

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.

- Restrictions for Guest Shell, on page 1
- Information About the Guest Shell, on page 2
- How to Enable the Guest Shell, on page 12
- Configuration Examples for the Guest Shell, on page 21
- Additional References for Guest Shell, on page 27
- Feature Information for Guest Shell, on page 27

Restrictions for Guest Shell

- NETCONF sessions cannot be established on the standby Route Processor (RP).
- Python scripts fail when running commands like the **show tech-support wireless** command, when the scale is set to 2000Aps, and clients are set to 10000.

  The output of commands like **show tech-support wireless** is huge and can cause memory exhaustion inside the Guest Shell. When using commands with huge output, redirect the output to a file. The IOS CLI can write the output to a file in the */bootflash/guest-share* directory, and it can be accessed from the Guest Shell.
- Cisco Catalyst 9200CX Series Switches do not support the Management interface, AppGigabitEthernet interface, or VirtualPortGroup interface. Applications or scripts running in the Guest Shell will not be able to communicate with the external network.
- In Cisco IOS XE 17.16.1 and later releases, Catalyst 9200CX Series Switches will use VirtualPortGroup interface with Network Address Translation (NAT) to communicate with the external network. This communication is only supported for day zero ZTP running in Guest Shell.
- When you run a Python script inside Guest Shell, background services are unable to access the **dohost** command. As a workaround, run the Python script without the sudo command, or use sudo with the -E option.

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 root filesystem. 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.

Guest Shell Software Requirements

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 8.0 minimal rootfs. You can install other Python libraries such as, Python Version 3.0 during runtime using the Yum utility in CentOS 8.0. You can also install or update python packages using PIP.

Table 1: Guest Shell Software Requirements

	Guest Shell (LXC Container)
Operating System	Cisco IOS XE
Platform	All supported Cisco IOS XE platforms
Guest Shell Environment	• CentOS 7 is supported in Cisco IOS XE Amsterdam 17.2.1 and previous releases. • CentOS 8 is supported in Cisco IOS XE Amsterdam 17.3.1 and later releases. **Note** CentOS supports only Python 3.6. • CentOS 9 is supported in Cisco IOS XE 17.18.1 and later releases. CentOS 9 supports Python 3.9.
Python 2.7	Supported till Cisco IOS XE Amsterdam 17.3.1

	Guest Shell (LXC Container)
Python 3.6	Supported in Cisco IOS XE Amsterdam 17.1.1 and later releases. In Cisco IOS XE Amsterdam 17.1.1 and Cisco IOS XE Amsterdam 17.2.1, Python V2 is the default. However, in Cisco IOS XE Amsterdam 17.3.1 and later releases, Python V3 is the default. **Note** Cisco Catalyst 9200 Series Switches support Python version 3 in Cisco IOS XE Amsterdam 17.3.1 and later releases.
Python 3.9	Supported in Cisco IOS XE 17.18.1 and later releases.
Pre-installed Custom Python Libraries	• Cisco Embedded Event Manager • Cisco IOS XE CLIs • NCCLIENT library for the NETCONF API
Supported Rootfs	SSH, Yum install, and Python PIP install
GNU C Compiler	Not supported
RPM Install	Supported
Architecture	x86 and ARM

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 which have variable memory configurations.

Table 2: Guest Shell Resource Requirements

Platforms	Minimum Memory
Cisco 1000 Series Integrated Services Routers	4 GB
Cisco Catalyst 8000V Edge Software	4 GB
Cisco Cloud Services Router 1000V Series	4 GB
Cisco ISR 4000 Series Integrated Services Routers	8 GB DRAM (In Cisco IOS XE Fuji 16.8.1 and previous releases.) 4GB DRAM (In Cisco IOS XE Fuji 16.8.1 and later releases.)

All other platforms are shipped with sufficient resources to support Guest Shell.

