ゲスト シェル

Guest Shell について

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

ゲスト シェル内から、network-admin には次の機能があります。

- Linux ネットワーク インターフェイスを介したネットワークへのアクセス。
- スイッチのブートフラッシュへのアクセス。
- スイッチの揮発性 tmpfs へのアクセス。
- スイッチの CLI へのアクセス。
- Cisco NX-API REST へのアクセス。
- Python スクリプトをインストールして実行する機能。
- 32 ビットおよび 64 ビットの Linux アプリケーションをインストールして実行する機能。

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

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

Guest Shell へのアクセス

Cisco NX-OS では、Guest Shell にはネットワーク管理者がアクセスできます。システムで自動的に有効になり、**run guestshell** コマンドを使用してアクセスできます。**run bash** コマンドと一致して、これらのコマンドは、NX-OS CLI コマンドの **run guestshell** コマンド形式を使用して Guest Shell 内で発行できます。

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

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

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

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

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

ゲスト シェルに使用されるリソース

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

リソース	デフォルト	最小/最大
CPU	1 %	1/%

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

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

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

ゲストシェルの機能

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

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

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

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

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

Guest Shell の NX-OS CLI

ゲスト シェルは、ユーザーがゲスト シェル環境からホスト ネットワーク要素に NX-OS コマンドを発行できるようにするアプリケーションを提供します。**dohost** アプリケーションは、有

効な NX-OS 構成または exec コマンドを受け入れ、それらをホスト ネットワーク要素に発行します。

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

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

NX-OS CLI は連鎖させることができます：

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

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

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

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

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

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

Guest Shell でのネットワーク アクセス

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

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

```
[guestshell@guestshell bootflash]$ ifconfig Eth1-47
Eth1-47: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 13.0.0.47 netmask 255.255.255.0 broadcast 13.0.0.255
```

```
ether 54:7f:ee:8e:27:bc txqueuelen 100 (Ethernet)
RX packets 311442 bytes 21703008 (20.6 MiB)
RX errors 0 dropped 185 overruns 0 frame 0
TX packets 12967 bytes 3023575 (2.8 MiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

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

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

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

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

chvrf ユーティリティは便宜のために提供されています。

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

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

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

例：

```
switch# guestshell
[guestshell@guestshell ~]$ cd /bootflash
[guestshell@guestshell bootflash]$ chvrf management scp foo@10.28.38.48:/foo/index.html
 index.html
foo@10.28.38.48's password:
index.html 100% 1804 1.8KB/s 00:00
[guestshell@guestshell bootflash]$ ls -al index.html
-rw-r--r-- 1 guestshe users 1804 Sep 13 20:28 index.html
[guestshell@guestshell bootflash]$
[guestshell@guestshell bootflash]$ chvrf management curl cisco.com
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>301 Moved Permanently</title>
</head><body>
<h1>Moved Permanently</h1>
<p>The document has moved <a href="http://www.cisco.com/">here</a>.</p>
</body></html>
[guestshell@guestshell bootflash]$
```

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

例：

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

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

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

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

例：

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

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

例：

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

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

例：

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

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

ゲスト シェルでのブートフラッシュへのアクセス

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

例：

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

Guest Shell の Python

Python はインタラクティブに使用できますが、python スクリプトをゲスト シェルで実行することもできます。

例：

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

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

例：

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

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

Installing RPMs in the Guest Shell

The /etc/yum.repos.d/CentOS-Base.repo file is set up to use the CentOS mirror list by default. Follow instructions in that file if changes are needed.

Yum can be pointed to one or more repositories at any time by modifying the `yumrepo_x86_64.repo` file or by adding a new `.repo` file in the `repos.d` directory.

For applications to be installed inside Guest Shell 2.x, go to the CentOS 7 repo at http://mirror.centos.org/centos/7/os/x86_64/Packages/.

Yum resolves the dependencies and installs all the required packages.

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

Dependencies Resolved

================================================================================
Package Arch Version Repository Size
================================================================================
Installing:
glibc i686 2.17-78.el7 base 4.2 M
Installing for dependencies:
nss-softokn-freebl i686 3.16.2.3-9.el7 base 187 k

Transaction Summary
================================================================================
Install 1 Package (+1 Dependent package)

Total download size: 4.4 M
Installed size: 15 M
Downloading packages:
Delta RPMs disabled because /usr/bin/applydeltarpm not installed.
(1/2): nss-softokn-freebl-3.16.2.3-9.el7.i686.rpm | 187 kB 00:00:25
(2/2): glibc-2.17-78.el7.i686.rpm | 4.2 MB 00:00:30
--------------------------------------------------------------------------------
Total 145 kB/s | 4.4 MB 00:00:30
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
Installing : nss-softokn-freebl-3.16.2.3-9.el7.i686 1/2
Installing : glibc-2.17-78.el7.i686 2/2
error: lua script failed: [string "%triggerin(glibc-common-2.17-78.el7.x86_64)"]:1:
attempt to compare number with nil
Non-fatal "<"unknown">" scriptlet failure in rpm package glibc-2.17-78.el7.i686
Verifying : glibc-2.17-78.el7.i686 1/2
Verifying : nss-softokn-freebl-3.16.2.3-9.el7.i686 2/2

Installed:
glibc.i686 0:2.17-78.el7

Dependency Installed:
nss-softokn-freebl.i686 0:3.16.2.3-9.el7
```

```
Complete!
```

Note When more space is needed in the Guest Shell root file system for installing or running packages, the **guestshell resize roofs** *size-in-MB* command is used to increase the size of the file system.

Note Some open source software packages from the repository might not install or run as expected in the Guest Shell as a result of restrictions that have been put into place to protect the integrity of the host system.

