Guest Shell

About the Guest Shell

In addition to the NX-OS CLI and Bash access on the underlying Linux environment, the Cisco Nexus 9000 Series devices support access to a decoupled execution space running within a Linux Container (LXC) called the “guest shell”.

From within the guest shell the network-admin has the following capabilities:

- Access to the network.
- Access to Cisco Nexus 9000 bootflash.
- Access to Cisco Nexus 9000 CLI.
- Access to Cisco onePK APIs.
- The ability to install and run python scripts.
- The ability to install and run 64-bit Linux applications.

Decoupling the execution space from the native host system allows customization of the Linux environment to suit the needs of the applications without impacting the host system or applications running in other Linux Containers.

On NX-OS devices, Linux Containers are installed and managed with the virtual-service commands. The guest shell will appear in the virtual-service show command output.

Note By default the guest shell occupies approximately 5 MB of RAM and 200 MB of bootflash when enabled. Use the **guestshell destroy** command to reclaim resources if the guest shell is not used.

Accessing the Guest Shell

In Cisco NX-OS, the guest shell is accessible to the network-admin. It is automatically enabled in the system and can be accessed using the **run guestshell** command. Consistent with the **run bash** command, these commands can be issued within the guest shell with the **run guestshell** *command* form of the NX-OS CLI command.

```
switch# run guestshell ls -al /bootflash/*.ova
-rw-rw-rw- 1 2002 503 117616640 Aug 21 18:04 /bootflash/chef.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
```

Note When running in the guest shell, you have network-admin level privileges.

Capabilities in the Guest Shell

The guest shell has a number of utilities and capabilities available by default.

NX-OS CLI in the Guest Shell

The guest shell provides an application to allow the user to issue NX-OS commands from the guest shell environment to the host network element. The **dohost** application accepts any valid NX-OS configuration or exec commands and issues them to the host network element.

When invoking the **dohost** command each NX-OS command must be in double quotes:

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

The NX-OS CLI can be chained together:

```
guestshell:~$ dohost "conf t ; cdp timer 20"
{0}{}
{0}{}
guestshell:~$ dohost "show run | inc cdp"
{0}{cdp timer 20}
```

The value between the first set of brackets is the result code from the NXOS parser. The value between the second set of brackets is the output result from the command issued.

The NX-OS CLI can also be chained together using the NX-OS style command chaining technique by adding a semicolon between each command. (A space on either side of the semicolon is required.):

```
guestshell:~$ dohost "conf t ; cdp timer 13 ; show run | inc cdp"
{0}{cdp timer 13}
```

In this example, the **dohost** command received only one quoted command string. It returns one result string back from the NX-OS parser.

Note Commands issued on the host through the **dohost** command are run with network-admin level privileges.

Network Access in Guest Shell

The guest shell has a number of typical network utilities included by default and they can be used on different VRFs using the **chvrf** *vrf command* command.

Note Commands that are run without the **chvrf** command are run within the context of the default VRF.

For example, to ping IP address 10.0.0.1 over the management VRF, the command is "**chvrf** *management ping 10.0.0.1*". Other utilities such as **scp** or **ssh** would be similar.

Example:

```
switch# guestshell
guestshell:~$ cd /bootflash
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:/bootflash$ ls -al index.html
-rw-r--r-- 1 guestshe users 1804 Sep 13 20:28 index.html
guestshell:/bootflash$
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:/bootflash$
```

To obtain a list of VRFs on the system, use the **show vrf** command. The command can be run natively from the NX-OS CLI or by with the dohost *show vrf* command in the guest shell.

Example:

```
guestshell:/bootflash$ dohost "show vrf"
{0}{VRF-Name VRF-ID State Reason
default 1 Up --
management 2 Up --}
```

To resolve domain names from within the guest shell, the resolver needs to be configured. Edit the /etc/resolv.conf file in the guest shell to include a DNS nameserver and domain as appropriate for the network.

Example:

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

The nameserver and domain information should match what is configured through the NX-OS configuration.

Example:

```
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
```

If the Cisco Nexus 9000 device is in a network that uses an HTTP proxy server, the **http_proxy** and **https_proxy** environment variables must be set up within the guest shell also.

Example:

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

These environment variables should be set in the .bashrc file or in an appropriate script to ensure that they are persistent.

Access to Bootflash in Guest Shell

Network administrators can manage files with Linux commands and utilities in addition to using NX-OS CLI commands. By mounting the system bootflash at /bootflash in the guest shell environment, the network-admin can operate on these files with Linux commands.

