



Guest Shell

Guestshell is a virtualized Linux-based environment, designed to run custom Linux applications, including Python for automated control and management of Cisco devices. It also includes the automated provisioning (Day zero) of systems. This container shell provides a secure environment, decoupled from the host device, in which users can install scripts or software packages and run them.

This module describes Guest Shell and how to enable it.

- [Information About the Guest Shell, on page 1](#)
- [How to Enable the Guest Shell, on page 9](#)
- [Configuration Examples for the Guest Shell, on page 15](#)
- [Additional References for Guest Shell, on page 19](#)
- [Feature Information for Guest Shell, on page 20](#)

Information About the Guest Shell

Guest Shell Overview

The Guest Shell is a virtualized Linux-based environment, designed to run custom Linux applications, including Python for automated control and management of Cisco devices. Using the Guest Shell, you can also install, update, and operate third-party Linux applications. The guest shell is bundled with the system image and can be installed using the **guestshell enable** Cisco IOS command.

The Guest Shell environment is intended for tools, Linux utilities, and manageability rather than networking.

Guest Shell shares the kernel with the host (Cisco switches and routers) system. Users can access the Linux shell of Guest Shell and update scripts and software packages in the container rootfs. However, users within the Guest Shell cannot modify the host file system and processes.

Guest Shell container is managed using IOx. IOx is Cisco's Application Hosting Infrastructure for Cisco IOS XE devices. IOx enables hosting of applications and services developed by Cisco, partners, and third-party developers in network edge devices, seamlessly across diverse and disparate hardware platforms.

This table provides information about the various Guest Shell capabilities and the supported platforms.

Table 1: Cisco Guest Shell Capabilities

	Guest Shell Lite (Limited LXC Container)	Guest Shell (LXC Container)
Operating System	Cisco IOS XE	Cisco IOS XE
Supported Platforms	<ul style="list-style-type: none"> • Cisco Catalyst 3650 Series Switches (all models) • Cisco Catalyst 3850 Series Switches (all models) 	<ul style="list-style-type: none"> • Cisco ISR 4000 Series Integrated Services Routers (Models with a minimum of 8 GB RAM.)
Guest Shell Environment	Montavista CGE7	CentOS 7
Python 2.7	Supported (Python V2.7.11)	Supported (Python V2.7.5)
Custom Python Libraries	<ul style="list-style-type: none"> • Cisco Embedded Event Manager • Cisco IOS XE CLIs • Ncclient 	<ul style="list-style-type: none"> • Cisco Embedded Event Manager • Cisco IOS XE CLIs
Supported Rootfs	Busybox, SSH, and Python PIP install	SSH, Yum install, and Python PIP install
GNU C Compiler	Not supported	Not supported
RPM Install	Not supported	Supported
Architecture	MIPS	x86

Guest Shell Vs Guest Shell Lite

The Guest Shell container allows users to run their scripts and apps on the system. The Guest Shell container on Intel x86 platforms will be a Linux container (LXC) with a CentOS 7.0 minimal rootfs. You can install other Python libraries such as, Python Version 3.0 during runtime using the Yum utility in CentOS 7.0. You can also install or update python packages using PIP.

The Guest Shell Lite container on MIPS platforms such as, Catalyst 3650 and Catalyst 3850 Series Switches have the Montavista Carrier Grade Edition (CGE) 7.0 rootfs. You can only install or run scripts in Guest Shell Lite. Yum install is not supported on these devices.

Guest Shell Security

Cisco provides security to ensure that users or apps in the Guest Shell do not compromise the host system. Guest Shell is isolated from the host kernel, and it runs as an unprivileged container.

Hardware Requirements for the Guest Shell

This section provides information about the hardware requirements for supported platforms. The Cisco CSR 1000v and Cisco ISRV (virtual platforms) implement these requirements in the software.

Table 2: Guest Shell Support on Catalyst Switches

Platforms	Default DRAM	Guest Shell Support
WS-3650-xxx (all)	4 GB	Supported
WS-3850-xxx (all)	4 GB	Supported
C9300-xx-x (all)	8 GB	Supported
C9500-24Q-x (all)	16 GB	Supported

The minimum system requirement for Catalyst 3850 Series Switches is 4 GB DRAM.

Table 3: Guest Shell Support on ISR 4000 Series Integrated Services Routers

Platform	Default DRAM	Guest Shell Support
ISR 4221	4GB	Not Supported
ISR 4321	4 GB	Not Supported
	8 GB	Supported
ISR 4331	8 GB	Supported
	16 GB	Supported
ISR 4351	8 GB	Supported
	16 GB	Supported
ISR 4431	8 GB	Supported
	16 GB	Supported
ISR 4451	8 GB	Supported
	16 GB	Supported

The minimum system requirement for ISR 4000 Series Integrated Services Routers is 8 GB DRAM.