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

Guest Shell Storage Requirements

Cisco Catalyst 9300 Series Switches and Cisco Catalyst 9500 Series Switches require 1100 MB free hard 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)-Solid State Drive (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-SSD) space for Guest Shell to install successfully.

For Cisco 4000 Series Integrated Services Routers and Cisco ASR 1000 Series Aggregation Services 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 Cisco ISR 4000 Series Integrated Services 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 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 3: 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 Catalyst 8000V Edge Software and Cisco Cloud Services Router 1000V Series)	256/256 MB 512/512 MB (Cisco Catalyst 8000V Edge Software and Cisco Cloud Services Router 1000V Series)

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.

Accessing Guest Shell on a Device

Network administrators can use Cisco 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 Cisco 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 Cisco 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 Cisco IOS configuration. For more information, see the *Sample VirtualPortGroup Configuration* section.

Cisco Catalyst 9200 Series Switches, Cisco Catalyst 9300 Series Switches, and Cisco Catalyst 9400 Series Switches support the AppGigabitEthernet interface and management interface (mgmt-if) to access Guest Shell.

Cisco Catalyst 9500 and 9500 High-Performance Series Switches and Cisco Catalyst 9600 Series Switches do not support AppGigabitEthernet interfaces.



Note Cisco Catalyst 9200L SKUs do not support Guest Shell.

Day Zero Guest Shell Provisioning Using Front-Panel Port or Fiber Uplink

On Day Zero, when the device has no management connectivity, and the only connectivity is either through the front-panel port or fibre uplink port, Guest Shell is internally configured to use the available port. The AppGigabitEthernet interface connects Guest Shell to the server.

When Guest Shell is connected to the server, the device downloads the configuration script, and configures the device. This configuration also includes downloading, setting, and starting of the virtual machine (VM). After the day zero configuration is complete, based on your configuration the system may reboot. Ensure that the system boots with only the user-specific configuration.

Guest Shell Connectivity Using the USB Port

The device uses a serial adapter to connect to multiple other devices. This serial adapter is connected through the USB port that is present on the front panel of the device.

The VM controls the serial adapter, and if there are any changes to the connected devices that are attached to the USB interface while VM is running, the VM is notified.

Stacking with Guest Shell

Guest Shell supports 1+1 high availability. 1+1 high availability is when one device is designated as the active, and the other is designated as the standby. N+1 high availability is not supported.

When Guest Shell is installed, a *guest-share* directory is automatically created in the flash file system. This directory is synchronized across stack members. Any files stored in the *guest-share* folder will be maintained when the active device goes down and the standby takes over. To preserve up to 50 MB of data during high availability switchover, ensure that data is placed in this directory. If the size of the *guest-share* folder is more than 50 MB, it will not be synched to stack members.

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 file system is not maintained. Guest Shell state is internally synchronized across all stack members.

Cisco IOx Overview

Cisco IOx (IOs + linuX) is an 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.

Cisco IOx facilitates the life cycle management of applications and data exchange by providing a set of services that helps developers to package prebuilt applications, and host them on a target device. IOx life cycle management includes distribution, deployment, hosting, starting, stopping (management), and monitoring of

applications and data. IOx services also include application distribution and management tools that help users discover and deploy applications to the IOx framework.

Cisco IOx application hosting provides the following features:

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

IOx Tracing and Logging Overview

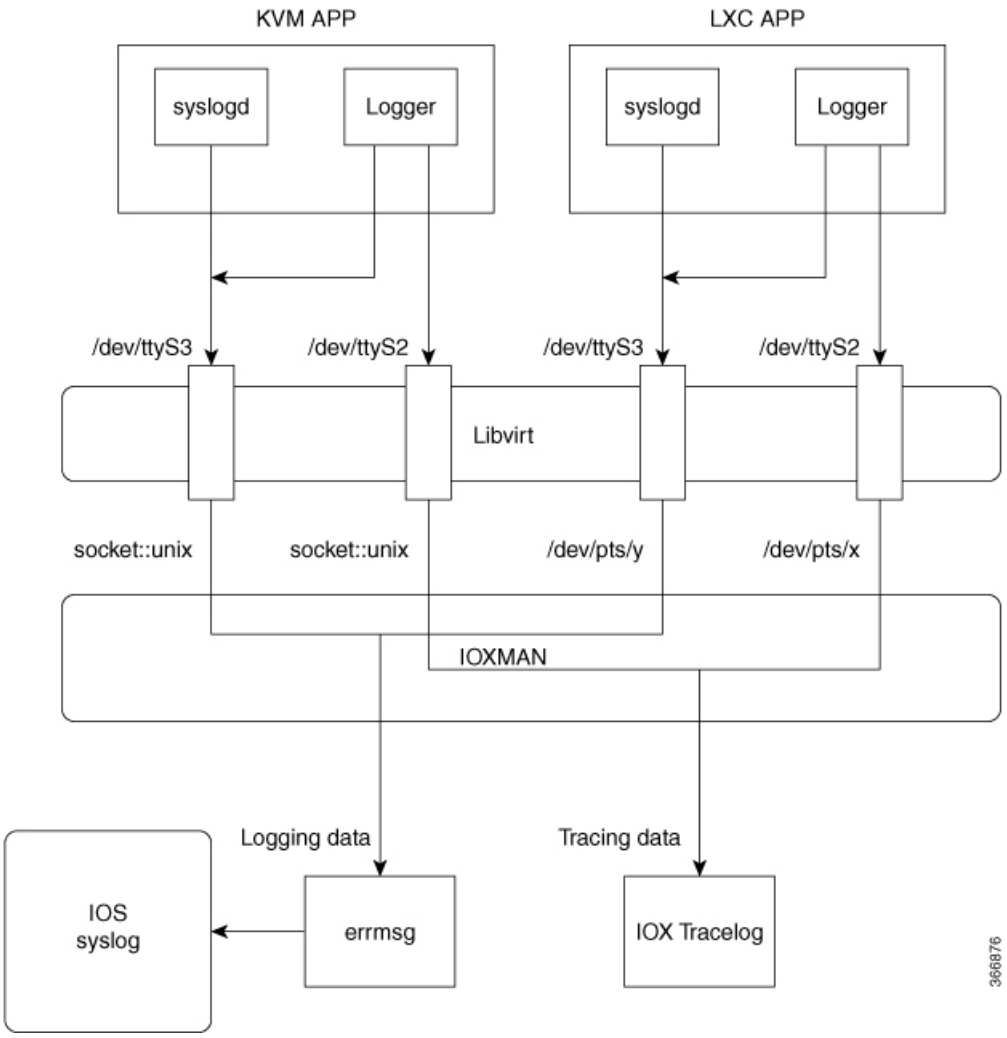
IOx tracing and logging feature allows guest application to run separately on the host device that can help reporting the logging and tracing of the data to the host. The tracing data is saved into IOx tracelog, and the logging data is saved into the Cisco IOS syslog on the host device.

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 the Cisco 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 the 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 the Cisco IOS syslog, to save tracing data to IOx tracelog, and to maintain IOx tracelog for each guest application.

NETCONF Access from Guest Shell

NETCONF-YANG can be accessed from within the Guest Shell, so that users can run Python scripts and invoke Cisco-custom package CLIs using the NETCONF protocol.

The Guest Shell application will establish an SSH connection without a passwordless SSH connection to the localhost and NETCONF port, by using *guestshell* as the username. This username does not correspond to any actual users configured on the device. Even if the device does have a *guestshell* user configured, there is no connection to this passwordless access. Only users with PRIV15 privilege level can access NETCONF from within the Guest Shell.

Authentication and authorization is not bypassed; instead, authentication and authorization happens while granting access to Guest Shell. Only users with the maximum privilege are granted this access.

