The Cisco IOS Scripting with Tcl feature provides the ability to run Tool Command Language (Tcl) version 8.3.4 commands from the Cisco IOS command-line interface (CLI).

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see 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 for Cisco IOS Scripting with Tcl” section on page 21.

Use Cisco Feature Navigator to find information about platform support and Cisco IOS and Catalyst OS 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 Cisco IOS Scripting with Tcl

- Familiarity with Tcl programming and Cisco IOS commands is required.
- Tcl commands can be executed from the Tcl configuration mode using the Cisco IOS CLI. Tcl configuration mode is accessed from privileged EXEC mode. Access to privileged EXEC mode should be managed by restricting access using the `enable` command password.

Restrictions for Cisco IOS Scripting with Tcl

- If Cisco IOS configuration commands are used within the Tcl scripts, submode commands must be entered as quoted arguments on the same line as the configuration command.
- Error messages are provided, but you must check that the Tcl script will run successfully because errors may cause the Tcl shell to run in an infinite loop.

Caution

The use of Tcl server sockets to listen to telnet and FTP ports (23 and 21 respectively) will preempt the normal handling of these ports in Cisco IOS software.

- Table 1 lists Tcl commands and library calls that do not behave within Cisco IOS software as documented in standard Tcl documents.

<table>
<thead>
<tr>
<th>Command</th>
<th>Keyword</th>
<th>Argument</th>
<th>Supported</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>after</td>
<td>ms</td>
<td>script</td>
<td>Partially</td>
<td>When the CLI <code>tclsh</code> command is used, there is no event loop implemented unless Embedded Syslog Manager (ESM) is active on the same router. Commands entered using the <code>after</code> Tcl command will not run unless forced using the <code>update</code> command. Sleep mode (the <code>after</code> command) works only with the <code>ms</code> keyword.</td>
</tr>
<tr>
<td>file</td>
<td>-time</td>
<td>atime</td>
<td>No</td>
<td>The optional <code>-time</code> keyword to set the file access time is not supported in Cisco IOS software.</td>
</tr>
<tr>
<td>file</td>
<td>-time</td>
<td>mtime</td>
<td>No</td>
<td>The optional <code>-time</code> keyword to set the file modification time is not supported in Cisco IOS software.</td>
</tr>
<tr>
<td>fileevent</td>
<td></td>
<td></td>
<td>Partially</td>
<td>When the CLI <code>tclsh</code> command is used, there is no event loop implemented unless Embedded Syslog Manager (ESM) is active on the same router. Commands entered using the <code>fileevent</code> Tcl command will not run unless forced using the <code>update</code> command.</td>
</tr>
</tbody>
</table>
Information About Cisco IOS Scripting with Tcl

To create and use Tcl scripts within Cisco IOS software, you should understand the following concepts:

- **Tcl Shell for Cisco IOS Software**, page 3
- **Tcl Precompiler**, page 3
- **SNMP MIB Object Access**, page 4

### Tcl Shell for Cisco IOS Software

The Cisco IOS Tcl shell was designed to allow customers to run Tcl commands directly from the Cisco IOS CLI prompt. Cisco IOS software does contain some subsystems such as Embedded Syslog Manager (ESM) and Interactive Voice Response (IVR) that use Tcl interpreters as part of their implementation. These subsystems have their own proprietary commands and keyword options that are not available in the Tcl shell.

Several methods have been developed for creating and running Tcl scripts within Cisco IOS software. A Tcl shell can be enabled, and Tcl commands can be entered line by line. After Tcl commands are entered, they are sent to a Tcl interpreter. If the commands are recognized as valid Tcl commands, the commands are executed and the results are sent to the TTY device. If a command is not a recognized Tcl command, it is sent to the Cisco IOS CLI parser. If the command is not a Tcl or Cisco IOS command, two error messages are displayed. A predefined Tcl script can be created outside of Cisco IOS software, transferred to flash or disk memory, and run within Cisco IOS software. It is also possible to create a Tcl script and precompile the code before running it under Cisco IOS software.

Multiple users on the same router can be in Tcl configuration mode at the same time without interference because each Tcl shell session launches a separate interpreter and Tcl server process. The TTY interface number served by each Tcl process is represented in the server process name and can be displayed using the `show process` CLI command.

