Automating Cisco IOS XE Command Sequences Using TCL and EEM

Contents

Introduction

Prerequisites

- Requirements
- Components Used

Background Information

Configuration andStep-by-Step Procedure

- Step 1. Creation of Text File with Sequence of Commands
  - Creating a Text File Using TCL Scripting
- Step 2. Execute the File Using EEM Script
  - Creation of EEM Script
  - Explanation of the EEM Script
  - EEM Script Execution
  - Result of File Execution

Summary

Introduction

This document describes a method to create a unique and single EEM script capable of executing any sequence of commands as defined by a network administrator.

Prerequisites

Requirements

Cisco recommends that you have knowledge of these topics:

- Knowledge of Cisco IOS® XE operations
- Understanding of Embedded Event Manager (EEM) scripting
- Familiarity with Cisco Catalyst Switches
- Basic knowledge of TCL scripting

Components Used

The information in this document is based on Cisco Catalyst 9600X Series Switch (CS-C9600X) running

IOS XE 17.18.2 release.

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

Background Information

The administrator creates a plain text file containing the desired command sequence. The EEM script then calls this file upon a specified trigger and executes all commands listed in the text file.

Configuration and Step-by-Step Procedure

Step 1. Creation of Text File with Sequence of Commands

1. Create a text file containing the required commands in the desired sequence.
2. Copy the file to the flash memory of the device.
3. Use FTP, TFTP, or TCL scripting to create or transfer the file.

Creating a Text File Using TCL Scripting

Syntax:

<#root>

```
tclsh
puts [open "flash:COMMANDS_TO_EXECUTE.txt" w+] {


Enter Commands to Execute



}
exit
```

Configuration Example:

In order to create a sequence of commands that shuts down interface FiftyGigE1/1/0/3:

```
C9600-Silicon1-SVL#tclsh
```

```
C9600-Silicon1-SVL(tcl)#puts [open "flash:COMMANDS_TO_EXECUTE.txt" w+] {
configure terminal
interface fiftyGigE 1/1/0/3
shutdown
end
show run interface fiftyGigE 1/1/0/3
}
C9600-Silicon1-SVL(tcl)#exit
C9600-Silicon1-SVL#
```

Verify the file in flash:

```
C9600-Silicon1-SVL#dir flash:
Directory of flash:/

357117 -rw- 101 Jan 25 2026 21:37:09 +05:30 COMMANDS_TO_EXECUTE.txt
```

Contents of **COMMANDS_TO_EXECUTE.txt**

```
configure terminal
interface fiftyGigE 1/1/0/3
shutdown
end
show run interface fiftyGigE 1/1/0/3
```

This file can be created either by TCL scripting or transferred using FTP, TFTP, or RCP. The file must be edited as needed in order to include any sequence of commands required by the network administrator.

Step 2. Execute the File Using EEM Script

Creation of EEM Script

This EEM applet executes all commands from the specified file **COMMANDS_TO_EXECUTE**.

```
event manager applet Cisco_Live authorization bypass
event none maxrun 1000
action 010 cli command "enable"
action 020 cli command "term exec prompt timestamp"
action 030 cli command "term exec prompt expand"
action 040 file open fileout1 flash:OUTPUT.txt a+
action 050 file open filein1 flash:COMMANDS_TO_EXECUTE.txt r
action 060 file read filein1 input
action 070 foreach line $input "\n"
action 080 cli command "$line"
action 090 file write fileout1 "$_cli_result"
action 100 end
```

```
action 110 file close fileout1
action 120 file close filein1
```

Explanation of the EEM Script

++ event manager applet Cisco_Live authorization bypass:
Event manager applet applet-name: Registers an EEM applet with the specified name ('Cisco_Live' in this case).
Authorization bypass: Optional keywords that specify the applet must bypass AAA authorization.

++ event none maxrun 1000:
Event none: This specifies that the EEM policy is registered with the 'none' event detector, meaning the policy is not triggered by any automatic event but is intended to be run manually.
Maxrun 1000: This optional parameter sets the maximum runtime for the applet or policy to 1000 seconds. If not specified, the default maximum runtime is 20 seconds.

++ action 020 cli command **term exec prompt timestamp**:
Action 020: This is the label or sequence identifier for the action within the EEM script.
term exec prompt timestamp is a CLI command that enables timestamps on the prompt in the current terminal session.
This feature helps in troubleshooting and auditing by providing a time reference for each command executed.
The timestamp appears before the output of commands, making it easier to correlate events and command executions with specific times.