Users can access the NETCONF service from Guest Shell without opening any external ports. Before connecting to the NETCONF-YANG server on the device, you must run the initializing commands in Guest Shell. These commands are:

```
iosp_client -f netconf_enable guestshell <port-number> and
iosp_client -f netconf_enable_passwordless guestshell <username>
```

The **iosp_client -f netconf_enable guestshell** *port-number* command configures the **netconf-yang ssh local-vrf guestshell** command, and blocks connections until NETCONF-YANG is up and running.

The **iosp_client -f netconf_enable_passwordless guestshell <username>** command creates the SSH keys required for Guest Shell access.

To remove the NETCONF-YANG access from Guest Shell, use the following commands:

```
iosp_client -f netconf_disable guestshell and
iosp_client -f netconf_disable_passwordless guestshell  <username>
```

The **iosp_client -f netconf_disable guestshell** command disables access to NETCONF from within the Guest Shell; however, the NETCONF-YANG configuration will still exist. To shut down NETCONF-YANG, use the **no netconf-yang** command.

The **iosp_client -f netconf_disable_passwordless guestshell** *username* command removes the SSH keys for the specified user. The user will not be able to access NETCONF without a password; however, the user would still be able to connect by using a password.

The *netconf_enable_guestshell* python API runs a combination of the iosp_client functions, *iosp_client -f netconf_enable guestshell 830* and *iosp_client -f netconf_enable_passwordless guestshell guestshell*. This API hides the *unfamiliar-to-user iosp_client* function. When this function is called, it does not return a response until all commands are completed. Unless the function returns an error, you can be sure that NETCONF is running, and the passwordless setup is complete; and you can start creating connections.

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 the 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 the Cisco IOS syslog.

KVM Logging

1. Guest OS enables **/dev/ttyS2** on the 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 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. Starts the timer to send the message to **/dev/log** device on the host using **errmsg**.
8. Data is saved to the Cisco IOS syslog.

LXC Tracing

1. Guest OS enables **/dev/ttyS3** on the guest application.
2. Configures **syslogd** to copy message to **/dev/ttyS3**.
3. Guest application writes data to **/dev/ttyS3**.
4. Libvirt 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 the guest application.
2. Configures syslog to copy the message to **/dev/ttyS3**.
3. Guest application writes data to **/dev/ttyS3**.
4. Libvirt 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 the available data from the socket, sets the severity level 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 explains the logging and tracing of messages in to the Cisco IOS syslog.

Logging Messages in Cisco IOS Syslog

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

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

To comply with the Cisco IOS syslog, the IOXMAN does support severity levels for logging messages. To report logging messages with severity, a guest application must append a 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
```

The 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 a guest application to the Cisco IOS syslog:

1. If you are using C programming, use **write()** to send logging data to the 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 a Shell console, use **echo** to send logging data to the host.

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

Tracing Message to IOx Tracelog

Perform the following steps to report tracing messages from a guest application to IOx tracelog:

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

   ```
   #define SYSLOG_TEST       "tracelog test"

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

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

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

How to Enable the Guest Shell

Managing IOx

Before you begin

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

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **iox**
4. **exit**
5. **show iox-service**
6. **show app-hosting list**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
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.

	Command or Action	Purpose
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.

Example

The following is sample output from the **show iox-service** command:

```
Device# show iox-service

IOx Infrastructure Summary:
---------------------------
IOx service (CAF) 1.10.0.0 : Running
IOx service (HA)           : Running
IOx service (IOxman)       : Running
IOx service (Sec storage) : Not Running
Libvirtd   1.3.4           : Running
Dockerd    18.03.0         : Running
Application DB Sync Info   : Available
Sync Status               : Disabled
```

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, the following message is displayed on the device console.

```
iox feature is not enabled.
```

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.