Note Virtual-service installed applications and Guest Shell container cannot co-exist.

The minimum system requirement for CSR 1000v and ISRv is 4GB RAM.

Guest Shell Storage Requirements

On Catalyst 3650 and Catalyst 3850 Series Switches, Guest Shell can only be installed on the flash filesystem. Bootflash of Catalyst 3850 Series Switches require 75 MB free disk space for Guest Shell to install successfully.

On Cisco 4000 Series Integrated Services Routers, the Guest Shell is installed on the Network Interface Module (NIM)-Service Set Identifier (SSD) (hard disk), if available. If the hard disk drive is available, there

is no option to select bootflash to install Guest Shell. Cisco 4000 Series Integrated Services Routers require 1100 MB free hard disk (NIM-SSID) space for Guest Shell to install successfully.

For Cisco 4000 Series Integrated Services Routers and ASR 1000 routers (when an optional hard disk has been added to that router) you can only do resource resizing if you have installed the Guest Shell on the hard disk and inserted the hard disk into the router.



Note A Guest Shell installed via bootflash does not allow you to do resource resizing using application hosting configuration commands.)

During Guest Shell installation, if enough hard disk space is not available, an error message is displayed.

The following is a sample error message on an ISR 4000 Series router:

```
% Error:guestshell_setup.sh returned error:255, message:
Not enough storage for installing guestshell. Need 1100 MB free space.
```

Bootflash or hard disk space can be used to store additional data by Guest Shell. On Cisco Catalyst 3850 Series Switches, Guest Shell has 18 MB of storage space available and on Cisco 4000 Series Integrated Services Routers, Guest Shell has 800 MB of storage space available. Because Guest Shell accesses the bootflash, it can use the entire space available.

Table 4: Resources Available to Guest Shell and Guest Shell Lite

Resource	Default	Minimum/Maximum
CPU	1% Note 1% is not standard; 800 CPU units/ total system CPU units.	1/100%
Memory	256 MB 512 MB (Cisco CSR 1000v)	256/256 MB 512/512 MB (Cisco CSR 1000v)

Accessing Guest Shell on a Device

Network administrators can use IOS commands to manage files and utilities in the Guest Shell.

During the Guest Shell installation, SSH access is setup with a key-based authentication. The access to the Guest Shell is restricted to the user with the highest privilege (15) in IOS. This user is granted access into the Linux container as the *guestshell* Linux user, who is a sudoer, and can perform all root operations. Commands executed through the Guest Shell are executed with the same privilege that a user has when logged into the IOS terminal.

At the Guest Shell prompt, you can execute standard Linux commands.

Accessing Guest Shell Through the Management Port

By default, Guest Shell allows applications to access the management network. Users cannot change the management VRF networking configurations from inside the Guest Shell.



Note For platforms without a management port, a VirtualPortGroup can be associated with Guest Shell in the IOS configuration. For more information, see the *Sample VirtualPortGroup Configuration* section.

Stacking with Guest Shell

When Guest Shell is installed, a directory is automatically created in the flash filesystem. This directory is synchronized across stack members. During a switchover, only contents of the this directory are synchronized across all stack members. To preserve data during high availability switchover, place data in this directory.

During a high availability switchover, the new active device creates its own Guest Shell installation and restores Guest Shell to the synchronized state; the old filesystem is not maintained. Guestshell state is internally synchronized across all stack members.

IOx Overview

IOx is a Cisco-developed end-to-end application framework that provides application hosting capabilities for different application types on Cisco network platforms. The Cisco Guest Shell, a special container deployment, is one such application, that is useful in system deployment/use.

IOx facilitates the life-cycle management of app and data exchange by providing a set of services that helps developers to package pre-built apps, and host them on a target device. IOx life-cycle management includes distribution, deployment, hosting, starting, stopping (management), and monitoring of apps and data. IOx services also include app distribution and management tools that help users discover and deploy apps to the IOx framework.

App hosting provides the following features:

- Hides network heterogeneity.
- IOx application programming interfaces (APIs), remotely manage the life cycle of applications hosted on a device.
- Centralized app life-cycle management.
- Cloud-based developer experience.