Example:

```
find . –name “foo.txt”
rm “/bootflash/junk/foo.txt”
```

Python in Guest Shell

Python can be used interactively or python scripts can be run in the guest shell.

Example:

```
guestshell:~$ python
Python 2.7.3 (default, Aug 22 2014, 12:09:58)
[GCC 4.8.1] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> quit()
guestshell:~$
```

The pip python package manager is included in the guest shell to allow the network-admin to install new python packages.

Example:

```
guestshell:~$ pip list
argparse (1.2.1)
async (0.6.1)
iniparse (0.3.2)
ipaddress (branches-3144)
pexpect (2.3)
pip (1.5.6)
pycurl (7.19.0)
setuptools (0.6c11)
smart (1.4.1)
urlgrabber (3.9.1)
yum-metadata-parser (1.1.4)
```

Installing RPMs in the Guest Shell

By default, the Yum RPM package manager is included in the guest shell for the installation of software packages. Yum is pointed to the yocto repository.

```
guestshell:~$ cat /etc/yum/repos.d/yumrepo_x86_64.repo
[poky_1_5_1_x86_64]
baseurl=http://downloads.yoctoproject.org/releases/yocto/yocto-1.5.1/rpm/x86_64/
name=Poky 1.5.1 repository (x86_64)
enabled=1
```

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.

Yum resolves the dependancies and installs all the required packages.

```
guestshell:~$ sudo chvrf management yum install perl
Setting up Install Process
Resolving Dependencies
--> Running transaction check
---> Package perl.x86_64 0:5.14.3-r1 set to be updated
--> Processing Dependency: libperl5 >= 5.14.3 for package: perl-5.14.3-r1.x86_64
--> Processing Dependency: libperl.so.5()(64bit) for package: perl-5.14.3-r1.x86_64
--> Running transaction check
---> Package libperl5.x86_64 0:5.14.3-r1 set to be updated
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package         Arch          Version           Repository                Size
================================================================================
Installing:
 perl            x86_64        5.14.3-r1         poky_1_5_1_x86_64         16 k
Installing for dependencies:
 libperl5        x86_64        5.14.3-r1         poky_1_5_1_x86_64        712 k

Transaction Summary
================================================================================
Install       2 Package(s)
Upgrade       0 Package(s)

Total size: 728 k
Installed size: 1.6 M
```

```
Is this ok [y/N]: y
Downloading Packages:
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing     : libperl5-5.14.3-r1.x86_64                                  1/2
  Installing     : perl-5.14.3-r1.x86_64                                      2/2

Installed:
  perl.x86_64 0:5.14.3-r1

Dependency Installed:
  libperl5.x86_64 0:5.14.3-r1

Complete!
guestshell:~$
```

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.

Resources Used for the Guest Shell

By default, the resources for the guest shell have a small impact on resources available for normal switch operations. If the network-admin requires additional resources for the guest shell, the **guestshell resize** {*cpu* | *memory* | *rootfs*} command changes these limits.

Resource	Default	Minimum/Maximum
CPU	1%	1/20%
Memory	256MB	256/3840MB
Storage	204MB	204/1024MB

The CPU limit is the percentage of the system compute capacity that tasks running within the guest shell are given when there is contention with other compute loads in the system. When there is no contention for CPU resources, the tasks within the guest shell are not limited.

Note A guest shell reboot is required after changing the resource allocations. This can be accomplished with the **guestshell reboot** command.

Security Posture for Virtual Services

Use of the guest shell and virtual services in Cisco Nexus 9000 series devices are only two of the many ways that the network-admin can manage or extend the functionality of the system. These options are geared towards providing an execution environment that is decoupled from the native host context. This separation allows the introduction of software into the system that may not be compatible with the native execution environment. It also allows the software to run in an environment that does not interfere with the behavior, performance, or scale of the system.

Digitally Signed Application Packages

By default, Cisco network elements require applications to provide a valid Cisco digital signature at runtime. The Cisco digital signature ensures the integrity of Cisco-developed packages and applications.

The Cisco Nexus 9000 Series switches support the configuration of a signing level policy to allow for unsigned OVA software packages. To allow unsigned and Cisco-signed packages for creating virtual-services, the network-admin can configure the following:

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

Note The guest shell software package has a Cisco signature and does not require this configuration.

