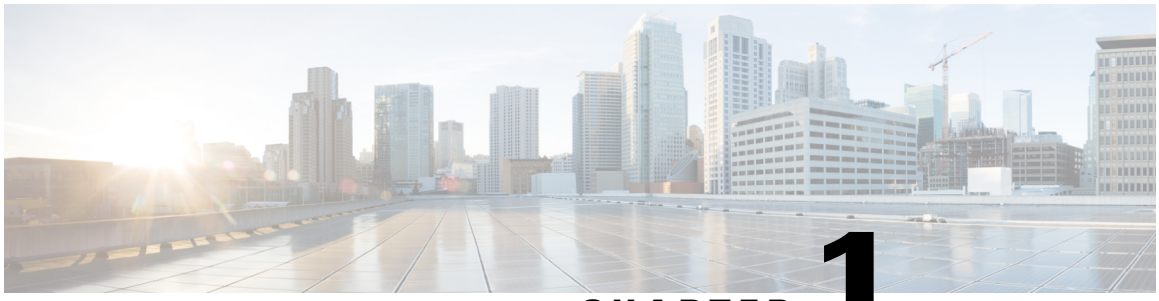
Data Models Configuration Guide

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS



CHAPTER 1

YANG Infrastructure

The Polaris Yang Infrastructure 16.3 feature facilitates a programmatic and standards-based way of writing configurations and reading operational data from network devices.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table at the end of this module.

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

Introduction to Data Models - Programmatic and Standards-based Configuration

The traditional way of managing network devices is by using Command Line Interfaces (CLIs) for configurational (configuration commands) and operational data (show commands). For network management, Simple Network Management Protocol (SNMP) is widely used, especially for exchanging management information between various network devices. Although CLIs and SNMP are heavily used, they have several restrictions. CLIs are highly proprietary, and human intervention is required to understand and interpret their text-based specification. SNMP does not distinguish between configurational and operational data.

The solution lies in adopting a programmatic and standards-based way of writing configurations to any network device, replacing the process of manual configuration. Network devices running on Cisco IOS XE support the automation of configuration for multiple devices across the network using data models. Data models are developed in a standard, industry-defined language, that can define configuration and state information of a network.

Cisco IOS XE supports the Yet Another Next Generation (YANG) data modeling language. YANG can be used with the Network Configuration Protocol (NETCONF) to provide the desired solution of automated and programmable network operations. NETCONF (RFC 6241) is an XML-based protocol that client applications use to request information from and make configuration changes to the device. YANG is primarily used to model the configuration and state data used by NETCONF operations.

In Cisco IOS XE, model-based interfaces interoperate with existing device CLI, Syslog, and SNMP interfaces. These interfaces are optionally exposed northbound from network devices. YANG is used to model each protocol based on RFC 6020.

Note To access Cisco YANG models in a developer-friendly way, please clone the GitHub repository *https://github.com/YangModels/yang*, and visit the *vendor/cisco* (*https://github.com/YangModels/yang/tree/master/vendor/cisco*) subdirectory. Models for various release of IOS-XE, IOS-XR, and NX-OS platforms are available here.

NETCONF

NETCONF provides a simpler mechanism to install, manipulate, and delete the configuration of network devices.

It uses an Extensible Markup Language (XML)-based data encoding for the configuration data as well as the protocol messages.

NETCONF uses a simple RPC-based (Remote Procedure Call) mechanism to facilitate communication between a client and a server. The client can be a script or application typically running as part of a network manager. The server is typically a network device (switch or router). It uses Secure Shell (SSH) as the transport layer across network devices.

NETCONF also supports capability discovery and model downloads. Supported models are discovered using the *ietf-netconf-monitoring* model. Revision dates for each model are shown in the capabilities response. Data models are available for optional download from a device using the *get-schema* rpc. You can use these YANG models to understand or export the data model.

For more details, refer RFC 6241.

Configuring NETCONF

Before You Begin

You must configure NETCONF-YANG as follows.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **netconf-yang**
4. **exit**

DETAILED STEPS

	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	**netconf-yang** **Example:** `Device (config)# netconf-yang`	Enables the NETCONF interface on your network device. **Note** After the initial enablement through the CLI, network devices can be managed subsequently through a model based interface. The complete activation of model-based interface processes may require up to 90 seconds.
Step 4	**exit** **Example:** `Device (config)# exit`	Exits global configuration mode.

Configuring NETCONF Options

Configuring SNMP

Enable the SNMP Server in IOS to enable NETCONF to access SNMP MIB data using YANG models generated from supported MIBs, and to enable supported SNMP traps in IOS to receive NETCONF notifications from the supported traps.

Perform the following steps:

SUMMARY STEPS

1. Enable SNMP features in IOS.
2. After NETCONF-YANG starts, enable SNMP Trap support by sending the following RPC <edit-config> message to the NETCONF-YANG port.
3. Send the following RPC message to the NETCONF-YANG port to save the running configuration to the startup configuration.

DETAILED STEPS

Step 1 Enable SNMP features in IOS.

Example:

```
configure terminal
logging history debugging
logging snmp-trap emergencies
logging snmp-trap alerts
logging snmp-trap critical
logging snmp-trap errors
logging snmp-trap warnings
logging snmp-trap notifications
logging snmp-trap informational
logging snmp-trap debugging
!
snmp-server community public RW
snmp-server trap link ietf
snmp-server enable traps snmp authentication linkdown linkup snmp-server enable traps syslog
snmp-server manager
exit
```

Step 2 After NETCONF-YANG starts, enable SNMP Trap support by sending the following RPC <edit-config> message to the NETCONF-YANG port.

Example:

```
<?xml version="1.0" encoding="utf-8"?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <netconf-yang xmlns="http://cisco.com/yang/cisco-self-mgmt">
        <cisco-ia xmlns="http://cisco.com/yang/cisco-ia">
          <snmp-trap-control>
            <trap-list>
              <trap-oid>1.3.6.1.4.1.9.9.41.2.0.1</trap-oid>
            </trap-list>
            <trap-list>
              <trap-oid>1.3.6.1.6.3.1.1.5.3</trap-oid>
            </trap-list>
            <trap-list>
              <trap-oid>1.3.6.1.6.3.1.1.5.4</trap-oid>
            </trap-list>
          </snmp-trap-control>
        </cisco-ia>
      </netconf-yang>
    </config>
```

```
  </edit-config>
</rpc>
```

Step 3 Send the following RPC message to the NETCONF-YANG port to save the running configuration to the startup configuration.

Example:

```
<?xml version="1.0" encoding="utf-8"?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="">
  <cisco-ia:save-config xmlns:cisco-ia="http://cisco.com/yang/cisco-ia"/>
</rpc>
```

Configuring Operational Data Manager

The NETCONF Operational Data Manager provides the ability to read operational data using YANG models. Each ODM parser polls the specified operational data according to the specified polling interval in milliseconds.

Perform the following steps:

SUMMARY STEPS

1. After NETCONF-YANG starts, send the following RPC <edit-config> message to the NETCONF-YANG port.
2. Send the following RPC message to the NETCONF-YANG port to save the running configuration to the startup configuration.

DETAILED STEPS

Step 1 After NETCONF-YANG starts, send the following RPC <edit-config> message to the NETCONF-YANG port.