IOx Tracing and Logging Overview

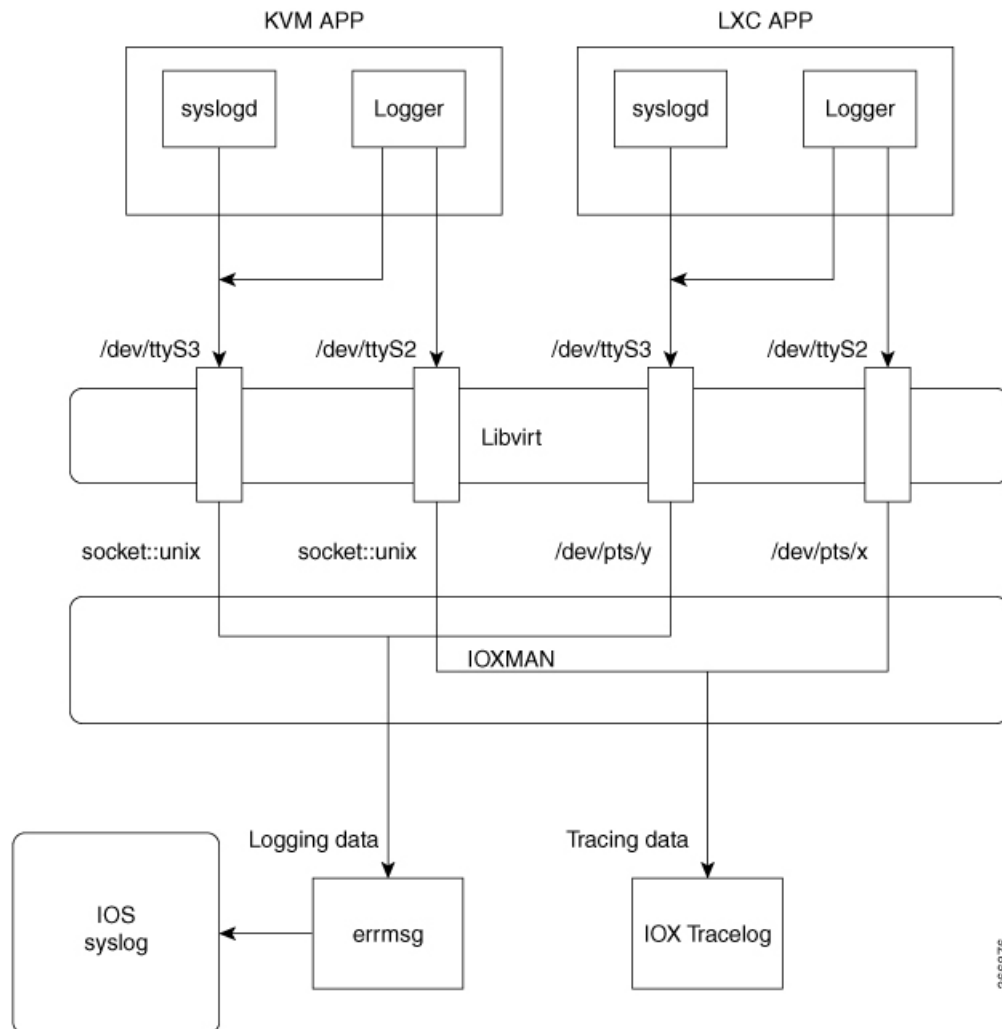
IOx tracing and logging feature allows guest application to run separately on the host device that can help reporting the logging and tracing of the data to the host. The tracing data is saved into IOx tracelog, and the logging data is saved into IOS syslog on the host device.

You can redirect the tracing data to the appropriate storage device on the host device which can help in debugging of guest application.

IOXMAN Structure

Each guest application, a system LXC or a KVM instance is configured with its own syslogd and logfiles stored within a visible file system and are not accessible to the host device. To support logging data to IOS syslog and tracing data to IOx tracelog on the host, two serial devices, `/dev/ttyS2` and `/dev/ttyS3`, are designated on the guest application for delivering data to the host as shown in the following figure.

Figure 1: IOXMAN Structure



IOXMAN is a process to establish the tracing infrastructure to provide logging or tracing services for guest application, except Libvirt that emulates serial devices. IOXMAN is based on the lifecycle of the guest application to enable and disable tracing service, to send logging data to IOS syslog, to save tracing data to IOx tracelog, and to maintain IOx tracelog for each guest application.

Logging and Tracing System Flow

The following sections describes how the IOx logging and tracing works:

LXC Logging

1. Guest OS enables **/dev/ttyS2** on Guest application.
2. Guest application writes data to **/dev/ttyS2**.
3. Libvirt emulates **/dev/ttyS2** to **/dev/pts/x** on the host.
4. IOXMAN gets the emulated serial device, **/dev/pts/x** from the XML file.

5. IOXMAN listens and reads available data from **/dev/pts/x**, sets the severity for the message, filters, parses and queues the message.
6. Start timer to send the message to **/dev/log** device on the host using **errmsg**.
7. Data is saved to IOS syslog.

KVM Logging

1. Guest OS enables **/dev/ttyS2** on Guest application.
2. Guest application writes data to **/dev/ttyS2**.
3. Libvert emulates **/dev/ttyS2** to **/dev/pts/x** on the host.
4. IOXMAN gets the emulated TCP path from the XML file.
5. IOXMAN opens an unix socket, and connects to the remote socket.
6. IOXMAN reads available data from the socket, sets the severity for the message, filters, parses, and queues the message.
7. Start timer to send the message to **/dev/log** device on the host using **errmsg**.
8. Data is saved to IOS syslog.