Kernel Vulnerability Patches

Cisco responds to pertinent Common Vulnerabilities and Exposures (CVEs) with platform updates that address known vulnerabilities.

ASLR and X-Space Support

Cisco Nexus 9000 NX-OS supports the use of Address Space Layout Randomization (ASLR) and Executable Space Protection (X-Space) for runtime defense. The software in Cisco-signed packages make use of this capability. If other software is installed on the system, it is recommended that it be built using a host OS and development toolchain that supports these technologies. Doing so reduces the potential attack surface that the software presents to potential intruders.

Root-User Restrictions

As a best practice for developing secure code, it is recommend running applications with the least privilege needed to accomplish the assigned task. To help prevent unintended accesses, software added into the guest shell should follow this best practice.

All processes within a virtual service are subject to restrictions imposed by reduced Linux capabilities. If your application must perform operations that require root privileges, restrict the use of the root account to the smallest set of operations that absolutely requires root access, and impose other controls such as a hard limit on the amount of time that the application can run in that mode.

The set of Linux capabilities that are dropped for root within virtual services follow:

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

As root within a virtual-service, bind mounts may be used as well as tmpfs and ramfs mounts. Other mounts are prevented.

Namespace Isolation

The host and virtual service are separated into separate namespaces. This provides the basis of separating the execution spaces of the virtual services from the host. Namespace isolation helps to protect against data loss and data corruption due to accidental or intentional data overwrites between trust boundaries. It also helps to ensure the integrity of confidential data by preventing data leakage between trust boundaries: an application in one virtual service cannot access data in another virtual service

Guest File System Access Restrictions

To preserve the integrity of the files within the virtual services, the file systems of the virtual services are not accessible from the NX-OS CLI. If a given virtual-service allows files to be modified, it needs to provide an alternate means by which this can be done (i.e. yum install, scp, ftp, etc).

The guest shell mounts the `bootflash` of the host system at `/bootflash`. The network-admin can access the file using an NX-OS CLI or Linux command from within the guest shell.

Resource Management

A Denial-of-Service (DoS) attack attempts to make a machine or network resource unavailable to its intended users. Misbehaving or malicious application code can cause DoS as the result of over-consumption of connection bandwidth, disk space, memory, and other resources. The host provides resource-management features that ensure fair allocation of resources among all virtual services on the host.

Secure IPC

Applications in a guest shell or virtual service can be made more integrated with the host by using Cisco onePK services. The applications communicate with the host network element over TIPC. Applications within various containers are not allowed to communicate with each other over TIPC, they are only allowed to talk to the host. This prevents issues of one container from spoofing that it is where the Cisco onePK services are running. Applications in containers are also not allowed to listen on TIPC ports.

To ensure that only know virtual services can communicate with the host, a unique identifier for each virtual service is created when it is enabled and verified at the time when the onePK communication channel is established.

The system also limits the rate at which an application in an individual virtual service can send messages to the host. This behavior prevents a misbehaving application from sending messages frequently enough to prevent normal operation of the host or to block other virtual services on the same host from communicating with the host.

Guidelines and Limitations

The guest shell has the following guidelines and limitations:

- By default, the guest shell starts an Open-SSH v6.2p2 server upon boot up. The server listens on port 4022 on the localhost ip address interface 127.0.0.1 only. This provides the password-less connectivity into the guest shell from the NX-OS vegas-shell when the **guestshell** keyword is entered. If this server is killed or its configuration (residing in `/etc/ssh/sshd_config`) is altered, access to the guest shell from the NX-OS CLI may not work. When this happens, you should navigate back into the guest shell with the **virtual-service connect name guestshell+ console** command. The username/password for this access is `guestshell/guestshell`.

  To instantiate your own Open-SSH server within the guest shell use the following steps as root:

  - Determine which VRF you want to establish your ssh connections through.
  - Determine which port you want your Open-SSH server to listen for connections. Use the NX-OS CLI **show socket connection** command to view which ports that are already in use.

    Note Do not select port 4022.

  - Start your Open-SSH server with the following command:

    chvrf *vrf_name /usr/sbin/sshd -p port_number*

- The time zone within the guest shell is not updated when the **clock timezone** command is configured.

  Use the Linux *TZ* environment variable to change the timezone within the guest shell.