Example:

```
<?xml version="1.0" encoding="utf-8"?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <netconf-yang xmlns="http://cisco.com/yang/cisco-self-mgmt">
        <cisco-odm xmlns="http://cisco.com/yang/cisco-odm">
          <polling-enable>true</polling-enable>
          <on-demand-default-time>30000</on-demand-default-time>
          <on-demand-enable>false</on-demand-enable>
          <actions>
            <action-name>parse.showACL</action-name>
            <polling-interval>120000</polling-interval>
            <mode>poll</mode>
          </actions>
          <actions>
            <action-name>parse.showArchive</action-name>
            <polling-interval>120000</polling-interval>
            <mode>poll</mode>
```

```
          </actions>
          <actions>
            <action-name>parse.showEnvironment</action-name>
            <polling-interval>120000</polling-interval>
            <mode>poll</mode>
          </actions>
          <actions>
            <action-name>parse.showFlowMonitor</action-name>
            <polling-interval>120000</polling-interval>
            <mode>poll</mode>
          </actions>
          <actions>
            <action-name>parse.showInterfaces</action-name>
            <polling-interval>120000</polling-interval>
            <mode>poll</mode>
          </actions>
          <actions>
            <action-name>parse.showIpRoute</action-name>
            <polling-interval>120000</polling-interval>
            <mode>poll</mode>
          </actions>
          <actions>
            <action-name>parse.showMemoryStatistics</action-name>
            <polling-interval>120000</polling-interval>
            <mode>poll</mode>
          </actions>
          <actions>
            <action-name>parse.showPlatformSoftware</action-name>
            <polling-interval>120000</polling-interval>
            <mode>poll</mode>
          </actions>
          <actions>
            <action-name>parse.showProcessesCPU</action-name>
            <polling-interval>120000</polling-interval>
            <mode>poll</mode>
          </actions>
          <actions>
            <action-name>parse.showProcessesMemory</action-name>
            <polling-interval>120000</polling-interval>
            <mode>poll</mode>
          </actions>
        </cisco-odm>
      </netconf-yang>
    </config>
  </edit-config>
</rpc>
```

The <edit-config> can be modified to enable or disable specific actions and/or to choose a different update frequency.

Step 2 Send the following RPC message to the NETCONF-YANG port to save the running configuration to the startup configuration.

Example:

```
<?xml version="1.0" encoding="utf-8"?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="">
  <cisco-ia:save-config xmlns:cisco-ia="http://cisco.com/yang/cisco-ia"/>
</rpc>
```

Additional References

Related Documents

Related Topic	Document Title
YANG data models for various release of IOS-XE, IOS-XR, and NX-OS platforms	Clone the GitHub repository *https://github.com/YangModels/yang* and visit the *vendor/cisco* (*https://github.com/YangModels/yang/tree/master/vendor/cisco*) subdirectory.

Standards and RFCs

Standard/RFC	Title
RFC 6020	*YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)*
RFC 6241	Network Configuration Protocol (NETCONF)

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

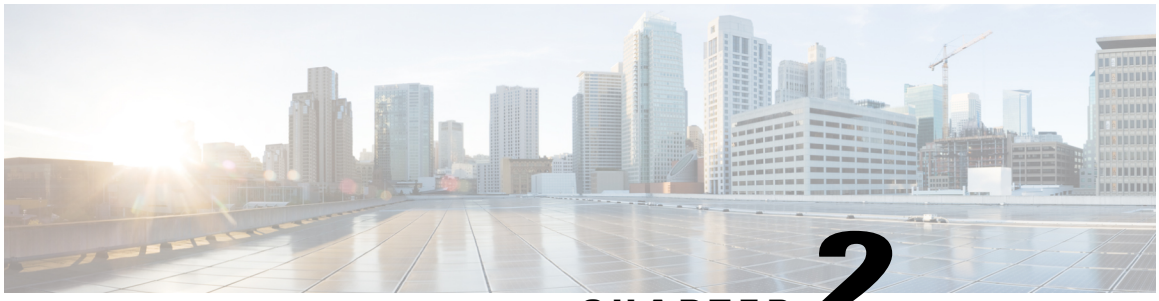
Feature Information for YANG Infrastructure

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 1: Feature Information for YANG Data Model

Feature Name	Releases	Feature Information
Polaris Yang Infrastructure 16.3	Cisco IOS XE Denali 16.3.1	The Polaris Yang Infrastructure 16.3 feature facilitates a programmatic and standards-based way of writing configurations and reading operational data from network devices. The following commands were introduced or modified: **netconf-yang**.



CHAPTER 2

In Service Model Update

This module describes how to update the YANG data models on a device through an In Service Model Update.

This module contains the following sections:

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table at the end of this module.

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

Restrictions for In Service Model Update

- In Service Model Update does not support In-Service Software Upgrade (ISSU).
- After a switchover, users must install the Software Maintenance Update (SMU) on the standby device.

Information About In Service Model Updates

Overview of In Service Model Updates

In Service Model Update adds new data models or extend functionality to existing data models. The In Service Model Update provides YANG model enhancements outside of a release cycle. The update package is a superset of all existing models; it includes all existing models as well as updated YANG models.

The data model infrastructure implements the YANG model-defined management interfaces for Cisco IOS XE devices. The data model infrastructure exposes the NETCONF interface northbound from Cisco IOS XE devices. The supported data models include industry standard models such as IETF, and Cisco IOS XE device-specific models.

The functionality provided by the In Service Model Update is integrated into the subsequent Cisco IOS XE software maintenance release. Data model update packages can be downloaded from the Cisco Download Software Center.

Compatibility of In Service Model Update Packages

An update package is built on a per image basis.

All contents of an update package will be part of future mainline or maintenance release images. The image and platform versions are checked by the In Service Model Update commands during the package add and activate. If an image or platform mismatch occurs, the package install fails.

Update Package Naming Conventions

In Service Model Updates are packaged as a .bin files. This file includes all updates for a specific release and platform and the Readme file. These files have a release date and are updated periodically with additional model updates.

The naming convention of the data model update package follows the format—platform type-license level.release version.DDTS ID-file. The following is an example of a data model update file:

- asr1000-universalk9.2017-08-23_17.48.0.CSCxxxxxxx.SSA.dmp.bin

The readme file provides the following information:

- Console and error messages during data model activation or deactivation
- Data model installation impact
- Side effects and possible workarounds
- Package(s) that the In Service Model Update impacts
- Restart type

Installing the Update Package

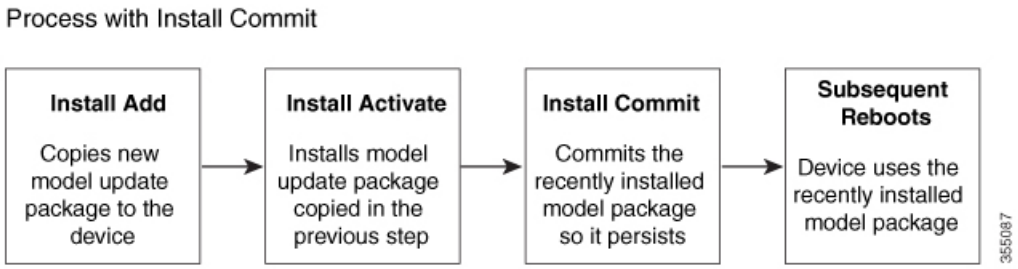
You can install the In Service Model Update package on a device by using the **install add**, **install activate**, and **install commit** commands in privileged EXEC mode.

The **install add** command copies the update package from a remote location to the device. You can also use other methods to copy the package; however, you must still enable the **install add** command for the installation to work. For the **install activate** command to work, the package must be available in the device bootflash. Enable the **install commit** command to make updates persistent over reloads.

Installing an update replaces any previously installed data models. At any time, only one update is installed on the device. A data model package includes all updated YANG models and all existing YANG models previously installed on the device.

The following flow chart explains how the model update package works:

Figure 1: Committing a Model Update Package



If NETCONG-YANG is enabled during package activation, NETCONF processes are restarted. All active NETCONF sessions are killed during package activation. Failure during a package verification aborts the activation process.

Deactivating the Update Package

You can deactivate an update package by using the **install deactivate** command. Enable the **install commit** command to make changes persistent.

Table 2: Deactivating a Model Update Package

Action	Command to Use
To remove a package.	Use the **install remove** command. **Note** Deactivate a package before removing it.