LXC Tracing

1. Guest OS enables **/dev/ttyS3** on Guest application.
2. Configures **syslogd** to copy message to **/dev/ttyS3**.
3. Guest application writes data to **/dev/ttyS3**.
4. Libvert emulates **/dev/ttyS3** to **/dev/pts/y** on the host.
5. IOXMAN gets the emulated serial device, **/dev/pts/y** from the XML file.
6. IOXMAN listens and reads available data from **/dev/pts/y**, filters, parses, and saves the message to IOx tracelog.
7. If IOx tracelog is full, IOXMAN rotates the tracelog file to **/bootflash/tracelogs**.

KVM Tracing

1. Guest OS enables **/dev/ttyS3** on Guest application.
2. Configures **syslogd** to copy message to **/dev/ttyS3**.
3. Guest application writes data to **/dev/ttyS3**.
4. Libvert emulates **/dev/ttyS3** to TCP path on the host.
5. IOXMAN gets the emulated TCP path from the XML file.
6. IOXMAN opens an unix socket, and connects to the remote socket.

7. IOXMAN reads available data from the socket, sets the severity for the message, filters, parses, and saves the message to IOx tracelog.
8. If IOx tracelog is full, IOXMAN rotates the tracelog file to **/bootflash/tracelogs**.

Logging and Tracing of Messages

The following sections explain the logging and tracing of messages in to IOS syslog.

Logging Messages in IOS Syslog

For any logging messages received from the Guest Application, IOXMAN sets the severity of the message to NOTICE by default, before sending it to IOS syslog. When a message is received by IOSd, it is displayed on the console and saved on the IOS syslog in the following message format.

***Apr 7 00:48:21.911: %IM-5-IOX_INST_NOTICE:ioxman: IOX SERVICE guestshell LOG: Guestshell test**

In order to comply with IOS syslog, IOXMAN does support severity for logging message. To report logging message with severity, Guest Application needs to append the header to the front of the message.

```
[a123b234,version,severity]

a123b234 is magic number.
Version:      severity support version.  Current version is 1.
Severity:     CRIT is 2
              ERR is 3
              WARN is 4
              NOTICE is 5
              INFO is 6
              DEBUG is 7
```

Following is an example of a message log:

```
echo "[a123b234,1,2]Guestshell failed" > /dev/ttyS2
```

Perform the following steps to report logging data from Guest Application to IOS syslog:

1. If you are using C programming, use **write()** to send logging data to host.

```
#define SYSLOG_TEST      "syslog test"
int fd;
fd = open("/dev/ttyS2", O_WRONLY);
write(fd, SYSLOG_TEST, strlen(SYSLOG_TEST));
close(fd);
```

2. If you are using Shell console, use **echo** to send logging data to host.

```
echo "syslog test" > /dev/ttyS2
```

Tracing Message to IOx Tracelog

Perform the following steps to report tracing messages from Guest Application to IOx tracelog:

1. If you are using C programming, use **write()** to send tracing message to host.


```
#define SYSLOG_TEST      "tracelog test"
int fd;
fd = open("/dev/ttyS3", O_WRONLY);
write(fd, SYSLOG_TEST, strlen(SYSLOG_TEST));
close(fd);
```

2. If you are using C programming, use **syslog()** to send tracing message to host.

```
#define SYSLOG_TEST      "tracelog test"

syslog(LOG_INFO, "%s\n", SYSLOG_TEST);
```

3. If you are using Shell console, use **echo** to send tracing data to host.

```
echo "tracelog test" > /dev/ttyS3
or
logger "tracelog test"
```

Example: Guest Shell Networking Configuration

For Guest Shell networking, the following configurations are required.

- Configure Domain Name System (DNS)
- Configure proxy settings
- Configure YUM or PIP to use proxy settings

How to Enable the Guest Shell

Managing IOx

Before you begin

IOx takes upto two minutes to start. CAF, IOXman, and Libird services must be running to enable Guest Shell successfully.

Procedure

	Command or Action	Purpose
Step 1	enable Example: Device> enable	Enables privileged EXEC mode. <ul style="list-style-type: none"> • Enter your password if prompted.
Step 2	configure terminal Example: Device# configure terminal	Enters global configuration mode.

	Command or Action	Purpose
Step 3	iox Example: Device(config)# iox	Configures IOx services.
Step 4	exit Example: Device(config)# exit	Exits global configuration mode and returns to privileged EXEC mode.
Step 5	show iox-service Example: Device# show iox-service	Displays the status of the IOx service
Step 6	show app-hosting list Example: Device# show app-hosting list	Displays the list of app-hosting services enabled on the device.