SUMMARY STEPS

1. **enable**
2. **guestshell enable**
3. **guestshell run** *linux-executable*
4. **guestshell run bash**
5. **guestshell disable**
6. **guestshell destroy**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**guestshell enable** **Example:** `Device# guestshell enable`	Enables the Guest Shell service. **Note** • 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** *linux-executable*
	Example:
	`Device# guestshell run python`
	or
	`Device# guestshell run python3`
		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# guestshell run bash`	Starts a Bash shell to access the Guest Shell.
Step 5	**guestshell disable** **Example:**	Disables the Guest Shell service.

	Command or Action	Purpose
	`Device# guestshell disable`	
Step 6	**guestshell destroy** **Example:** `Device# guestshell destroy`	Deactivates and uninstalls the Guest Shell service.

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, the following message is displayed on the device console.

```
iox feature is not enabled.
```

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 13.

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

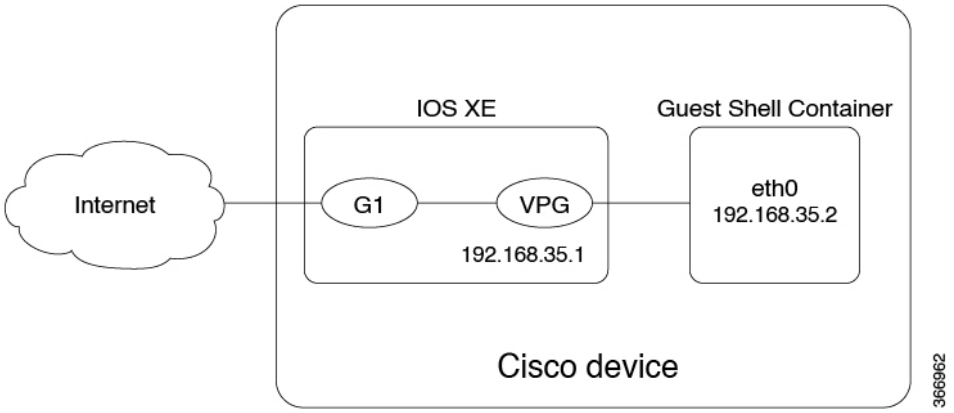
Device(config-if)# interface VirtualPortGroup0
Device(config-if)# ip address 192.168.35.1 255.255.255.0
Device(config-if)# ip nat inside
Device(config-if)# exit

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
Device(config-app-hosting-gateway)# guest-ipaddress 192.168.35.2 netmask 255.255.255.0
Device(config-app-hosting-gateway)# exit
Device(config-app-hosting)# app-default-gateway 192.168.35.1 guest-interface 0
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.

Configuring the AppGigabitEthernet Interface for Guest Shell

Note The following task is applicable only to Catalyst switches that have the AppGigabitEthernet interface. All other Catalyst switches use the management port.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface AppGigabitEthernet** *interface-number*
4. **switchport mode trunk**
5. **exit**
6. **app-hosting appid** *name*
7. **app-vnic AppGigabitEthernet trunk**
8. **vlan** *vlan-ID* **guest-interface** *guest-interface-number*
9. **guest-ipaddress** *ip-address* **netmask** *netmask*
10. **exit**

11. **exit**
12. **app-default-gateway** *ip-address* **guest-interface** *network-interface*
13. **nameserver#** *ip-address*
14. **end**
15. **guestshell enable**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface AppGigabitEthernet** *interface-number* **Example:** `Device(config)# interface AppGigabitEthernet 1/0/1`	Configures the AppGigabitEthernet interface and enters interface configuration mode.
Step 4	**switchport mode trunk** **Example:** `Device(config-if)# switchport mode trunk`	Sets the interface into permanent trunking mode and negotiates to convert the neighboring link into a trunk link.
Step 5	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 6	**app-hosting appid** *name* **Example:** `Device(config)# app-hosting appid guestshell`	Configures an application and enters application-hosting configuration mode.
Step 7	**app-vnic AppGigabitEthernet trunk** **Example:** `Device(config-app-hosting)# app-vnic AppGigabitEthernet trunk`	Configures a trunk port as the front-panel port for application hosting, and enters application-hosting trunk configuration mode.
Step 8	**vlan** *vlan-ID* **guest-interface** *guest-interface-number* **Example:** `Device(config-config-app-hosting-trunk)# vlan 4094 guest-interface 0`	Configures a VLAN guest interface and enters application-hosting VLAN-access IP configuration mode.