Action	Command to Use
To deactivate a package	Use the **install deactivate** command, followed by the **install commit** command. **Note** The **install commit** command must be used to ensure that the deactivation of the model package is persistent across reloads. Subsequent attempts at removal of the package will fail, if the deactivation is not committed.

When you deactivate an update, if more than one model update package is installed, the most recently committed model update package becomes the model package used by the device. If there are no other previously committed model packages, then the base version of data models included with the standard image is used.

Rollback of the Update Package

Rollback provides a mechanism to move a device back to the state in which it was operating prior to an update. After a rollback, NETCONF-YANG processes are restarted before changes are visible.

You can roll back an update to the base version, the last committed version, or a known commit ID by using the **install rollback** command.

How to Manage In Service Software Updates

Managing the Update Package

SUMMARY STEPS

1. **enable**
2. **install add file tftp:** *filename*
3. **install activate file bootflash:** *filename*
4. **install commit**
5. **install deactivate file bootflash:** *filename*
6. **install commit**
7. **install rollback to** {**base** | **committed** | **id** *commit-ID*}
8. **install remove** {**file bootflash:** *filename* | **inactive**}
9. **show install summary**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**install add file tftp:** *filename* **Example:** `Device# install add file tftp://172.16.0.1//tftpboot/folder1/ asr1000-universalk9.2017-08-23_17.48.0.CSCxxxxxxx.SSA.dmp.bin`	Copies the model update package from a remote location (via FTP, TFTP) to the device, and performs a compatibility check for the platform and image versions. • You can use other methods to copy the update package from the remote location to the device, however; you still have to execute the **install add** command before the package is activated.
Step 3	**install activate file bootflash:** *filename* **Example:** `Device# install activate file bootflash:asr1000-universalk9.2017-08-23_17.48.0.CSCxxxxxxx.SSA.dmp.bin`	Validates whether the update package is added through the **install add** command, and restarts the NETCONF processes. • Perform the **install add** operation prior to activating an update package.
Step 4	**install commit** **Example:** `Device# install commit`	Makes the changes persistent over reload. • NETCONF processes are not restarted.
Step 5	**install deactivate file bootflash:** *filename* **Example:** `Device# install deactivate file bootflash: asr1000-universalk9.2017-08-23_17.48.0.CSCxxxxxxx.SSA.dmp.bin`	Deactivates the specified update package, and restarts the NETCONF processes.
Step 6	**install commit** **Example:** `Device# install commit`	Makes the changes persistent over reload. • NETCONF processes are not restarted.

	Command or Action	Purpose
Step 7	**install rollback to** {**base** | **committed** | **id** *commit-ID*} **Example:** `Device# install rollback to base`	Rolls back the update to the base version, the last committed version, or a known commit ID, and restarts NETCONF processes. • Valid values for the *commit-id* argument are from 1 to 4294967295. • Older versions of data models updates are available for use.
Step 8	**install remove** {**file bootflash:** *filename* | **inactive**} **Example:** `Device# install remove file bootflash: √asr1000-universalk9.2017-08-23_17.48.0.CSCxxxxxxx.SSA.dmp.bin`	Removes the specified update package from the bootflash. • A package must be deactivated before it is removed.
Step 9	**show install summary** **Example:** `Device# show install summary`	Displays information about the active package. • The output of this command varies according to the **install** commands that are configured.

Configuration Examples for In Service Software Updates

Example: Managing an Update Package

The following example shows how to add a model update package file:

```
Device# install add file tftp://172.16.0.1//tftpboot/folder1/
asr1000-universalk9.2017-08-23_17.48.0.CSCxxxxxxx.SSA.dmp.bin

install_add: START Sun Feb 26 05:57:04 UTC 2017
Downloading file
tftp://172.16.0.1//tftpboot/folder1/asr1000-universalk9.2017-08-23_17.48.0.CSCxxxxxxx.SSA.dmp.bin
Finished downloading file
tftp://172.16.0.1//tftpboot/folder1/asr1000-universalk9.2017-08-23_17.48.0.CSCxxxxxxx.SSA.dmp.bin
to bootflash:asr1000-universalk9.2017-08-23_17.48.0.CSCxxxxxxx.SSA.dmp.bin
SUCCESS: install_add /bootflash/asr1000-universalk9.2017-08-23_17.48.0.CSCxxxxxxx.SSA.dmp.bin

Sun Feb 26 05:57:22 UTC 2017
Device#
```

The following is sample output from the **show install summary** command after adding an update package file to the device:

```
Device# show install summary
```

```
Active Packages:
No packages
Inactive Packages:
bootflash: isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin
Committed Packages:
No packages
Uncommitted Packages:
No packages
Device#
```

The following example shows how to activate an added update package file:

```
Device# install activate file bootflash:
asr1000-universalk9.2017-08-23_17.48.0.CSCxxxxxxx.SSA.dmp.bin

install_activate: START Sun Feb 26 05:58:41 UTC 2017
DMP package.
Netconf processes stopped
SUCCESS: install_activate
/bootflash/asr1000-universalk9.2017-08-23_17.48.0.CSCxxxxxxx.SSA.dmp.bin
Sun Feb 26 05:58:58 UTC 2017*Feb 26 05:58:47.655: %DMI-4-CONTROL_SOCKET_CLOSED:
SIP0: nesd: Confd control socket closed Lost connection to ConfD (45): EOF on socket to
ConfD.
*Feb 26 05:58:47.661: %DMI-4-SUB_READ_FAIL: SIP0: vtyserverutild:
Confd subscription socket read failed Lost connection to ConfD (45):
EOF on socket to ConfD.
*Feb 26 05:58:47.667: %DMI-4-CONTROL_SOCKET_CLOSED: SIP0: syncfd:
Confd control socket closed Lost connection to ConfD (45): EOF on socket to ConfD.
*Feb 26 05:59:43.269: %DMI-5-SYNC_START: SIP0: syncfd:
External change to running configuration detected.
The running configuration will be synchronized to the NETCONF running data store.
*Feb 26 05:59:44.624: %DMI-5-SYNC_COMPLETE: SIP0: syncfd:
The running configuration has been synchronized to the NETCONF running data store.
Device#
```

The following sample output from the **show install summary** command displays the status of the model package as active and uncommitted:

```
Device# show install summary

Active Packages:
bootflash:asr1000-universalk9.2017-08-23_17.48.0.CSCxxxxxxx.SSA.dmp.bin
Inactive Packages:
No packages
Committed Packages:
No packages
Uncommitted Packages:
bootflash:asr1000-universalk9.2017-08-23_17.48.0.CSCxxxxxxx.SSA.dmp.bin
Device#
```

The following example shows how to execute the **install commit** command:

```
Device# install commit

install_commit: START Sun Feb 26 06:46:48 UTC 2017
SUCCESS: install_commit Sun Feb 26 06:46:52 UTC 2017
Device#
```

The following sample output from the **show install summary** command displays that the update package is now committed, and that it will be persistent across reloads:

```
Device# show install summary

Active Packages:
bootflash:asr1000-universalk9.2017-08-23_17.48.0.CSCxxxxxxx.SSA.dmp.bin
Inactive Packages:
No packages
Committed Packages:
bootflash:asr1000-universalk9.2017-08-23_17.48.0.CSCxxxxxxx.SSA.dmp.bin
Uncommitted Packages:
No packages
```

```
Device#
```