What to do next

The following is sample output from the **show iox-service** command on an ISR 4000 Series Router:

```
Device# show iox-service
```

```
Virtual Service Global State and Virtualization Limits:
```

```
Infrastructure version : 1.7
Total virtual services installed : 0
Total virtual services activated : 0
```

```
Machine types supported : KVM, LXC
Machine types disabled : none
```

```
Maximum VCPUs per virtual service : 6
Resource virtualization limits:
```

Name	Quota	Committed	Available
system CPU (%)	75	0	75
memory (MB)	10240	0	10240
bootflash (MB)	1000	0	1000
harddisk (MB)	20000	0	18109
volume-group (MB)	190768	0	170288

```
IOx Infrastructure Summary:
```

```
-----
IOx service (CAF) : Running
IOx service (HA) : Not Running
IOx service (IOxman) : Running
Libvirtd : Running
```

The following is truncated sample output from the **show iox-service** command on a Catalyst 3850 Series Switch:

```
Device# show iox-service
```

```
IOx Infrastructure Summary:
-----
IOx service (CAF)      : Running
IOx service (HA)       : Running
IOx service (IOxman)   : Running
LibvirtD               : Running
```

The following is sample output from the **show app-hosting list** command:

```
Device# show app-hosting list

App id                               State
-----
guestshell                           RUNNING
```

Managing the Guest Shell



Note VirtualPortGroups are supported only on routing platforms.

Before you begin

IOx must be configured and running for Guest Shell access to work. If IOx is not configured, a message to configure IOx is displayed. Removing IOx removes access to the Guest Shell, but the rootfs remains unaffected.

An application or management interface must also be configured to enable and operate Guest Shell. See "Configuring the AppGigabitEthernet Interface for Guest Shell" and "Enabling Guest Shell on the Management Interface" sections for more information on enabling an interface for Guest Shell.

Procedure

	Command or Action	Purpose
Step 1	enable Example: Device> enable	Enables privileged EXEC mode. <ul style="list-style-type: none">• Enter your password if prompted.
Step 2	guestshell enable Example:	Enables the Guest Shell service.

	Command or Action	Purpose
	Device# <code>guestshell enable</code>	Note <ul style="list-style-type: none"> • The guestshell enable command uses the management virtual routing and forwarding (VRF) instance for networking. • When using VirtualPortGroups (VPGs) for front panel networking, the VPG must be configured first. • The guest IP address and the gateway IP address must be in the same subnet.
Step 3	guestshell run <i>linux-executable</i> Example: Device# <code>guestshell run python</code> or Device# <code>guestshell run python3</code>	Executes or runs a Linux program in the Guest Shell. Note In Cisco IOS XE Amsterdam 17.3.1 and later releases, only Python version 3 is supported.
Step 4	guestshell run bash Example: Device# <code>guestshell run bash</code>	Starts a Bash shell to access the Guest Shell.
Step 5	guestshell disable Example: Device# <code>guestshell disable</code>	Disables the Guest Shell service.
Step 6	guestshell destroy Example: Device# <code>guestshell destroy</code>	Deactivates and uninstalls the Guest Shell service.

Enabling and Running the Guest Shell

The **guestshell enable** command installs Guest Shell. This command is also used to reactivate Guest Shell, if it is disabled.

When Guest Shell is enabled and the system is reloaded, Guest Shell remains enabled.



Note

IOx must be configured before the **guestshell enable** command is used.

The **guestshell run bash** command opens the Guest Shell bash prompt. Guest Shell must already be enabled for this command to work.



Note If the following message is displayed on the console, it means that IOx is not enabled; check the output of the **show iox-service** command to view the status of IOx.

```
The process for the command is not responding or is otherwise unavailable
```

For more information on how to enable Guest Shell, see the "Configuring the AppGigabitEthernet Interface for Guest Shell" and "Enabling Guest Shell on the Management Interface" sections.

Disabling and Destroying the Guest Shell

The **guestshell disable** command shuts down and disables Guest Shell. When Guest Shell is disabled and the system is reloaded, Guest Shell remains disabled.

The **guestshell destroy** command removes the rootfs from the flash filesystem. All files, data, installed Linux applications and custom Python tools and utilities are deleted, and are not recoverable.

Managing the Guest Shell Using Application Hosting



Note This section is applicable to Cisco routing platforms. VirtualPortGroups are not supported on Cisco Catalyst Switching platforms.

IOx must be configured and running for Guest Shell access to work. If IOx is not configured, a message to configure IOx is displayed. Removing IOx removes access to the Guest Shell, but the rootfs remains unaffected.