	Command or Action	Purpose
Step 9	**guest-ipaddress** *ip-address* **netmask** *netmask* **Example:** `Device(config-config-app-hosting-vlan-access-ip)# guest-ipaddress 192.168.2.2 netmask 255.255.255.0`	(Optional) Configures a static IP address.
Step 10	**exit** **Example:** `Device(config-config-app-hosting-vlan-access-ip)# exit`	Exits application-hosting VLAN-access IP configuration mode and returns to application-hosting trunk configuration mode
Step 11	**exit** **Example:** `Device(config-config-app-hosting-trunk)# exit`	Exits application-hosting trunk configuration mode and returns to application-hosting configuration mode.
Step 12	**app-default-gateway** *ip-address* **guest-interface** *network-interface* **Example:** `Device(config-app-hosting)# app-default-gateway 192.168.2.1 guest-interface 0`	Configures the default management gateway.
Step 13	**nameserver#** *ip-address* **Example:** `Device(config-app-hosting)# name-server0 172.16.0.1`	Configures the Domain Name System (DNS) server.
Step 14	**end** **Example:** `Device(config-app-hosting)# end`	Exits application-hosting configuration mode and returns to privileged EXEC mode.
Step 15	**guestshell enable** **Example:** `Device# guestshell enable`	Enables the Guest Shell service.

Enabling Guest Shell on the Management Interface

Note This task is applicable to Cisco Catalyst 9200 Series Switches, Cisco Catalyst 9300 Series Switches, Cisco Catalyst 9400 Series Switches, Cisco Catalyst 9500 Series Switches, and Cisco Catalyst 9600 Series Switches.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **app-hosting appid** *name*

4. **app-vnic management guest-interface** *interface-number*
5. **end**
6. **show app-hosting list**
7. **guestshell enable**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**app-hosting appid** *name* **Example:** `Device(config)# app-hosting appid guestshell`	Configures an application and enters application-hosting configuration mode.
Step 4	**app-vnic management guest-interface** *interface-number* **Example:** `Device(config-app-hosting)# app-vnic management guest-interface 0`	Configures the management gateway of the virtual network interface and guest interface, and enters application-hosting management-gateway configuration mode.
Step 5	**end** **Example:** `Device(config-app-hosting-mgmt-gateway)# end`	Exits application-hosting management-gateway configuration mode and returns to privileged EXEC mode.
Step 6	**show app-hosting list** **Example:** `Device# show app-hosting list`	Displays the current status of the installed applications. **Note** Guest Shell is displayed in the list of applications, only if it is installed.
Step 7	**guestshell enable** **Example:** `Device# guestshell enable`	Enables the Guest Shell service.

Enabling and Disabling NETCONF Access from Guest Shell

Before you begin

Initialize the following commands from within the Guest Shell to initialize the NETCONF-YANG access:

SUMMARY STEPS

1. **iosp_client -f netconf_enable guestshell** *port-number*
2. **iosp_client -f netconf_enable_passwordless guestshell** *username*
3. **iosp_client -f netconf_disable guestshell**
4. **iosp_client -f netconf_disable_passwordless guestshell** *username*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**iosp_client -f netconf_enable guestshell** *port-number* **Example:** `Guest Shell: iosp_client -f netconf_enable guestshell 3`	Configures the **netconf-yang ssh local-vrf guestshell** command, and blocks connections until NETCONF-YANG is up and running.
Step 2	**iosp_client -f netconf_enable_passwordless guestshell** *username* **Example:** `Guest Shell: iosp_client -f netconf_enable guestshell guestshell`	Creates the SSH keys required for Guest Shell access.
Step 3	**iosp_client -f netconf_disable guestshell** **Example:** `GuestShell: iosp_client -f netconf_disable guestshell`	Removes access to NETCONF from within the Guest Shell. • NETCONF-YANG configuration will still exist. To shut down NETCONF-YANG use the **no netconf-yang** command.
Step 4	**iosp_client -f netconf_disable_passwordless guestshell** *username* **Example:** `Guest Shell: iosp_client -f netconf_disable_passwordless guestshell guestshell`	Removes the access keys for the specified user. • NETCONF access is still enabled for the user; however the user will have to use a password to connect to NETCONF.