The following example shows how to rollback an update package to the base package:

```
Device# install rollback to base

install_rollback: START Sun Feb 26 06:50:29 UTC 2017
7 install_rollback: Restarting impacted processes to take effect
7 install_rollback: restarting confd
*Feb 26 06:50:34.957: %DMI-4-CONTROL_SOCKET_CLOSED: SIP0: syncfd:
Confd control socket closed Lost connection to ConfD (45): EOF on socket to ConfD.
*Feb 26 06:50:34.962: %DMI-4-CONTROL_SOCKET_CLOSED: SIP0: nesd:
Confd control socket closed Lost connection to ConfD (45): EOF on socket to ConfD.
*Feb 26 06:50:34.963: %DMI-4-SUB_READ_FAIL: SIP0: vtyserverutild:
Confd subscription socket read failed Lost connection to ConfD (45):
EOF on socket to ConfD.Netconf processes stopped
7 install_rollback: DMP activate complete
SUCCESS: install_rollback Sun Feb 26 06:50:41 UTC 2017
*Feb 26 06:51:28.901: %DMI-5-SYNC_START: SIP0: syncfd:
External change to running configuration detected.
The running configuration will be synchronized to the NETCONF running data store.
*Feb 26 06:51:30.339: %DMI-5-SYNC_COMPLETE: SIP0: syncfd:
The running configuration has been synchronized to the NETCONF running data store.
Device#
```

The following is sample output from the **show install package** command:

```
Device# show install package bootflash:
asr1000-universalk9.2017-08-23_17.48.0.CSCxxxxxxx.SSA.dmp.bin

Name: asr1000-universalk9.2017-08-23_17.48.0.CSCxxxxxxx.SSA.dmp.bin
Version: 16.7.1.0.199.1484082952..Everest
Platform: ASR1000
Package Type: dmp
Defect ID: CSCxxxxxxx
Package State: Added
Supersedes List: {}
Smu ID: 1
Device#
```

The following sample NETCONF hello message verifies the new data model package version:

```
Getting Capabilities: (admin @ 172.16.0.1:830)
PROTOCOL netconf
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<capabilities>
<capability>urn:ietf:params:netconf:base:1.0</capability>
<capability>urn:ietf:params:netconf:base:1.1</capability>
<capability>urn:ietf:params:netconf:capability:writable-running:1.0</capability>
<capability>urn:ietf:params:netconf:capability:xpath:1.0</capability>
<capability>urn:ietf:params:netconf:capability:validate:1.0</capability>
<capability>urn:ietf:params:netconf:capability:validate:1.1</capability>
<capability>urn:ietf:params:netconf:capability:rollback-on-error:1.0</capability>
<capability>urn:ietf:params:netconf:capability:notification:1.0</capability>
<capability>urn:ietf:params:netconf:capability:interleave:1.0</capability>
<capability>http://tail-f.com/ns/netconf/actions/1.0</capability>
<capability>http://tail-f.com/ns/netconf/extensions</capability>
<capability>urn:ietf:params:netconf:capability:with-defaults:1.0?basic-mode=
explicit&amp;also-supported=report-all-tagged</capability>
<capability>urn:ietf:params:xml:ns:yang:ietf-netconf-with-defaults?
revision=2011-06-01&amp;module=ietf-netconf-with-defaults</capability>
<capability>http://cisco.com/ns/yang/Cisco-IOS-XE-aaa?module=
Cisco-IOS-XE-aaa&amp;revision=2017-02-07</capability>
<<capability>http://cisco.com/ns/yang/Cisco-IOS-XE-native?module=
Cisco-IOS-XE-native&amp;revision=2017-01-07&amp;features=virtual-
template,punt-num,multilink,eth-evc,esmc,efp,dot1x</capability>
Device#
```

The following is sample output from the **show install log** command:

```
Device# show install log

[0|install_op_boot]: START Fri Feb 24 19:20:19 Universal 2017
[0|install_op_boot]: END SUCCESS Fri Feb 24 19:20:23 Universal 2017
[3|install_add]: START Sun Feb 26 05:55:31 UTC 2017
[3|install_add( FATAL)]: File path (scp) is not yet supported for this command
[4|install_add]: START Sun Feb 26 05:57:04 UTC 2017
[4|install_add]: END SUCCESS
/bootflash/asr1000-universalk9.2017-08-23_17.48.0.CSCxxxxxxx.SSA.dmp.bin
Sun Feb 26 05:57:22 UTC 2017
[5|install_activate]: START Sun Feb 26 05:58:41 UTC 2017
Device#
```

Additional References for In Service Model Updates

Related Documents

Related Topic	Document Title
Programmability commands	

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

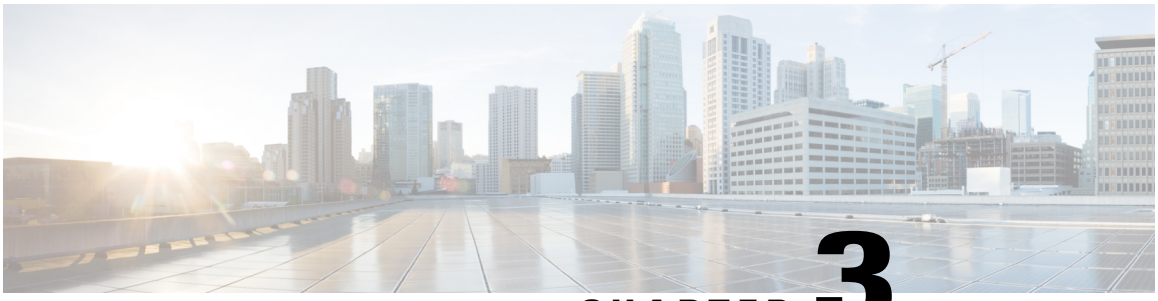
Feature Information for In Service Model Update

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 3: Feature Information for In Service Model Update

Feature Name	Release	Feature Information
In Service Model Update		This module describes how to update YANG data models through In Service Model Update. This feature is supported on the following platforms: The following commands were introduced or updated: **install (Programmability)**, **show install (Programmability)**.



CHAPTER 3

Zero-Touch Provisioning

To address network provisioning challenges, Cisco introduces a zero-touch provisioning model. This module describes the Zero-Touch Provisioning feature.

Note The Zero-Touch Provisioning feature is enabled automatically; no configuration is required.

Information About Zero-Touch Provisioning

Zero-Touch Provisioning Overview

Zero-Touch Provisioning provides open bootstrap interfaces to automate network device provisioning in heterogeneous network environments.

When a device that supports Zero-Touch Provisioning boots up, and does not find the startup configuration (during initial installation), the device enters the Zero-Touch Provisioning mode. The device searches for a Dynamic Host Control Protocol (DHCP) server, bootstraps itself with its interface IP address, gateway, and Domain Name System (DNS) server IP address, and enables Guest Shell. The device then obtains the IP address or URL of an HTTP/TFTP server, and downloads the Python script from an HTTP/TFTP server to configure the device.

Guest Shell provides the environment for the Python script to run. Guest Shell executes the downloaded Python script and applies an initial configuration to the device.

After initial provisioning is complete, Guest Shell remains enabled. For more information, see the *Guest Shell* chapter.

Note In case Zero-Touch Provisioning fails, the device falls back to AutoInstall to load configuration files. For more information, see Using AutoInstall and Setup.

DHCP Server Configuration for Zero-Touch Provisioning

In Zero-Touch Provisioning, a DHCP server must be running on the same network as the new device that is being provisioned. Zero-Touch Provisioning is supported on both management ports and in-band ports.

When the new device is switched on, it retrieves the IP address information of the HTTP/TFTP server where the Python script resides, and the folder path of the Python script from the DHCP server. For more information on Python Scripts, see the *Python API* and *Python CLI Module* chapters.

The DHCP server responds to DHCP discovery events with the following options:

- Option 150—(Optional) Contains a list of IP addresses that points to the HTTP/TFTP server on the management network that hosts the Python scripts to be run.
- Option 67—Contains the Python script file path on the HTTP/TFTP server.

After receiving these DHCP options, the device connects to the HTTP/TFTP server, and downloads the Python script. The device, at this point does not have any route to reach the HTTP/TFTP server, so it uses the default route provided by the DHCP server.