The Tcl shell can be used to run Cisco IOS CLI EXEC commands within a Tcl script. Using the Tcl shell to run CLI commands allows customers to build menus to guide novice users through tasks, to automate repetitive tasks, and to create custom output for `show` commands.

### Tcl Precompiler

The Cisco IOS Tcl implementation offers support for loading scripts that have been precompiled by the TclPro precompiler. Precompiled scripts allow a measure of security and consistency because they are obfuscated.
SNMP MIB Object Access

Designed to make access to Simple Network Management Protocol (SNMP) MIB objects easier, a set of UNIX-like SNMP commands has been created. The Tcl shell is enabled either manually or by using a Tcl script, and the new commands can be entered to allow you to perform specified get and set actions on MIB objects. To increase usability, the new commands have names similar to those used for UNIX SNMP access. To access the SNMP commands go to, “Using the Tcl Shell to Access SNMP MIB Objects” section on page 10.

Custom Extensions in the Tcl Shell

The Cisco IOS implementation of the Tcl shell contains some custom command extensions. These extensions operate only under Tcl configuration mode. Table 2 displays these command extensions.

Table 2 Cisco IOS Custom Tcl Command Extensions

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fconfigure</td>
<td>Specifies the options in a channel and enables you to associate a virtual routing and forwarding (VRF) table name with it.</td>
</tr>
<tr>
<td>fconfigure -remote [host port] -broadcast boolean vrf [vrf_table_name]</td>
<td></td>
</tr>
<tr>
<td>ios_config</td>
<td>Runs a Cisco IOS CLI configuration command.</td>
</tr>
<tr>
<td>log_user</td>
<td>Toggles Tcl command output under Tcl configuration mode.</td>
</tr>
<tr>
<td>socket -myvrf</td>
<td>Opens a TCP network connection and enables you to associate a VRF table name with it.</td>
</tr>
<tr>
<td>[vrf_table_name]</td>
<td></td>
</tr>
<tr>
<td>typeahead</td>
<td>Writes text to the router standard input (stdin) buffer file.</td>
</tr>
<tr>
<td>tclquit</td>
<td>Leaves Tcl shell—synonym for exit.</td>
</tr>
<tr>
<td>udp_open -ipv6 port</td>
<td>Opens a User Datagram Protocol (UDP) socket.</td>
</tr>
<tr>
<td>udp_peek sock</td>
<td>Enables peeking into a UDP socket.</td>
</tr>
<tr>
<td>-buffersize buffer-size</td>
<td></td>
</tr>
</tbody>
</table>

SNMP MIB Custom Extensions in the Tcl Shell

The Cisco IOS implementation of the Tcl shell contains some custom command extensions for SNMP MIB object access. These extensions operate only under Tcl configuration mode. Table 3 displays these command extensions.
### Table 3 Cisco IOS Custom Tcl Command Extensions for SNMP MIB Access

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
</table>
| **snmp_getbulk** | Retrieves a large section of a MIB table. This command is similar to the SNMP `getbulk` command. The syntax is in the following format:  

```
snmp_getbulk community-string non-repeaters max-repetitions oid [oid2 oid3…]
```

- Use the `community-string` argument to specify the SNMP community from which the objects will be retrieved.
- Use the `non-repeaters` argument to specify the number of objects that can be retrieved with a get-next operation.
- Use the `max-repetitions` argument to specify the maximum number of get-next operations to attempt while trying to retrieve the remaining objects.
- Use the `oid` argument to specify the object ID(s) to retrieve. |
| **snmp_getid** | Retrieves the following variables from the SNMP entity on the router:

- sysDescr.0
- sysObjectID.0
- sysUpTime.0
- sysContact.0
- sysName.0
- sysLocation.0

This command is similar to the SNMP `getid` command. The syntax is in the following format:

```
snmp_getid community-string
```
| **snmp_getnext** | Retrieves a set of individual variables from the SNMP entity on the router. This command is similar to the SNMP `getnext` command. The syntax is in the following format:

```
snmp_getnext community-string oid [oid2 oid3…]
``` |
How to Configure Cisco IOS Scripting with Tcl

This section contains the following tasks:

- Enabling the Tcl Shell and Using the CLI to Enter Commands, page 7 (required)
- Using the Tcl Shell to Access SNMP MIB Objects, page 10 (optional)
- Running Predefined Tcl Scripts, page 13 (optional)
Enabling the Tcl Shell and Using the CLI to Enter Commands

Perform this task to enable the interactive Tcl shell and to enter Tcl commands line by line through the Cisco IOS CLI prompt. Optional steps include specifying a default location for encoding files and specifying an initialization script.