  The following is an example that configures the time zone in the guest shell:

```
switch(config)# clock timezone PDT -7 0
switch(config)# clock set 11:01:15 12 Sep 2014
Fri Sep 12 11:01:15 PDT 2014
switch(config)# show clock
11:01:26.421 PDT Fri Sep 12 2014
switch(config)# run guestshell
guestshell:~$ export TZ='PDT7'
guestshell:~$ date
Fri Sep 12 11:01:59 PDT 2014
```

- Cisco Nexus 9000 NX-OS automatically installs and enables the guest shell by default. However, if the device is reloaded with a Cisco NX-OS image that does not provide guest shell support, the existing guest shell is automatically removed and a **%VMAN-2-INVALID_PACKAGE** message is issued.

  As a best practice, remove the guest shell with the **guestshell destroy** command before reloading the older Cisco NX-OS image.

Use the **install all** command to validate the compatibility between the current Cisco NX-OS image and the target Cisco NX-OS image.

The following is an example of incompatible images:

```
switch# install all nxos n9kpregs.bin
Installer will perform compatibility check first. Please wait.
uri is: /n9kpregs.bin
2014 Aug 29 20:08:51 switch %$ VDC-1 %$ %VMAN-2-ACTIVATION_STATE:
Successfully activated virtual service 'guestshell+'
Verifying image bootflash:/n9kpregs.bin for boot variable "nxos".
[####################] 100% -- SUCCESS
Verifying image type.
[####################] 100% -- SUCCESS
Preparing "lcn9k" version info using image bootflash:/n9kpregs.bin.
[####################] 100% -- SUCCESS
Preparing "bios" version info using image bootflash:/n9kpregs.bin.
[####################] 100% -- SUCCESS
Preparing "lcn9k" version info using image bootflash:/n9kpregs.bin.
[####################] 100% -- SUCCESS
Preparing "lcn9k" version info using image bootflash:/n9kpregs.bin.
[####################] 100% -- SUCCESS
Preparing "nxos" version info using image bootflash:/n9kpregs.bin.
[####################] 100% -- SUCCESS
Preparing "lcn9k" version info using image bootflash:/n9kpregs.bin.
[####################] 100% -- SUCCESS
Preparing "lcn9k" version info using image bootflash:/n9kpregs.bin.
[####################] 100% -- SUCCESS
"Running-config contains configuration that is incompatible with the new image (strict
 incompatibility).
Please run 'show incompatibility-all nxos <image>' command to find out which feature
needs to be disabled.".
Performing module support checks.
[####################] 100% -- SUCCESS
Notifying services about system upgrade.
[#                    ]   0% -- FAIL.
Return code 0x42DD0006 ((null)).
"Running-config contains configuration that is incompatible with the new image (strict
 incompatibility).
Please run 'show incompatibility-all nxos <image>' command to find out
which feature needs to be disabled."
Service "vman" in vdc 1: Guest shell not supported, do 'guestshell destroy' to remove
it and then retry ISSU
Pre-upgrade check failed. Return code 0x42DD0006 ((null)).
switch#
```

Managing the Guest Shell

The following are commands to manage the guest shell:

Table 1: Guest Shell CLI Commands

Commands	Description
guestshell enable [**package** *guest shell OVA file*]	Installs and activates the guest shell using the OVA that is embedded in the system image. Installs and activates the guest shell using the specified software package (OVA file) or the embedded package from the system image (when no package is specified). Initially, guest shell packages are only available by being embedded in the system image. When the guest shell is already installed, this command enables the installed guest shell. Typically this is used after a **guestshell disable** command.
guestshell disable	Shuts down and disables the guest shell.
guestshell upgrade [**package** *guest shell OVA file*]	Deactivates and upgrades the guest shell using the specified software package (OVA file) or the embedded package from the system image (if no package is specified). Initially guest shell packages are only available by being embedded in the system image. The current rootfs for the guest shell is replaced with the rootfs in the software package. The guest shell does not make use of secondary filesystems that persist across an upgrade. Without persistent secondary filesystems, a **guestshell destroy** command followed by a **guestshell enable** command could also be used to replace the rootfs. When an upgrade is successful, the guest shell is activated. You are prompted for a confirmation prior to carrying out the upgrade command.