Example

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

In Cisco IOS XE Amsterdam 17.1.x to Cisco IOS XE Amsterdam 17.2.x

The following example shows how to enable Guest Shell. In Cisco IOS XE Amsterdam 17.1.x and Cisco IOS XE Amsterdam 17.2.x, both Python V2.7 and Python V3.6 are supported. However, Python V2.7 is the default in these releases.

```
Device> enable
Device# guestshell enable
```

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

Device# guestshell run python
or
Device# guestshell run python3

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.
>>>>>


Device# guestshell run bash

[guestshell@guestshell ~]$

Device# guestshell disable

Guestshell disabled successfully

Device# guestshell destroy

Guestshell destroyed successfully
```

In Cisco IOS XE Amsterdam 17.3.1 and Later Releases

The following example shows how to enable Guest Shell. In Cisco IOS XE Amsterdam 17.3.1 and later releases, only Python V3.6 is supported.

```
Device> enable
Device# guestshell enable

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

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.>>>>>

>>>>>

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
Device(config-app-hosting-gateway)# guest-ipaddress 192.168.35.2 netmask 255.255.255.0
Device(config-app-hosting-gateway)# exit
Device(config-app-hosting)# app-resource profile custom
Device(config-app-resource-profile-custom)# cpu 1500
Device(config-app-resource-profile-custom)# memory 512
Device(config-app-resource-profile-custom)# end

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

Example: Configuring the AppGigabitEthernet Interface for Guest Shell



Note The following task is applicable only to Catalyst switches that have the AppGigabitEthernet interface. All other Catalyst switches use the management port.

The following example shows how to configure an AppGigabitEthernet interface for Guest Shell. Here, VLAN 4094 creates a Network Address Translation (NAT) this is used for Guest Shell. VLAN 1 is an external interface.

```
Device> enable
Device# configure terminal
Device(config)# ip nat inside source list NAT_ACL interface vlan 1 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
Device(config)# vlan 4094
Device(config-vlan)# exit
Device(config)# interface vlan 4094
Device(config-if)# ip address 192.168.2.1 255.255.255.0
Device(config-if)# ip nat inside
Device(config-if)# exit
Device(config)# interface vlan 1
Device(config-if)# ip nat outside
Device(config-if)# exit
Device(config)# ip routing
Device(config)# ip route 0.0.0.0 0.0.0.0 209.165.201.1
Device(config)# interface AppGigabitEthernet 1/0/1
Device(config-if)# switchport mode trunk
Device(config-if)# exit
Device(config)# app-hosting appid guestshell
Device(config-app-hosting)# app-vnic AppGigEthernet trunk
Device(config-config-app-hosting-trunk)# vlan 4094 guest-interface 0
Device(config-config-app-hosting-vlan-access-ip)# guest-ipaddress 192.168.2.2 netmask
255.255.255.0
Device(config-config-app-hosting-vlan-access-ip)# exit
Device(config-config-app-hosting-trunk)# exit
Device(config-app-hosting)# app-default-gateway 192.168.2.1 guest-interface 0
Device(config-app-hosting)# name-server0 172.16.0.1
Device(config-app-hosting)# name-server1 198.51.100.1
Device(config-app-hosting)# end
Device# guestshell enable
```