Note Use this procedure (Managing the Guest Shell Using Application Hosting) to enable the Guest Shell in Cisco IOS XE Fuji 16.7.1 and later releases. For Cisco IOS XE Everest 16.6.x and previous releases, use the procedure in [Managing the Guest Shell, on page 11](#).

```
Device(config)# interface GigabitEthernet1
Device(config-if)# ip address dhcp
Device(config-if)# ip nat outside
Device(config-if)# exit
```

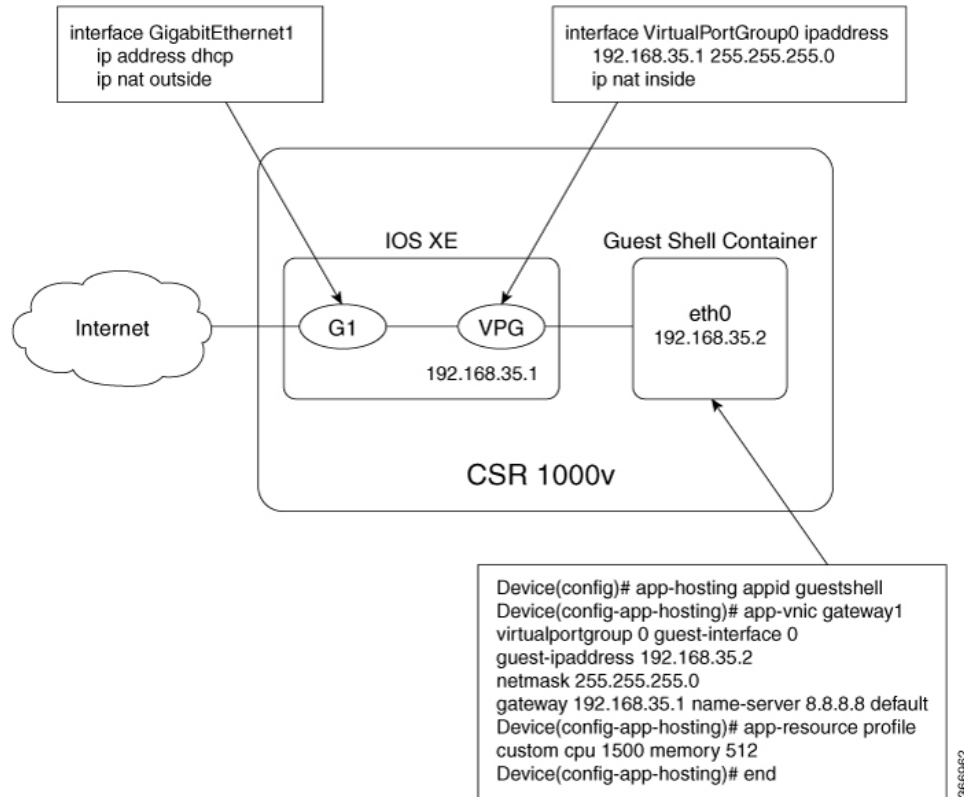
```
Device(config-if)# interface VirtualPortGroup0
Device(config-if)# ip address 192.168.35.1 255.255.255.0
Device(config-if)# ip nat inside
Device(config-if)# exit
```

```
Device(config)# ip nat inside source list GS_NAT_ACL interface GigabitEthernet1 overload
Device(config)# ip access-list standard GS_NAT_ACL
Device(config)# permit 192.168.0.0 0.0.255.255
```

```
Device(config)# app-hosting appid guestshell
Device(config-app-hosting)# app-vnic gateway1 virtualportgroup 0 guest-interface 0
guest-ipaddress 192.168.35.2
netmask 255.255.255.0 gateway 192.168.35.1 name-server 8.8.8.8 default
Device(config-app-hosting)# app-resource profile custom cpu 1500 memory 512
Device(config-app-hosting)# end
```

```
Device# guestshell enable
Device# guestshell run python
```

Figure 2: Managing the Guest Shell using Application Hosting



For front panel networking, you must configure the GigabitEthernet and VirtualPortGroup interfaces as shown above. The Guest Shell uses a Virtualportgroup as the source interface to connect to the outside network through NAT.

The following commands are used to configure inside NAT. They allow the Guest Shell to reach the internet; for example, to obtain Linux software updates:

```
ip nat inside source list
ip access-list standard
permit
```

The **guestshell run** command in the example above, runs a python executable. You can also use the **guestshell run** command to run other Linux executables; for example, see the example **guestshell run bash** command, which starts a Bash shell or the **guestshell disable** command which shuts down and disables the Guest Shell. If the system is later reloaded, the Guest Shell remains disabled.

Accessing the Python Interpreter

Python can be used interactively or Python scripts can be run in the Guest Shell. Use the **guestshell run python** command to launch the Python interpreter in Guest Shell and open the Python terminal.