Commands	Description
guestshell reboot	Deactivates the guest shell and then reactivates it. You are prompted for a confirmation prior to carrying out the reboot command. **Note** This is the equivalent of a **guestshell disable** command followed by a **guestshell enable** command in exec mode. This is useful when processes inside the guest shell have been stopped and need to be restarted. The **run guestshell** command relies on `sshd` running in the guest shell. If the command does not work, the `sshd` process may have been inadvertently stopped. Performing a reboot of the guest shell from the NX-OS CLI allows it to restart and restore the command.
guestshell destroy	Deactivates and uninstalls the guest shell. All resources associated with the guest shell are returned to the system. The **show virtual-service global** command indicates when these resources become available. Issuing this command results in a prompt for a confirmation prior to carrying out the destroy command.
guestshell **run guestshell**	Connects to the guest shell that is already running with a shell prompt. No username/password is required.
guestshell run *command* **run guestshell** *command*	Executes a Linux/UNIX command within the context of the guest shell environment. After execution of the command you are returned to the switch prompt.
guestshell resize [**cpu** | **memory** | **rootfs**]	Changes the allotted resources available for the guest shell. The changes take effect the next time the guest shell is enabled or rebooted.

Commands	Description
guestshell sync	On systems that have active and standby supervisors, this command synchronizes the guest shell contents from the active supervisor to the standby supervisor. The network-admin issues this command when the guest shell rootfs has been set up to a point that they would want the same rootfs used on the standby supervisor when it becomes the active supervisor. If this command is not used, the guest shell is freshly installed when the standby supervisor transitions to an active role using the guest shell package available on that supervisor.
virtual-service reset force	In the event that the guestshell or virtual-services cannot be managed, even after a system reload, the reset command is used to force the removal of the guest shell and all virtual-services. The system needs to be reloaded for the cleanup to happen. No guest shell or additional virtual-services can be installed or enabled after issuing this command until after the system has been reloaded. You are prompted for a confirmation prior to initiating the reset.

Note Administrative privileges are necessary to enable/disable and to gain access to the guest shell environment.

Note The guest shell is implemented as a Linux container (LXC) on the host system. On NX-OS devices, LXCs are installed and managed with the virtual-service commands. The guest shell appears in the virtual-service commands as a virtual service named `guestshell+`.

Disabling the Guest Shell

The **guestshell disable** command shuts down and disables the guest shell.

When the guest shell is disabled and the system is reloaded, the guest shell remains disabled.

Example:

```
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
```

Note The guest shell is reactivated with the **guestshell enable** command.

Destroying the Guest Shell

The **guestshell destroy** command uninstalls the guest shell and its artifacts. The command does not remove the guest shell OVA.

When the guest shell is destroyed and the system is reloaded, the guest shell remains destroyed.

```
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:
```

Note The guest shell can be re-enabled with the **guestshell enable** command.

Note In the Cisco NX-OS software, the **oneP** feature is automatically enabled for local access when a container is installed. Since the guest shell is a container, the **oneP** feature is automatically started.

If you do not want to use the guest shell, you can remove it with the **guestshell destroy** command. Once the guest shell has been removed, it remains removed for subsequent reloads. This means that when the guest shell container has been removed and the switch is reloaded, the guest shell container and the **oneP** feature are not automatically started.

Enabling the Guest Shell

The **guestshell enable** command installs the guest shell from a guest shell software package. By default, the package embedded in the system image is used for the installation. The command is also used to reactivate the guest shell if it has been disabled.

When the guest shell is enabled and the system is reloaded, the guest shell remains enabled.

Example:

```
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
```

Verifying Virtual Service and Guest Shell Information

You can verify virtual service and guest shell information with the following commands:

Command	Description

show virtual-service global
 	
`switch# show virtual-service global`
 	
`Virtual Service Global State and Virtualization Limits:`
 	
`Infrastructure version : 1.8`
`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 (%) 6 1 5`
`memory (MB) 2304 256 2048`
`bootflash (MB) 8192 248`
`3710`
	Displays the global state and limits for virtual services.

show virtual-service list
 	
`switch# show virtual-service list *`
 	
`Virtual Service List:`
 	
`Name Status Package Name`
`---`
`guestshell+ Activated guestshell.ova`
`chef Installed chef-0.8.1-n9000-spa-k9.ova`
	Displays a summary of the virtual services, the status of the virtual services, and installed software packages.

Command	Description

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 : 1.0(0.1)`
`      Description       : Cisco Systems Guest Shell`
`    Signing`
`      Key type          : Cisco key`
`      Method            : SHA-1`
`    Licensing`
`      Name              : None`
`      Version           : None`
`  Resource reservation`
`    Disk                : 204 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`
	Displays details about the guestshell package (such as version, signing resources, and devices).