SUMMARY STEPS

1. enable
2. configure terminal
3. scripting tcl encdir location-url
4. scripting tcl init init-url
5. scripting tcl low-memory bytes
6. exit
7. tclsh
8. Enter the required Tcl command language syntax.
9. ios_config “cmd” “cmd-option”
10. socket -myaddr addr -myport port -myvrf vrf-table-name host port
11. socket -server -myaddr addr -myvrf vrf-table-name port
12. fconfigure channelname -remote [host port] -broadcast boolean vrf [vrf_table_name]
13. udp_open -ipv6 port
14. udp_peek sock buffersize
15. exec “exec-cmd”
16. exit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>Example:</td>
<td>Router&gt; enable</td>
<td></td>
</tr>
<tr>
<td>Step 2</td>
<td>configure terminal</td>
<td>(Optional) Enters global configuration mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Perform Step 2 through Step 6 if you are using encoding files, an initialization script, or both.</td>
</tr>
<tr>
<td>Example:</td>
<td>Router# configure terminal</td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td>scripting tcl encdir location-url</td>
<td>(Optional) Specifies the default location of external encoding files used by the Tcl encoding command.</td>
</tr>
<tr>
<td>Example:</td>
<td>Router(config)# scripting tcl encdir tftp://10.18.117.23/enctcl/</td>
<td></td>
</tr>
</tbody>
</table>
### Command or Action

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 4</td>
<td>scripting tcl init init-url</td>
<td>(Optional) Specifies an initialization script to run when the Tcl shell is enabled.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td>Router(config)# scripting tcl init ftp://user:password@172.17.40.3/tclscript/initfiles3.tcl</td>
</tr>
<tr>
<td>Step 5</td>
<td>scripting tcl low-memory bytes</td>
<td>(Optional) Specifies a low water memory mark for free memory for Tcl-based applications. The memory threshold can be set anywhere between 0-4294967295 bytes.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td>Router(config)# scripting tcl low-memory 33117513</td>
</tr>
<tr>
<td>Step 6</td>
<td>exit</td>
<td>(Optional) Exits global configuration mode and returns to privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td>Router(config)# exit</td>
</tr>
<tr>
<td>Step 7</td>
<td>tclsh</td>
<td>Enables the interactive Tcl shell and enters Tcl configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td>Router# tclsh</td>
</tr>
<tr>
<td>Step 8</td>
<td>Enter the required Tcl command language syntax.</td>
<td>Commands entered in Tcl configuration mode are sent first to the interactive Tcl interpreter. If the command is not a valid Tcl command, it is then sent to the CLI parser.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td>Router(tcl)# proc get_bri {}</td>
</tr>
<tr>
<td>Step 9</td>
<td>ios_config &quot;cmd&quot; &quot;cmd-option&quot;</td>
<td>(Optional) Modifies the router configuration using a Tcl script by specifying the Tcl command ios_config with CLI commands and options. All arguments and submode commands must be entered on the same line as the CLI configuration command.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td>Router(tcl)# ios_config &quot;interface Ethernet 2/0&quot; &quot;no keepalive&quot;</td>
</tr>
</tbody>
</table>

- In this example, the first argument in quotes configures an Ethernet interface and enters interface configuration mode. The second argument in quotes sets the keepalive option. If these two CLI statements were entered on separate Tcl command lines, the configuration would not work.
### Command or Action

<table>
<thead>
<tr>
<th>Step 10</th>
<th><code>socket -myaddr addr -myport port -myvrf vrf-table-name host port</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong></td>
<td><code>Router(tcl)# socket -myaddr 10.4.9.34 -myport 12345 -myvrf testvrf 12346</code></td>
</tr>
</tbody>
</table>

Specifies the client socket and allows a TCL interpreter to connect via TCP over IPv4/IPv6 and opens a TCP network connection. You can specify a port and host to connect to; there must be a server to accept connections on this port.

- `-myaddr addr`—domain name or numerical IP address of the client-side network interface required for the connection. Use this option especially if the client machine has multiple network interfaces.
- `-myport port`—port number that is required for the client's connection.
- `-myvrf [vrf_table_name]`—specifies the vrf table name. If the vrf table is not configured, then the command will return a TCL_ERROR.