Sample Zero-Touch Provisioning Configurations

Sample DHCP Server Configuration on a Management Port Using TFTP Copy

The following is a sample DHCP server configuration using TFTP copy, when connected via the management port on a device:

```
Device> enable
Device# configure terminal
Device(config)# ip dhcp excluded-address 10.1.1.1
Device(config)# ip dhcp excluded-address vrf Mgmt-vrf 10.1.1.1 10.1.1.10
Device(config)# ip dhcp pool pnp_device_pool
Device(config-dhcp)# vrf Mgmt-vrf
Device(config-dhcp)# network 10.1.1.0 255.255.255.0
Device(config-dhcp)# default-router 10.1.1.1
Device(config-dhcp)# option 150 ip 203.0.113.254
Device(config-dhcp)# option 67 ascii /sample_python_dir/python_script.py
Device(config-dhcp)# exit
Device(config)# interface gigabitethernet 1/0/2
Device(config-if)# no ip dhcp client request tftp-server-address
Device(config-if)# end
```

Sample DHCP Server Configuration on a Management Port Using HTTP Copy

The following is a sample DHCP server configuration using HTTP copy, when connected via the management port on a device:

```
Device> enable
Device# configure terminal
Device(config)# ip dhcp pool pnp_device_pool
Device(config-dhcp)# vrf Mgmt-vrf
Device(config-dhcp)# network 10.1.1.0 255.255.255.0
Device(config-dhcp)# default-router 10.1.1.1
Device(config-dhcp)# option 67 ascii http://198.51.100.1:8000/sample_python_2.py
Device(config-dhcp)# end
```

Sample DHCP Server Configuration on an In-Band Port Using TFTP Copy

The following is a sample DHCP server configuration using TFTP copy, when connected via the in-band port on a device:

```
Device> enable
Device# configure terminal
Device(config)# ip dhcp excluded-address 10.1.1.1
Device(config)# ip dhcp pool pnp_device_pool
Device(config-dhcp)# network 10.1.1.0 255.255.255.0
Device(config-dhcp)# default-router 10.1.1.1
Device(config-dhcp)# option 150 ip 203.0.113.254
Device(config-dhcp)# option 67 ascii /sample_python_dir/python_script.py
Device(config-dhcp)# exit
Device(config)# interface gigabitethernet 1/0/2
Device(config-if)# no ip dhcp client request tftp-server-address
Device(config-if)# end
```

Sample DHCP Server Configuration on an In-Band Port Using HTTP Copy

The following is a sample DHCP server configuration using HTTP copy, when connected via the in-band port on a device:

```
Device> enable
Device# configure terminal
Device(config)# ip dhcp excluded-address 10.1.1.1
Device(config)# ip dhcp pool pnp_device_pool
Device(config-dhcp)# network 10.1.1.0 255.255.255.0
Device(config-dhcp)# default-router 10.1.1.1
Device(config-dhcp)# option 67 ascii http://192.0.2.1:8000/sample_python_2.py
Device(config-dhcp)# end
```

Sample DHCP Server Configuration on a Linux Ubuntu Device

The following sample DHCP server configuration displays that the server is either connected to the management port or in-band port on a device, and a Python script is copied from a TFTP server.

```
root@ubuntu-server:/etc/dhcp# more dhcpd.conf
```

```
subnet 10.1.1.0 netmask 255.255.255.0 {
range 10.1.1.2  10.1.1.255;
      host 3850 {
          fixed-address                      10.1.1.246 ;
          hardware ethernet                  CC:D8:C1:85:6F:00;
          option bootfile-name !<opt 67>     " /python_dir/python_script.py";
         option tftp-server-name !<opt 150>  "203.0.113.254";
      }
  }
```

The following sample DHCP configuration shows that a Python script is copied from an HTTP server to the device:

```
Day0_with_mgmt_port_http
------------------------
subnet 192.168.1.0 netmask 255.255.255.0 {
 range 192.168.1.2 192.168.1.255;
      host C2-3850 {
          fixed-address                   192.168.1.246 ;
          hardware ethernet               CC:D8:C1:85:6F:00;
          option bootfile-name            "http://192.168.1.46/sample_python_2.py";
      }
  }
```

Once the DHCP server is running, boot a management-network connected device, and the rest of the configuration is automatic.

Sample Python Provisioning Script

The following is a sample Python script can be used from either an HTTP or a TFTP server:

```
print "\n\n *** Sample ZTP Day0 Python Script *** \n\n"

# Importing cli module
import cli


print "\n\n *** Executing show platform  *** \n\n"
cli_command = "show platform"
cli.executep(cli_command)

print "\n\n *** Executing show version *** \n\n"
cli_command = "show version"
cli.executep(cli_command)

print "\n\n *** Configuring a Loopback Interface *** \n\n"
cli.configurep(["interface loop 100", "ip address 10.10.10.10 255.255.255.255", "end"])


print "\n\n *** Executing show ip interface brief  *** \n\n"
cli_command = "sh ip int brief"
cli.executep(cli_command)

print "\n\n *** ZTP Day0 Python Script Execution Complete *** \n\n"
```

Zero-Touch Provisioning Boot Log

The following sample Zero-Touch Provisioning boot log displays that Guest Shell is successfully enabled, the Python script is downloaded to the Guest Shell, and the Guest Shell executes the downloaded Python script and configures the device for Day Zero.

```
% failed to initialize nvram
! <This message indicates that the startup configuration
is absent on the device. This is the first indication that the Day Zero work flow is
going to start.>

This product contains cryptographic features and is subject to United
States and local country laws governing import, export, transfer and
use. Delivery of Cisco cryptographic products does not imply
third-party authority to import, export, distribute or use encryption.
Importers, exporters, distributors and users are responsible for
compliance with U.S. and local country laws. By using this product you
agree to comply with applicable laws and regulations. If you are unable
to comply with U.S. and local laws, return this product immediately.

A summary of U.S. laws governing Cisco cryptographic products may be found at:
http://www.cisco.com/wwl/export/crypto/tool/stqrg.html

If you require further assistance please contact us by sending email to
export@cisco.com.

cisco ISR4451-X/K9 (2RU) processor with 7941237K/6147K bytes of memory.
Processor board ID FJC1950D091
4 Gigabit Ethernet interfaces
32768K bytes of non-volatile configuration memory.
16777216K bytes of physical memory.
7341807K bytes of flash memory at bootflash:.
0K bytes of WebUI ODM Files at webui:.

%INIT: waited 0 seconds for NVRAM to be available


         --- System Configuration Dialog ---

Would you like to enter the initial configuration dialog? [yes/no]: %
!!<DO NOT TOUCH. This is Zero-Touch Provisioning>>
Generating 2048 bit RSA keys, keys will be non-exportable...
[OK] (elapsed time was 1 seconds)
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
Guestshell enabled successfully


 *** Sample ZTP Day0 Python Script ***


 *** Configuring a Loopback Interface ***


Line 1 SUCCESS: interface loop 100
Line 2 SUCCESS: ip address 10.10.10.10 255.255.255.255
Line 3 SUCCESS: end
```

```
 *** Executing show ip interface brief  ***

Interface              IP-Address      OK? Method Status                Protocol
GigabitEthernet0/0/0   unassigned      YES unset  down                  down
GigabitEthernet0/0/1   unassigned      YES unset  down                  down
GigabitEthernet0/0/2   unassigned      YES unset  down                  down
GigabitEthernet0/0/3   192.168.1.246   YES DHCP   up                    up
GigabitEthernet0       192.168.1.246   YES DHCP   up                    up
Loopback100            10.10.10.10     YES TFTP   up                    up

 *** ZTP Day0 Python Script Execution Complete ***


Press RETURN to get started!
```

The Day Zero provisioning is complete, and the IOS prompt is accessible.

