモジュールには、ファイルストレージ用の独自のブートフラッシュ ファイル システムがあり、通常、現用系ブートフラッシュ ファイル システムとスタンバイ ブートフラッシュ ファイル システムは互いに独立しています。現用系ブートフラッシュに特定のコンテンツが必要な場合、将来スイッチオーバーが発生した場合に備えて、同じコンテンツがスタンバイ ブートフラッシュにも必要でしょう。

Cisco NX-OS 9.2(2) リリースより前は、現用系スーパーバイザ モジュールとスタンバイ スーパーバイザ モジュール間でこのようなコンテンツを手動で管理する必要がありました。Cisco NX-OS 9.2(2) 以降では、スタンバイ スーパーバイザ モジュールが up 状態で使用可能なら、現用系スーパーバイザ モジュールまたは現用系ブートフラッシュ（`/bootflash`）上の特定のファイルとディレクトリを、スタンバイ スーパーバイザ モジュールまたはスタンバイ ブートフラッシュ（`/bootflash_sup-remote`）に自動的に同期できます。同期するファイルとディレクトリを選択するには、スイッチに Bash をロードし、現用系ブートフラッシュからスタンバイ ブートフラッシュに同期するファイルとディレクトリを、編集可能ファイル `/bootflash/bootflash_sync_list` に追加します。

次に例を示します。

```
switch# run bash
bash-4.2# echo "/bootflash/home/admin" | sudo tee --append /bootflash/bootflash_sync_list
bash-4.2# echo "/bootflash/nxos.7.0.3.I7.3.5.bin" | sudo tee --append
/bootflash/bootflash_sync_list
bash-4.2# cat /bootflash/bootflash_sync_list
/bootflash/home/admin
/bootflash/nxos.7.0.3.I7.3.5.bin
```

現用系ブートフラッシュのファイルまたはディレクトリに変更が加えられた場合、スタンバイ ブートフラッシュが up 状態で使用可能なら、これらの変更はスタンバイ ブートフラッシュに自動的に同期されます。スタンバイ ブートフラッシュが通常のブート、スイッチオーバー、または手動スタンバイ リロードのいずれかでリブートされると、スタンバイ スーパーバイザがオンラインになったとき、現用系ブートフラッシュへの変更のキャッチアップ同期がスタンバイ ブートフラッシュにプッシュされます。

次に、編集可能な `/bootflash/bootflash_sync_list` ファイルの特性と制限事項を示します。

- `/bootflash/bootflash_sync_list` ファイルは、最初の実行時に自動的に作成されますが、最初の作成状態では空です。
- `/bootflash/bootflash_sync_list` ファイルのエントリは、次の注意事項に従います。
  - 1 行に 1 エントリ

- エントリは Linux パスとして指定します（例：/bootflash/img.bin）
- エントリは /bootflash ファイルシステム内にある必要があります

- /bootflash/bootflash_sync_list ファイル自体は、自動的にスタンバイ ブートフラッシュに同期されます。**copy** virtual shell（VSH）コマンドを使用して、スーパーバイザモジュールとの間で /bootflash/bootflash_sync_list ファイルを手動でコピーすることもできます。
- 次のコマンドを使用して、スーパーバイザ モジュールで直接 /bootflash/bootflash_sync_list ファイルを編集できます。

  run bash vi /bootflash/bootflash_sync_list

同期イベントからのすべての出力は、ログファイル /var/tmp/bootflash_sync.log にリダイレクトされます。次のいずれかのコマンドを使用して、このログファイルを表示または追跡できます。

run bash less /var/tmp/bootflash_sync.log

run bash tail -f /var/tmp/bootflash_sync.log

同期スクリプトは、現用系ブートフラッシュ ディレクトリ上の対応するファイルの削除イベントを明示的に受信しない限り、スタンバイブートフラッシュディレクトリからファイルを削除しません。場合によっては、スタンバイ ブートフラッシュの使用中のスペースが現用系ブートフラッシュよりも多くなり、現用系ブートフラッシュと同期しているときにスタンバイ ブートフラッシュのスペースが不足することがあります。スタンバイ ブートフラッシュを現用系ブートフラッシュの正確なミラーにする（スタンバイ ブートフラッシュ上の余分なファイルを削除する）には、次のコマンドを入力します。