<table>
<thead>
<tr>
<th>Step 11</th>
<th><code>socket -server -myaddr addr -myvrf vrf-table-name port</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong></td>
<td><code>Router(tcl)# socket -server test -myvrf testvrf 12348</code></td>
</tr>
</tbody>
</table>

Specifies the server socket and allows a TCL interpreter to connect via TCP over IPv4/IPv6 and opens a TCP network connection. If the port is zero, Cisco IOS will allocate a free port to the server socket by using `fconfigure` command to read the `-sock0` argument.

- `-myaddr addr`—domain name or numerical IP address of the client-side network interface required for the connection. Use this option especially if the client machine has multiple network interfaces.
- `-myvrf vrf`—specifies the vrf table name. If the vrf table is not configured, then the command will return a TCL_ERROR and append “Cannot obtain VRF Table ID for VRF_table_name” to the interpreter result.

<table>
<thead>
<tr>
<th>Step 12</th>
<th><code>fconfigure channelname -remote [host port] -broadcast boolean -vrf [vrf_table_name]</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong></td>
<td><code>Router(tcl)# fconfigure sock1 -vrf vrf1 -remote [list 10.4.9.37 56009] -broadcast 1</code></td>
</tr>
</tbody>
</table>

Specifies the options in a channel.

- In case of UDP sockets that are created using the `udp_open`, the UDP socket can be mapped to a VRF using the `fconfigure` command.
- This command can also be used to display the properties of the channel.
- `-broadcast`—enables or disables the broadcasting.

<table>
<thead>
<tr>
<th>Step 13</th>
<th><code>udp_open -ipv6 port</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong></td>
<td><code>Router(tcl)# udp_open -ipv6 56005</code></td>
</tr>
</tbody>
</table>

Opens a UDP socket.

- If a port is specified the UDP socket will be opened on that port. Otherwise the system will choose a port and you can use the `fconfigure` command to obtain the port number, if required. If `-ipv6` argument is specified, the socket will be opened specifying the AF_INET6 protocol family.

<table>
<thead>
<tr>
<th>Step 14</th>
<th><code>udp Peek sock -buffersize buffer-size</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong></td>
<td><code>Router(tcl)# udp Peek sock0 -buffersize 100</code></td>
</tr>
</tbody>
</table>

Enables peeking into a UDP socket.

- `-buffersize buffer-size`—specifies the buffersize.
How to Configure Cisco IOS Scripting with Tcl

Examples

The following sample (partial) output shows information about Ethernet interface 0 on the router. The `show interfaces` command has been executed from Tcl configuration mode.

```
Router# tclsh
Router(tcl)# exec "show interfaces"
```

Troubleshooting Tips

Use the Tcl `puts` command in a Tcl script to trace command execution.

Using the Tcl Shell to Access SNMP MIB Objects

Perform this task to enable the interactive Tcl shell and enter Tcl commands to perform actions on MIB objects.

Prerequisites

The SNMP community configuration must exist in the running configuration of the router.

---

### Command or Action

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>exec “exec-cmd”</td>
<td>(Optional) Executes Cisco IOS CLI EXEC mode commands from a Tcl script by specifying the Tcl command exec with the CLI commands.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td>In this example, interface information for the router is displayed.</td>
</tr>
<tr>
<td></td>
<td>Router(tcl)# exec “show interfaces”</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>exit</td>
<td>Exits Tcl configuration mode and returns to privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Router(tcl)# exit</td>
<td></td>
</tr>
</tbody>
</table>

---

**Examples**

**Troubleshooting Tips**

Use the Tcl `puts` command in a Tcl script to trace command execution.

**Using the Tcl Shell to Access SNMP MIB Objects**

Perform this task to enable the interactive Tcl shell and enter Tcl commands to perform actions on MIB objects.