++ action 030 cli command **term exec prompt expand**:
term exec prompt expand enables the expanded prompt mode for the current terminal session.
And it configures the terminal to display the full prompt without truncation or abbreviation.
It helps in scripts or troubleshooting scenarios where the full prompt context is needed for clarity or parsing.
The command affects only the current terminal session and does not persist after the session ends.

++ action 040 file open fileout1 flash:OUTPUT.txt a+:
File open: This specifies the operation to open a file.
Fileout1: This is the file descriptor or handle name used within the script to refer to the opened file.
Flash:OUTPUT.txt: This is the path and filename in the flash file system where the file resides or will be created.
a+: This is the file access mode, meaning the file is opened for reading and appending. If the file does not exist, it will be created. Data written to the file will be appended to the end without truncating existing content.

++ action 050 file open filein1 flash:COMMANDS_TO_EXECUTE.txt r:
Filein1: This is the file descriptor or handle name used within the script to refer to the opened file.
Flash:COMMANDS_TO_EXECUTE.txt: This is the path and filename in the flash file system where the

file resides.
r: This is the file access mode, meaning the file is opened for reading only.

++ action 060 file read filein1 input:
File read: This specifies the operation to read from a file.
Filein1: This is the file descriptor or handle that was assigned when the file was opened (for example, with action 050 file open filein1. Flash:COMMANDS_TO_EXECUTE.txt r).
Input: This is the variable name where the read data will be stored for use in subsequent script actions.

++ action 070 foreach line $input '\n':
Foreach: This command specifies an iteration over a string.
Line: This is the iterator variable that will hold each token (line) during each iteration.
$input: This is the input string variable containing multiple lines or entries.
'\n': This is the delimiter used to split the input string into tokens, in this case, the newline character.
Each line is a token and \n is the splitter in this case.

++ action 080 cli command **$line**:
Cli command: Specifies that the action is to execute a CLI command on the device.
$line: The variable containing the CLI command string to be executed. The quotes ensure the entire string is treated as a single command.

++ action 090 file write fileout1 '$_cli_result':
File write: The EEM command to write data to a file.
Fileout1: The file handler previously opened by a command like file open fileout1 flash:OUTPUT.txt a+, representing the file to which data is written.
'$_cli_result': The variable holding the CLI command output or any string data to be written to the file. The quotes ensure the entire content is treated as a single string.
This command is typically used after executing CLI commands and capturing their output in $_cli_result, enabling the script to log or archive the output to a file on the flash storage of the device.

++ action 100 end:
End: The keyword that signifies the end of a conditional block such as if, else, or while.

++ action 120 file close filein1:
File close: The operation to close the file descriptor.
Filein1: The identifier of the file descriptor to be closed, which must have been opened earlier in the script.

EEM Script Execution

In order to manually trigger the EEM script:

```
C9600-Silicon1-SVL#event manager run Cisco_Live
```

In this example the EEM script is manually triggered. The script can have any trigger as per the network requirements.

Result of File Execution

Viewing the output file shows the commands executed and their results:

<#root>

```
C9600-Silicon1-SVL#more flash:OUTPUT.txt

Enter configuration commands, one per line. End with CNTL/Z.
C9600-Silicon1-SVL(config)#
C9600-Silicon1-SVL(config-if)#
C9600-Silicon1-SVL(config-if)#
C9600-Silicon1-SVL#Load for five secs: 0%/0%; one minute: 1%; five minutes: 1%
No time source, *22:01:35.406 IST Sun Jan 25 2026
------------------ show running-config interface FiftyGigE1/1/0/3 ------------------

Building configuration...

Current configuration : 127 bytes
!
interface FiftyGigE1/1/0/3
description To-Host3-Ten 1/1/3
switchport access vlan 10
switchport mode access

shutdown


end
```

The interface FiftyGigE1/1/0/3 is successfully shut down as per the command sequence.

Summary

This document demonstrated how to combine TCL scripting and the EEM of Cisco in order to create a flexible, single EEM script capable of executing any sequence of commands stored in a text file. The approach allows network administrators to define command sequences externally and trigger their execution automatically or manually, enhancing automation and operational efficiency on Cisco Catalyst switches running Cisco IOS XE.