Feature Information for Zero-Touch Provisioning

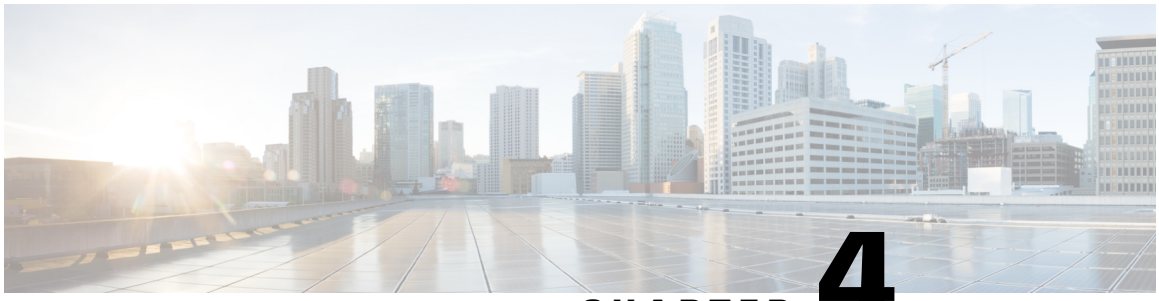
The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train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

Feature Name	Release	Feature Information
Zero-Touch Provisioning	Cisco IOS XE Everest 16.5.1a Cisco IOS XE Everest 16.5.1b	To address network provisioning challenges, Cisco introduces a zero-touch provisioning model. In Cisco IOS XE Everest 16.5.1a, this feature was implemented on the following platforms: • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches In Cisco IOS XE Everest 16.5.1b, this feature was implemented on the following platform: • Cisco 4000 Series Integrated Services Routers

Feature Name	Release	Feature Information
Zero-Touch Provisioning: HTTP Download	Cisco IOS XE Everest 16.6.1	Zero-Touch Provisioning supports HTTP and TFTP file download. In Cisco IOS XE Everest 16.6.1, this feature was implemented on the following platforms: • Cisco 4000 Series Integrated Services Routers • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches
	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.



CHAPTER 4

CLI Python Module

Python Programmability provides a Python module that allows users to interact with IOS using CLIs.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table at the end of this module.

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

Information About CLI Python Module

About Python

The Cisco IOS XE devices support Python Version 2.7 in both interactive and non-interactive (script) modes within the Guest Shell. The Python scripting capability gives programmatic access to a device's CLI to perform various tasks and Zero Touch Provisioning or Embedded Event Manager (EEM) actions.

Python Scripts Overview

Python run in a virtualized Linux-based environment, Guest Shell. For more information, see the *Guest Shell* chapter. Cisco provides a Python module that allows user's Python scripts to run IOS CLI commands on the host device.

Interactive Python Prompt

When you execute the **guestshell run python** command on a device, the interactive Python prompt is opened inside the Guest Shell. The Python interactive mode allows users to execute Python functions from the Cisco Python CLI module to configure the device.

The following example shows how to enable the interactive Python prompt:

```
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.
>>>

Device#
```

Python Script

Python scripts can run in non-interactive mode by providing the Python script name as an argument in the Python command. Python scripts must be accessible from within the Guest Shell. To access Python scripts from the Guest Shell, save the scripts in bootflash/flash that is mounted within the Guest Shell.

The following sample Python script uses different CLI functions to configure and print **show** commands:

```
Device# more flash:sample_script.py


import sys
import cli

intf= sys.argv[1:]
intf = ''.join(intf[0])

print "\n\n *** Configuring interface %s with 'configurep' function  *** \n\n" %intf
cli.configurep(["interface loopback55","ip address 10.55.55.55 255.255.255.0","no
shut","end"])

print "\n\n *** Configuring interface %s with 'configure' function  *** \n\n"
cmd='interface %s,logging event link-status ,end' % intf
cli.configure(cmd.split(','))

print "\n\n *** Printing show cmd with 'executep' function  *** \n\n"
cli.executep('show ip interface brief')

print "\n\n *** Printing show cmd with 'execute' function  *** \n\n"
output= cli.execute('show run interface %s' %intf)
print (output)

print "\n\n *** Configuring interface %s with 'cli' function  *** \n\n"
cli.cli('config terminal; interface %s; spanning-tree portfast edge default' %intf)

print "\n\n *** Printing show cmd with 'clip' function  *** \n\n"
cli.clip('show run interface %s' %intf)
```

```
To run a Python script from the Guest Shell, execute the guestshell run python
/flash/script.py command
at the device prompt.
The following example shows how to run a Python script from the Guest Shell:
```

The following example shows how to run a Python script from the Guest Shell:

```
Device# guestshell run python /flash/sample_script.py loop55

 *** Configuring interface loop55 with 'configurep' function  ***

Line 1 SUCCESS: interface loopback55
Line 2 SUCCESS: ip address 10.55.55.55 255.255.255.0
Line 3 SUCCESS: no shut
Line 4 SUCCESS: end

 *** Configuring interface %s with 'configure' function  ***


 *** Printing show cmd with 'executep' function  ***

Interface              IP-Address      OK? Method Status                Protocol
Vlan1                  unassigned      YES NVRAM  administratively down down
GigabitEthernet0/0     192.0.2.1       YES NVRAM  up                    up
GigabitEthernet1/0/1   unassigned      YES unset  down                  down
GigabitEthernet1/0/2   unassigned      YES unset  down                  down
GigabitEthernet1/0/3   unassigned      YES unset  down                  down
    :
          :
          :
Te1/1/4                unassigned      YES unset  down                  down
Loopback55             10.55.55.55     YES TFTP   up                    up
Loopback66             unassigned      YES manual up                    up


 *** Printing show cmd with 'execute' function  ***

Building configuration...
Current configuration : 93 bytes
!
interface Loopback55
 ip address 10.55.55.55 255.255.255.0
 logging event link-status
end

 *** Configuring interface %s with 'cli' function  ***

 *** Printing show cmd with 'clip' function  ***

Building configuration...
Current configuration : 93 bytes
!
interface Loopback55
 ip address 10.55.55.55 255.255.255.0
 logging event link-status
end
```

Supported Python Versions

Guest Shell is pre-installed with Python Version 2.7. Guest Shell is a virtualized Linux-based environment, designed to run custom Linux applications, including Python applications for automated control and management of Cisco devices. Platforms with Montavista CGE7 support Python Version 2.7.11, and platforms with CentOS 7 support Python Version 2.7.5.

The following table provides information about Python versions and the supported platforms:

Table 5: Python Version Support

Python Version	Platform

Platforms with CentOS 7 support the installation of Redhat Package Manager (RPM) from the open source repository.

Updating the Cisco CLI Python Module

The Cisco CLI Python module and EEM module are pre-installed on devices. However, when you update the Python version by using either Yum or prepackaged binaries, the Cisco-provided CLI module must also be updated.

Note When you update to Python Version 3 on a device that already has Python Version 2, both versions of Python exist on the device. Use one of the following IOS commands to run Python:

- The **guestshell run python2** command enables Python Version 2.
- The **guestshell run python3** command enables Python Version 3.
- The **guestshell run python** command enables Python Version 2.