のセキュリティ ポスチャ

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

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

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

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

ルートユーザーの制限

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

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

内のルートに対してドロップされる一連の Linux 機能は次のとおりです。

リソース管理

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

ゲスト ファイル システムのアクセス制限

セキュア IPC

ゲスト シェルまたは仮想サービス内のアプリケーションは、Cisco onePK サービスを使用してホストとより統合できます。アプリケーションは、TIPC を介してホスト ネットワーク要素と通信します。さまざまなコンテナー内のアプリケーションは、TIPC を介して相互に通信することはできません。ホストとの通話のみが許可されます。これにより、1 つのコンテナの問題が、それが Cisco onePK サービスが実行されている場所であるとスプーフィングを防ぎます。コンテナ内のアプリケーションは、TIPC ポートでリッスンすることもできません。

既知の仮想サービスのみがホストと通信できるようにするために、各仮想サービスの一意の識別子は、有効化時に作成され、onePK 通信チャネルが確立されるときに検証されます。

システムは、個々の仮想サービス内のアプリケーションがホストにメッセージを送信できるレートも制限します。この動作により、不正な動作をするアプリケーションがメッセージを頻繁に送信して、ホストの通常の操作を妨げたり、同じホスト上の他の仮想サービスがホストと通信するのをブロックしたりすることが防止されます。

ゲスト シェルの管理

以下は、ゲスト シェルを管理するためのコマンドです。

表 *1:* ゲスト シェル *CLI* コマンド

コマンド	説明

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

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

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

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

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

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

Guest Shell の無効化

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

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

例：

```
switch# show virtual-service list
Virtual Service List:
Name                      Status          Package Name
------------------------------------------------------------
guestshell+               Activated        guestshe11.ova
switch# guestshell disable
You will not be able to access your guest shell if it is disabled. Are you sure you want
 to disable the guest shell? (y/n) [n) y

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

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

ゲスト シェルの破棄

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

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

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

```
switch# guestshell destroy

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

 'guestshell+'
2014 Jul 30 18:49:10 switch %$ VDC-1 %$ %VMAN-2-INSTALL_STATE: Successfully destroyed
virtual service 'guestshell +'

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

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

（注） Cisco NX-OS ソフトウェアでは、コンテナがインストールされると、**oneP** 機能がローカル アクセスに対して自動的に有効になります。ゲスト シェルはコンテナであるため、**oneP** 機能が自動的に開始されます。

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

Guest Shell の有効化

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

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

例：

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

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

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

仮想サービスと Guest Shell 情報の検証

次のコマンドを使用して、仮想サービスとゲスト シェルの情報を検証できます。

コマンド	説明

show virtual-service global
`switch# show virtual-service global`
 	
`Virtual Service Global State and Virtualization Limits:`
 	
`Infrastructure version : 1.11`
`Total virtual services installed : 1`
`Total virtual services activated : 1`
 	
`Machine types supported : LXC`
`Machine types disabled : KVM`
 	
`Maximum VCPUs per virtual service : 1`
 	
`Resource virtualization limits:`
`Name Quota Committed Available`
`---`
`system CPU (%) 20 1 19`
`memory (MB) 3840 256 3584`
`bootflash (MB) 8192 200 7992`
`switch#`
	仮想サービスのグローバル状態と制限を表示します。

show virtual-service list
 	
`switch# show virtual-service list *`
 	
`Virtual Service List:`
 	
`Name Status Package Name`
`---`
`guestshell+ Activated guestshell.ova`
	仮想サービスの概要、仮想サービスのステータス、およびインストールされているソフトウェア パッケージを表示します。

コマンド	説明

show guestshell detail
 	
`switch# show guestshell detail`
`Virtual service guestshell+ detail`
`  State                 : Activated`
`  Package information`
`    Name                : guestshell.ova`
`    Path                : /isan/bin/guestshell.ova`
`    Application`
`      Name              : GuestShell`
`      Installed version : 2.2(0.2)`
`      Description       : Cisco Systems Guest Shell`
`    Signing`
`      Key type          : Cisco key`
`      Method            : SHA-1`
`    Licensing`
`      Name              : None`
`      Version           : None`
`  Resource reservation`
`    Disk                : 400 MB`
`    Memory              : 256 MB`
`    CPU                 : 1% system CPU`
 	
`  Attached devices`
`    Type              Name        Alias`
`    ---------------------------------------------`
`    Disk              _rootfs`
`    Disk              /cisco/core`
`    Serial/shell`
`    Serial/aux`
`    Serial/Syslog                 serial2`
`    Serial/Trace                  serial3`
	guestshell パッケージに関する詳細（バージョン、署名リソース、デバイスなど）を表示します。

ゲスト シェルからのアプリケーションの永続的な起動

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

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

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

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

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

Guest Shell からアプリケーションを永続的に起動する手順

ステップ 1 上記で作成したアプリケーション サービス ファイルを /usr/lib/systemd/system/*application_name* にインストールします。サービス

ステップ 2 **systemctl start** *application_name* でアプリケーションを開始します

ステップ 3 アプリケーションが **systemctl status** -l *application_name* で実行されていることを確認します

ステップ 4 **systemctl enable** *application_name* でリロード時にアプリケーションを再起動できるようにします

ステップ 5 アプリケーションが **systemctl status** -l *application_name* で実行されていることを確認します

ゲスト シェルでのサンプル アプリケーション

次の例は、ゲスト シェルのアプリケーションを示しています。

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

OUTPUTFILE=/tmp/hello

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

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

[Install]
WantedBy=multi-user.target
[root@guestshell system]#
[root@guestshell system]# systemctl start hello
```

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

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

リロード後

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

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

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

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。