Note In releases prior to Cisco IOS XE Amsterdam 17.3.1, Python V2 is the default. Python V3 is supported in Cisco IOS XE Amsterdam 17.1.1, and Cisco IOS XE Amsterdam 17.2.1. In Cisco IOS XE Amsterdam 17.3.1 and later releases, Python V3 is the default.

In Releases Prior to Cisco IOS XE Amsterdam 17.3.1

The **guestshell run** command is the Cisco IOS equivalent of running Linux executables, and when running a Python script from Cisco IOS, specify the absolute path. The following example shows how to specify the absolute path for the command:

```
Guestshell run python /flash/guest-share/sample_script.py parameter1 parameter2
```

The following example shows how to enable Python on a Cisco Catalyst 3650 Series Switch or a Cisco Catalyst 3850 Series Switch:

```
Device# guestshell run python

Python 2.7.11 (default, March 16 2017, 16:50:55)
[GCC 4.7.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>>>
```

The following example shows how to enable Python on a Cisco ISR 4000 Series Integrated Services Router:

```
Device# guestshell run python

Python 2.7.5 (default, Jun 17 2014, 18:11:42)
[GCC 4.8.2 20140120 (Red Hat 4.8.2-16)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>>>
```

In Cisco IOS XE Amsterdam 17.3.1 and Later Releases

The following example shows how to enable Python on Cisco Catalyst 9000 Series Switches:

```
Device# guestshell run python3

Python 3.6.8 (default, Nov 21 2019, 22:10:21)
[GCC 8.3.1 20190507 (Red Hat 8.3.1-4)] on linux
Type "help", "copyright", "credits" or "license" for more information.>>>>>
```

Configuration Examples for the Guest Shell

Example: Managing the Guest Shell

The following example shows how to enable Guest Shell on a Catalyst 3850 Series Switch:

```

Device> enable
Device# guestshell enable

Management Interface will be selected if configured
Please wait for completion
Guestshell enabled successfully

Device# guestshell run python

Python 2.7.11 (default, Feb 21 2017, 03:39:40)
[GCC 5.3.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.

Device# guestshell run bash

[guestshell@guestshell ~]$

Device# guestshell disable

Guestshell disabled successfully

Device# guestshell destroy

Guestshell destroyed successfully

```

Sample VirtualPortGroup Configuration



Note

VirtualPortGroups are supported only on Cisco routing platforms.

When using the VirtualPortGroup interface for Guest Shell networking, the VirtualPortGroup interface must have a static IP address configured. The front port interface must be connected to the Internet and Network Address Translation (NAT) must be configured between the VirtualPortGroup and the front panel port.

The following is a sample VirtualPortGroup configuration:

```

Device> enable
Device# configure terminal
Device(config)# interface VirtualPortGroup 0
Device(config-if)# ip address 192.168.35.1 255.255.255.0
Device(config-if)# ip nat inside
Device(config-if)# no mop enabled
Device(config-if)# no mop sysid
Device(config-if)# exit
Device(config)# interface GigabitEthernet 0/0/3
Device(config-if)# ip address 10.0.12.19 255.255.0.0
Device(config-if)# ip nat outside
Device(config-if)# negotiation auto
Device(config-if)# exit
Device(config)# ip route 0.0.0.0 0.0.0.0 10.0.0.1
Device(config)# ip route 10.0.0.0 255.0.0.0 10.0.0.1
!Port forwarding to use ports for SSH and so on.
Device(config)# ip nat inside source static tcp 192.168.35.2 7023 10.0.12.19 7023 extendable

```



```

Device(config)# ip nat outside source list NAT_ACL interface GigabitEthernet 0/0/3 overload
Device(config)# ip access-list standard NAT_ACL
Device(config-std-nacl)# permit 192.168.0.0 0.0.255.255
Device(config-std-nacl)# exit

! App-hosting configuration
Device(config)# app-hosting appid guestshell
Device(config-app-hosting)# app-vnic gateway1 virtualportgroup 0 guest-interface 0
guest-ipaddress 192.168.35.2
netmask 255.255.255.0 gateway 192.168.35.1 name-server 8.8.8.8 default
Device(config-app-hosting)# app-resource profile custom cpu 1500 memory 512
Device(config-app-hosting)# end

Device# guestshell enable
Device# guestshell run python

```

Example: Guest Shell Usage

From the Guest Shell prompt, you can run Linux commands. The following example shows the usage of some Linux commands.

```

[guestshell@guestshell~]$ pwd
/home/guestshell

[guestshell@guestshell~]$ whoami
guestshell

[guestshell@guestshell~]$ uname -a
Linux guestshell 3.10.101.cge-rt110 #1 SMP Sat Feb 11 00:33:02
PST 2017 mips64 GNU/Linux

```

Catalyst 3650 and Catalyst 3850 Series Switches have a defined set of Linux executables that are provided by BusyBox and Cisco 4000 Series Integrated Services Routers have commands provided by CentOS Linux release 7.1.1503.