Use one of the following methods to update the Python version:

- Standalone tarball installation
- PIP install for the CLI module

Additional References for the CLI Python Module

Related Documents

Related Topic	Document Title
Guest Shell	
EEM Python Module	

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

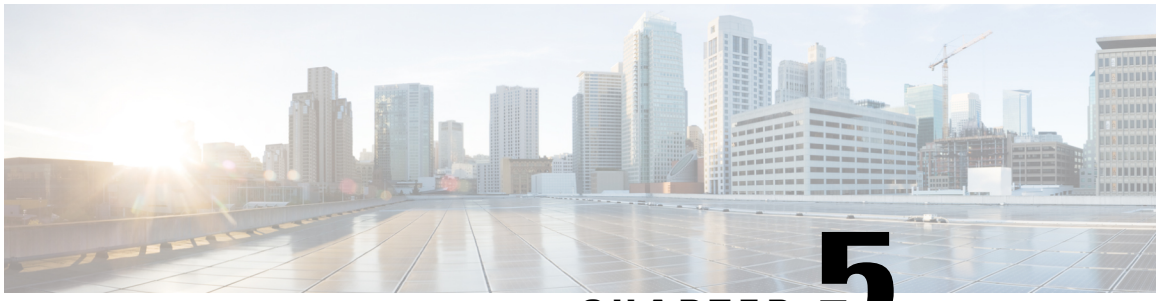
Feature Information for the CLI Python Module

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 6: Feature Information for the CLI Python Module

Feature Name	Release	Feature Information
CLI Python Module		Python programmabilty provides a Python module that allows users to interact with IOS using CLIs.



CHAPTER 5

EEM Python Module

Embedded Event Manager (EEM) policies support Python scripts. Python scripts can be executed as part of EEM actions in EEM applets.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table at the end of this module.

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

Prerequisites for the EEM Python Module

Guest Shell must be working within the container. Guest Shell is not enabled by default. For more information see the *Guest Shell* feature.

Information About the EEM Python Module

Python Scripting in EEM

Embedded Event Manager (EEM) policies support Python scripts. You can register Python scripts as EEM policies, and execute the registered Python scripts when a corresponding event occurs. The EEM Python script has the same event specification syntax as the EEM TCL policy.

Configured EEM policies run within the Guest Shell. Guest Shell is a virtualized Linux-based environment, designed to run custom Linux applications, including Python for automated control and management of Cisco devices. The Guest Shell container provides a Python interpreter.

EEM Python Package

The EEM Python package can be imported to Python scripts for running EEM-specific extensions.

Note The EEM Python package is available only within the EEM Python script (The package can be registered with EEM, and has the EEM event specification in the first line of the script.) and not in the standard Python script (which is run using the Python script name).

The Python package includes the following application programming interfaces (APIs):

- Action APIs—Perform EEM actions and have default parameters.
- CLI-execution APIs—Run IOS commands, and return the output. The following are the list of CLI-execution APIs:
  - eem_cli_open()
  - eem_cli_exec()
  - eem_cli_read()
  - eem_cli_read_line()
  - eem_cli_run()
  - eem_cli_run_interactive()
  - eem_cli_read_pattern()
  - eem_cli_write()
  - eem_cli_close()
- Environment variables-accessing APIs—Get the list of built-in or user-defined variables. The following are the environment variables-accessing APIs:
  - eem_event_reqinfo ()-Returns the built-in variables list.
  - eem_user_variables()-Returns the current value of an argument.

Python-Supported EEM Actions

The Python package (is available only within the EEM script, and not available for the standard Python script) supports the following EEM actions:

- Syslog message printing
- Send SNMP traps
- Reload the box
- Switchover to the standby device
- Run a policy
- Track Object read
- Track Object Set
- Cisco Networking Services event generation

The EEM Python package exposes the interfaces for executing EEM actions. You can use the Python script to call these actions, and they are forwarded from the Python package via Cisco Plug N Play (PnP) to the action handler.

EEM Variables

An EEM policy can have the following types of variables:

- Event-specific built-in variables—A set of predefinied variables that are populated with details about the event that triggered the policy. The eem_event_reqinfo () API returns the builtin variables list. These variables can be stored in the local machine and used as local variables. Changes to local variables do not reflect in builtin variables.
- User-defined variables—Variables that can be defined and used in policies. The value of these variables can be referred in the Python script. While executing the script, ensure that the latest value of the variable is available. The eem_user_variables() API returns the current value of the argument that is provided in the API.

EEM CLI Library Command Extensions

The following CLI library commands are available within EEM for the Python script to work:

- eem_cli_close()—Closes the EXEC process and releases the VTY and the specified channel handler connected to the command.
- eem_cli_exec—Writes the command to the specified channel handler to execute the command. Then reads the output of the command from the channel and returns the output.
- eem_cli_open—Allocates a VTY, creates an EXEC CLI session, and connects the VTY to a channel handler. Returns an array including the channel handler.
- eem_cli_read()—Reads the command output from the specified CLI channel handler until the pattern of the device prompt occurs in the contents read. Returns all the contents read up to the match.

- eem_cli_read_line()—Reads one line of the command output from the specified CLI channel handler. Returns the line read.
- eem_cli_read_pattern()—Reads the command output from the specified CLI channel handler until the pattern that is to be matched occurs in the contents read. Returns all the contents read up to the match.
- eem_cli_run()—Iterates over the items in the *clist* and assumes that each one is a command to be executed in the enable mode. On success, returns the output of all executed commands and on failure, returns error.
- eem_cli_run_interactive()—Provides a sublist to the *clist* which has three items. On success, returns the output of all executed commands and on failure, returns the error. Also uses arrays when possible as a way of making things easier to read later by keeping expect and reply separated.
- eem_cli_write()—Writes the command that is to be executed to the specified CLI channel handler. The CLI channel handler executes the command.

How to Configure the EEM Python Policy

For the Python script to work, you must enable the Guest Shell. For more information, see the *Guest Shell* chapter.

Registering a Python Policy

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **event manager directory user policy** *path*
4. **event manager policy** *policy-filename*
5. **exit**
6. **show event manager policy registered**
7. **show event manager history events**

DETAILED STEPS

	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.

	Command or Action	Purpose
Step 3	**event manager directory user policy** *path* **Example:** `Device(config)# event manager directory user policy flash:/user_library`	Specifies a directory to use for storing user library files or user-defined EEM policies. **Note** You must have a policy in the specified path. For example, in this step, the eem_script.py policy is available in the flash:/user_library folder or path.
Step 4	**event manager policy** *policy-filename* **Example:** `Device(config)# event manager policy eem_script.py`	Registers a policy with EEM. • The policy is parsed based on the file extension. If the file extension is .py, the policy is registered as Python policy. • EEM schedules and runs policies on the basis of an event specification that is contained within the policy itself. When the **event manager policy** command is invoked, EEM examines the policy and registers it to be run when the specified event occurs.
Step 5	**exit** **Example:** `Device(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.
Step 6	**show event manager policy registered** **Example:** `Device# show event manager policy registered`	Displays the registered EEM policies.
Step 7	**show event manager history events** **Example:** `Device# show event manager history events`	Displays EEM events that have been triggered.

The following is sample output from the **show event manager policy registered** command:

```
Device# show event manager policy registered

No.  Class     Type    Event Type          Trap  Time Registered           Name
1    script    user    multiple            Off   Tue Aug 2 22:12:15 2016   multi_1.py
 1: syslog: pattern {COUNTER}
 2: none: policyname {multi_1.py} sync {yes}
 trigger delay 10.000
  correlate event 1 or event 2
  attribute tag 1 occurs 1
 nice 0 queue-priority normal maxrun 100.000 scheduler rp_primary Secu none


2    script    user    multiple            Off   Tue Aug 2 22:12:20 2016   multi_2.py
 1: syslog: pattern {COUNTER}
 2: none: policyname {multi_2.py} sync {yes}
 trigger
  correlate event 1 or event 2
 nice 0 queue-priority normal maxrun 100.000 scheduler rp_primary Secu none


3    script    user    multiple            Off   Tue Aug 2 22:13:31 2016   multi.tcl
```

```
 1: syslog: pattern {COUNTER}
 2: none: policyname {multi.tcl} sync {yes}
 trigger
  correlate event 1 or event 2
  attribute tag 1 occurs 1
 nice 0 queue-priority normal maxrun 100.000 scheduler rp_primary Secu none
```