Example: Enabling Guest Shell on the Management Interface

This example is applicable to Cisco Catalyst 9200 Series Switches, Cisco Catalyst 9300 Series Switches, Cisco Catalyst 9400 Series Switches, Cisco Catalyst 9500 Series Switches, and Cisco Catalyst 9600 Series Switches.

```
Device> enable
Device# configure terminal
Device(config)# app-hosting appid guestshell
Device(config-app-hosting)# app-vnic management guest-interface 0
Device(config-app-hosting-mgmt-gateway)# end
```

```
Device# guestshell enable
```

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 5.4.85 #1 SMP Tue Dec 22 10:50:44 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
```

Cisco 4000 Series Integrated Services Routers use the **dohost** provided by CentOS Linux release 7.1.1503.



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

Example: Guest Shell Networking Configuration

The following is a sample Guest Shell networking configuration.

- 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-bost 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
Python module	CLI Python Module
Zero-Touch Provisioning	Zero-Touch Provisioning

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. 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. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	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 4: 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	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. In Cisco IOS XE Everest 16.5.1a, this feature was implemented on the following platforms: • Cisco Catalyst 9300 Series Switches Cisco Catalyst 9500 Series Switches In Cisco IOS Everest 16.5.1b, this feature was implemented on the following platform: • 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	n Cisco IOS XE Fuji 16.7.1, this feature was implemented on the following platforms: • Cisco ASR 1000 Series Aggregation Services Routers • Cisco Cloud Services Router 1000v Series In Cisco IOS XE Fuji 16.7.1, Logging and Tracing feature support was implemented on Cisco ASR 1000 Aggregation Services Routers.

Feature Name	Release	Feature Information
	Cisco IOS XE Fuji 16.8.1	In Cisco IOS XE Fuji 16.8.1, this feature was implemented on Cisco Catalyst 9500-High Performance Series Switches.
	Cisco IOS XE Fuji 16.9.1	In Cisco IOS XE Fuji 16.9.1, this feature was implemented on Cisco 1000 Series Integrated Services Routers.
	Cisco IOS XE Gibraltar 16.11.1b	In Cisco IOS XE Gibraltar 16.11.1b, this feature was implemented on the following platforms: • Cisco Catalyst 9800-40 Wireless Controllers • Cisco Catalyst 9800-80 Wireless Controllers
	Cisco IOS XE Gibraltar 16.12.1	In Cisco IOS XE Gibraltar 16.12.1, this feature was implemented on the following platforms: • Cisco Catalyst 9200 Series Switches **Note** This feature is not supported on C9200L SKUs. • Cisco Catalyst 9300L SKUs • Cisco Catalyst 9600 Series Switches
	Cisco IOS XE Amsterdam 17.3.1	In Cisco IOS XE Amsterdam 17.3.1, this feature was implemented on the following platforms: • Cisco Catalyst 8200 Series Edge Platforms • Cisco Catalyst 8300 Series Edge Platforms • Cisco Catalyst 8500 and 8500L Series Edge Platforms
	Cisco IOS XE Cupertino 17.7.1	

Feature Name	Release	Feature Information
		In Cisco IOS XE Gibraltar 17.7.1, this feature was implemented on the following platforms: • Cisco Catalyst 9500X Series Switches • Cisco Catalyst 9600X Series Switches
NETCONF Access from Guest Shell	Cisco IOS XE 17.6.1	NETCONF can be accessed from within the Guest Shell, so that users can run Python scripts and invoke Cisco-custom package CLIs using the NETCONF protocol. In 17.6.1, this feature was implemented on the following platforms: • Cisco Catalyst 9200 Series Switches • Cisco Catalyst 9300 and 9300L Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 and 9500-High Performance Series Switches • Cisco Catalyst 9600 Series Switches
Python 3 Support in Guest Shell	Cisco IOS XE Amsterdam 17.1.1	Python Version 3.6 is supported in Guest Shell. Python Version 3.6 is available on all supported platforms.