run bash sudo rsync -a --delete /bootflash/ /bootflash_sup-remote/

同期スクリプトは、クラッシュまたは終了することなく、バックグラウンドで実行され続ける必要があります。ただし、何らかの理由で実行が停止した場合は、次のコマンドを使用して手動で再起動できます。

run bash sudo /isan/etc/rc.d/rc.isan-start/S98bootflash_sync.sh start

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。

ネイティブ Bash シェルのアプリケーション例

次の例は、ネイティブ Bash シェルのアプリケーションを示しています：

```
bash-4.2# cat /etc/init.d/hello.sh
#!/bin/bash

PIDFILE=/tmp/hello.pid
OUTPUTFILE=/tmp/hello

echo $$ > $PIDFILE
rm -f $OUTPUTFILE
while true
do
    echo $(date) >> $OUTPUTFILE
    echo 'Hello World' >> $OUTPUTFILE
    sleep 10
done
bash-4.2#
bash-4.2#
bash-4.2# cat /etc/init.d/hello
#!/bin/bash
#
# hello Trivial "hello world" example Third Party App
#
# chkconfig: 2345 15 85
# description: Trivial example Third Party App
#
### BEGIN INIT INFO
# Provides: hello
# Required-Start: $local_fs $remote_fs $network $named
# Required-Stop: $local_fs $remote_fs $network
# Description: Trivial example Third Party App
```

```
### END INIT INFO

PIDFILE=/tmp/hello.pid

# See how we were called.
case "$1" in
start)
    /etc/init.d/hello.sh &
    RETVAL=$?
;;
stop)
    kill -9 `cat $PIDFILE`
    RETVAL=$?
;;
status)
    ps -p `cat $PIDFILE`
    RETVAL=$?
;;
restart|force-reload|reload)
    kill -9 `cat $PIDFILE`
    /etc/init.d/hello.sh &
    RETVAL=$?
;;
*)
echo $"Usage: $prog {start|stop|status|restart|force-reload}"
RETVAL=2
esac

exit $RETVAL
bash-4.2#
bash-4.2# chkconfig --add hello
bash-4.2# chkconfig --level 3 hello on
bash-4.2# chkconfig --list hello
hello           0:off   1:off   2:on    3:on    4:on    5:on    6:off
bash-4.2# ls -al /etc/rc3.d/*hello*
lrwxrwxrwx 1 root root 15 Sep 27 18:00 /etc/rc3.d/S15hello -> ../init.d/hello
bash-4.2#
bash-4.2# reboot
```

リロード後

```
bash-4.2# ps -ef | grep hello
root      8790     1  0 18:03 ?        00:00:00 /bin/bash /etc/init.d/hello.sh
root      8973  8775  0 18:04 ttyS0    00:00:00 grep hello
bash-4.2#
bash-4.2# ls -al /tmp/hello*
-rw-rw-rw- 1 root root 205 Sep 27 18:04 /tmp/hello
-rw-rw-rw- 1 root root   5 Sep 27 18:03 /tmp/hello.pid
bash-4.2# cat /tmp/hello.pid
8790
bash-4.2# cat /tmp/hello
Sun Sep 27 18:03:49 UTC 2015
Hello World
Sun Sep 27 18:03:59 UTC 2015
Hello World
Sun Sep 27 18:04:09 UTC 2015
Hello World
Sun Sep 27 18:04:19 UTC 2015
Hello World
Sun Sep 27 18:04:29 UTC 2015
Hello World
Sun Sep 27 18:04:39 UTC 2015
Hello World
bash-4.2#
```

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。