Running Python Scripts as Part of EEM Applet Actions

Python Script: eem_script.py

An EEM applet can include a Python script with an action command. In this example, an user is trying to run a standard Python script as part of the EEM action, however; EEM Python package is not available in the standard Python script. The standard Python script in IOS has a package named *from cli import cli,clip* and this package can be used to execute IOS commands.

```
import sys
from cli import cli,clip,execute,executep,configure,configurep

intf= sys.argv[1:]
intf = ''.join(intf[0])


print ('This script is going to unshut interface %s and then print show ip interface
brief'%intf)

if intf == 'loopback55':
configurep(["interface loopback55","no shutdown","end"])
else :
cmd='int %s,no shut ,end' % intf
configurep(cmd.split(','))

executep('show ip interface brief')
```

This following is sample output from the **guestshell run python** command.

```
Device#  guestshell run python /flash/eem_script.py loop55

This script is going to unshut interface loop55 and then print show ip interface brief
Line 1 SUCCESS: int loop55
Line 2 SUCCESS: no shut
Line 3 SUCCESS: end
Interface IP-Address OK? Method Status Protocol
Vlan1 unassigned YES NVRAM administratively down down
GigabitEthernet0/0 5.30.15.37 YES NVRAM up up
GigabitEthernet1/0/1 unassigned YES unset down down
GigabitEthernet1/0/2 unassigned YES unset down down
GigabitEthernet1/0/3 unassigned YES unset down down
GigabitEthernet1/0/4 unassigned YES unset up up
GigabitEthernet1/0/5 unassigned YES unset down down
GigabitEthernet1/0/6 unassigned YES unset down down
GigabitEthernet1/0/7 unassigned YES unset down down
GigabitEthernet1/0/8 unassigned YES unset down down
GigabitEthernet1/0/9 unassigned YES unset down down
GigabitEthernet1/0/10 unassigned YES unset down down
GigabitEthernet1/0/11 unassigned YES unset down down
GigabitEthernet1/0/12 unassigned YES unset down down
GigabitEthernet1/0/13 unassigned YES unset down down
GigabitEthernet1/0/14 unassigned YES unset down down
GigabitEthernet1/0/15 unassigned YES unset down down
GigabitEthernet1/0/16 unassigned YES unset down down
GigabitEthernet1/0/17 unassigned YES unset down down
GigabitEthernet1/0/18 unassigned YES unset down down
GigabitEthernet1/0/19 unassigned YES unset down down
GigabitEthernet1/0/20 unassigned YES unset down down
GigabitEthernet1/0/21 unassigned YES unset down down
```

```
GigabitEthernet1/0/22 unassigned YES unset down down
GigabitEthernet1/0/23 unassigned YES unset up up
GigabitEthernet1/0/24 unassigned YES unset down down
GigabitEthernet1/1/1 unassigned YES unset down down
GigabitEthernet1/1/2 unassigned YES unset down down
GigabitEthernet1/1/3 unassigned YES unset down down
GigabitEthernet1/1/4 unassigned YES unset down down
Te1/1/1 unassigned YES unset down down
Te1/1/2 unassigned YES unset down down
Te1/1/3 unassigned YES unset down down
Te1/1/4 unassigned YES unset down down
Loopback55 10.55.55.55 YES manual up up

Device#
Jun 7 12:51:20.549: %LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback55,
changed state to up
Jun 7 12:51:20.549: %LINK-3-UPDOWN: Interface Loopback55, changed state to up
```

The following is a sample script for printing messages to the syslog. This script must be stored in a file, copied to the file system on the device, and registered using the event manager policy file.

```
::cisco::eem::event_register_syslog tag "1"    pattern     COUNTER    maxrun 200

import eem
import time

eem.action_syslog("SAMPLE SYSLOG MESSAGE","6","TEST")
```

The following is sample script to print EEM environment variables. This script must be stored in a file, copied to the file system on the device, and registered using the event manager policy file.

```
::cisco::eem::event_register_syslog tag "1"    pattern     COUNTER    maxrun 200

import eem
import time

c = eem.env_reqinfo()


print "EEM Environment Variables"
for k,v in c.iteritems():
      print "KEY : " + k + str(" ---> ") + v

print "Built in Variables"
for i,j in a.iteritems() :
      print "KEY : " + i + str(" ---> ") + j
```

Adding a Python Script in an EEM Applet

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **event manager applet** *applet-name*
4. **event** [**tag** *event-tag*] **syslog pattern** *regular-expression*
5. **action** *label* **cli command** *cli-string*
6. **action** *label* **cli command** *cli-string* [**pattern** *pattern-string*]
7. **end**
8. **show event manager policy active**
9. **show event manager history events**

DETAILED STEPS

	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	**event manager applet** *applet-name* **Example:** `Device(config)# event manager applet interface_Shutdown`	Registers an applet with the Embedded Event Manager (EEM) and enters applet configuration mode.
Step 4	**event** [**tag** *event-tag*] **syslog pattern** *regular-expression* **Example:** `Device(config-applet)# event syslog pattern "Interface Loopback55, changed state to administratively down"`	Specifies a regular expression to perform the syslog message pattern match.
Step 5	**action** *label* **cli command** *cli-string* **Example:** `Device(config-applet)# action 0.0 cli command "en"`	Specifies the IOS command to be executed when an EEM applet is triggered.
Step 6	**action** *label* **cli command** *cli-string* [**pattern** *pattern-string*]	Specifies the action to be specified with the **pattern** keyword.

	Command or Action	Purpose
	Example: `Device(config-applet)# action 1.0 cli command "guestshell run python3 /bootflash/eem_script.py loop55"`	• Specify a regular expression pattern string that will match the next solicited prompt.
Step 7	**end** **Example:** `Device(config-applet)# end`	Exits applet configuration mode and returns to privileged EXEC mode.
Step 8	**show event manager policy active** **Example:** `Device# show event manager policy active`	Displays EEM policies that are executing.
Step 9	**show event manager history events** **Example:** `Device# show event manager history events`	Displays the EEM events that have been triggered.

What to Do Next

The following example shows how to trigger the Python script configured in the task:

```
Device(config)# interface loopback 55
Device(config-if)# shutdown
Device(config-if)# end
Device#

Mar 13 10:53:22.358 EDT: %SYS-5-CONFIG_I: Configured from console by console
Mar 13 10:53:24.156 EDT: %LINK-5-CHANGED:  Line protocol on Interface Loopback55, changed
state to down
Mar 13 10:53:27.319 EDT: %LINK-3-UPDOWN:  Interface Loopback55, changed state to
administratively down
Enter configuration commands, one per line.  End with CNTL/Z.
Mar 13 10:53:35.38 EDT:  %LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback55, changed
 state to up
*Mar 13 10:53:35.39 EDT %LINK-3-UPDOWN: Interface Loopback55, changed state to up
+++ 10:54:33 edi37(default) exec +++
show ip interface br
Interface              IP-Address      OK? Method Status                Protocol
GigabitEthernet0/0/0   unassigned      YES unset  down                  down
GigabitEthernet0/0/1   unassigned      YES unset  down                  down
GigabitEthernet0/0/2   10.1.1.31       YES DHCP   up                    up
GigabitEthernet0/0/3   unassigned      YES unset  down                  down
GigabitEthernet0       192.0.2.1       YES manual up                    up
Loopback55             198.51.100.1    YES manual up                    up
Loopback66             172.16.0.1      YES manual up                    up
Loopback77             192.168.0.1     YES manual up                    up
Loopback88             203.0.113.1     YES manual up                    up
```

Additional References EEM Python Module

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Command List, All Releases
EEM configuration	Embedded Event Manager Configuration Guide
EEM commands	Embedded Event Manager Command Reference
Guest Shell configuration	

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 EEM Python Module

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 7: Feature Information for EEM Python Module

Feature Name	Release	Feature Information
EEM Python Module	Cisco IOS XE Everest 16.5.1b	This feature supports Python scripts as EEM policies. No new commands were introduced.