The following example shows the usage of the **dohost** command on a Catalyst 3850 Series Switch.

```

[guestshell@guestshell ~]$ dohost "show version"

Cisco IOS Software [Everest], Catalyst L3 Switch Software [CAT3K_CAA-UNIVERSALK9-M],
Experimental Version 16.5.2017200014[v165_throttle-BLD-
BLD_V165_THROTTLE_LATEST_20170531_192849 132]

```



Note The **dohost** command requires the **ip http server** command to be configured on the device.

Example: Guest Shell Networking Configuration

For Guest Shell networking, the following configurations are required.

- Configure Domain Name System (DNS)
- Configure proxy settings
- Configure YUM or PIP to use proxy settings

Sample DNS Configuration for Guest Shell

The following is a sample DNS configuration for Guest Shell:

```
[guestshell@guestshell ~]$ cat/etc/resolv.conf
nameserver 192.0.2.1
```

Other Options:

```
[guestshell@guestshell ~]$ cat/etc/resolv.conf
domain cisco.com
search cisco.com
nameserver 192.0.2.1
search cisco.com
nameserver 198.51.100.1
nameserver 172.16.0.6
domain cisco.com
nameserver 192.0.2.1
nameserver 172.16.0.6
nameserver 192.168.255.254
```

Example: Configuring Proxy Environment Variables

If your network is behind a proxy, configure proxy variables in Linux. If required, add these variables to your environment.

The following example shows how to configure your proxy variables:

```
[guestshell@guestshell ~]$cat /bootflash/proxy_vars.sh
export http_proxy=http://proxy.example.com:80/
export https_proxy=http://proxy.example.com:80/
export ftp_proxy=http://proxy.example.com:80/
export no_proxy=example.com
export HTTP_PROXY=http://proxy.example.com:80/
export HTTPS_PROXY=http://proxy.example.com:80/
export FTP_PROXY=http://proxy.example.com:80/
guestshell ~] source /bootflash/proxy_vars.sh
```

Example: Configuring Yum and PIP for Proxy Settings

The following example shows how to use Yum for setting proxy environment variables:

```
cat /etc/yum.conf | grep proxy
[guestshell@guestshell~]$ cat /bootflash/yum.conf | grep proxy
proxy=http://proxy.example.com:80/
```

PIP install picks up environment variable used for proxy settings. Use sudo with -E option for PIP installation. If the environment variables are not set, define them explicitly in PIP commands as shown in following example:

```
sudo pip --proxy http://proxy.example.com:80/install requests
sudo pip install --trusted-host pypi.example.com --index-url
http://pypi.example.com/simple requests
```

The following example shows how to use PIP install for Python:

```
Sudo -E pip install requests
[guestshell@guestshell ~]$ python
Python 2.17.11 (default, Feb 3 2017, 19:43:44)
[GCC 4.7.0] on linux2
Type "help", "copyright", "credits" or "license" for more information
>>>import requests
```

Additional References for Guest Shell

Related Documents

Related Topic	Document Title
	Programmability Command Reference, Cisco IOS XE Everest 16.6.1
Python module	CLI Python Module
Zero-Touch Provisioning	Zero-Touch Provisioning

MIBs

MB	MIBs Link
	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

Technical Assistance

Description	Link
<p>The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies.</p> <p>To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds.</p> <p>Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.</p>	http://www.cisco.com/support

Feature Information for Guest Shell

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 5: Feature Information for Guest Shell

Feature Name	Release	Feature Information
Guest Shell	Cisco IOS XE Everest 16.5.1a Cisco IOS XE Everest 16.5.1b	<p>Guest Shell is a secure container that is an embedded Linux environment that allows customers to develop and run Linux and custom Python applications for automated control and management of Cisco switches. It also includes the automated provisioning of systems. This container shell provides a secure environment, decoupled from the host device, in which users can install scripts or software packages and run them.</p> <p>In Cisco IOS XE Everest 16.5.1a, this feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches <p>In Cisco IOS Everest 16.5.1b, this feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> • Cisco 4000 Series Integrated Services Routers
	Cisco IOS XE Everest 16.6.2	In Cisco IOS XE Everest 16.6.2, this feature was implemented on Cisco Catalyst 9400 Series Switches.
	Cisco IOS XE Fuji 16.7.1	<p>In Cisco IOS XE Fuji 16.7.1, this feature was implemented on Cisco CSR 1000v Series.</p> <p>In Cisco IOS XE Fuji 16.7.1, for Guest Shell feature, the Logging and Tracing support was implemented on Cisco ASR 1000 Aggregation Services Routers.</p>