**Prerequisites**

The SNMP community configuration must exist in the running configuration of the router.
SUMMARY STEPS

1. enable
2. configure terminal
3. scripting tcl encdir location-url
4. scripting tcl init init-url
5. exit
6. tclsh
7. Enter any required Tcl command language syntax.
8. snmp_getbulk community-string non-repeaters max-repetitions oid [oid2 oid3...]
9. snmp_getid community-string
10. snmp_getnext community-string oid [oid2 oid3...]
11. snmp_getone community-string oid [oid2 oid3...]
12. snmp_setany community-string oid type val [oid2 type2 val2...]
13. exit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td><strong>enable</strong></td>
</tr>
<tr>
<td>Example:</td>
<td>.Router&gt; enable</td>
</tr>
<tr>
<td></td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td><strong>configure terminal</strong></td>
</tr>
<tr>
<td>Example:</td>
<td>.Router# configure terminal</td>
</tr>
<tr>
<td></td>
<td>(Optional) Enters global configuration mode.</td>
</tr>
<tr>
<td></td>
<td>• Perform <strong>Step 2 through Step 5</strong> Perform Step 2 through Step 5 if you are using encoding files, an initialization script, or both.</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td><strong>scripting tcl encdir location-url</strong></td>
</tr>
<tr>
<td>Example:</td>
<td>.Router(config)# scripting tcl encdir tftp://10.18.117.23/enctcl/</td>
</tr>
<tr>
<td></td>
<td>(Optional) Specifies the default location of external encoding files used by the Tcl <strong>encoding</strong> command.</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td><strong>scripting tcl init init-url</strong></td>
</tr>
<tr>
<td>Example:</td>
<td>.Router(config)# scripting tcl init ftp://user:password@172.17.40.3/tclscript/initfiles3.tcl</td>
</tr>
<tr>
<td></td>
<td>(Optional) Specifies an initialization script to run when the Tcl shell is enabled.</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td><strong>exit</strong></td>
</tr>
<tr>
<td>Example:</td>
<td>.Router(config)# exit</td>
</tr>
<tr>
<td></td>
<td>(Optional) Exits global configuration mode and returns to privileged EXEC mode.</td>
</tr>
</tbody>
</table>
### Command or Action

**Step 6**

**tclsh**

**Example:**

Router# tclsh

**Purpose:** Enables the interactive Tcl shell and enters Tcl configuration mode.

**Step 7**

**Enter the required Tcl command language syntax.**

**Example:**

Router(tcl)# proc get_bri {}  

Commands entered in Tcl configuration mode are sent first to the interactive Tcl interpreter. If the command is not a valid Tcl command, it is sent to the CLI parser.

**Step 8**

**snmp_getbulk**  

**community-string**  

**non-repeaters**  

**max-repetitions**  

**oid** [**oid2 oid3...**]  

**Example:**

Router(tcl)# snmp_getbulk public 1 3  
1.3.6.1.2.1.1.1 1.3.6.1.2.1.10.18.8.1.1

(Optional) Retrieves a large section of a MIB table.  

- Use the **community-string** argument to specify the SNMP community from which the objects will be retrieved.  
- Use the **non-repeaters** argument to specify the number of objects that can be retrieved with a get-next operation.  
- Use the **max-repetitions** argument to specify the maximum number of get-next operations to attempt while trying to retrieve the remaining objects.  
- Use the **oid** argument to specify the object ID(s) to retrieve.

**Step 9**

**snmp_getid**  

**community-string**  

**Example:**

Router(tcl)# snmp_getid private

(Optional) Retrieves the following variables from the SNMP entity on the router: sysDescr.0, sysObjectID.0, sysUpTime.0, sysContact.0, sysName.0, and sysLocation.0.  

- Use the **community-string** argument to specify the SNMP community from which the objects will be retrieved.

**Step 10**

**snmp_getnext**  

**community-string**  

**oid** [**oid2 oid3...**]  

**Example:**

Router(tcl)# snmp_getnext public 1.3.6.1.2.1.1.1.0 1.3.6.1.2.1.1.2.0

(Optional) Retrieves a set of individual variables from a MIB table.  

- Use the **community-string** argument to specify the SNMP community from which the objects will be retrieved.  
- Use the **oid** argument to specify the object ID(s) to retrieve.

**Step 11**

**snmp_getone**  

**community-string**  

**oid** [**oid2 oid3...**]  

**Example:**

Router(tcl)# snmp_getone public 1.3.6.1.2.1.1.1.0 1.3.6.1.2.1.1.2.0

(Optional) Retrieves a set of individual variables from a MIB table.  

- Use the **community-string** argument to specify the SNMP community from which the objects will be retrieved.  
- Use the **oid** argument to specify the object ID(s) to retrieve.
### Troubleshooting Tips

Use the Tcl `puts` command in a Tcl script to trace command execution.

### Running Predefined Tcl Scripts

Perform this optional task to run a predefined Tcl script in Cisco IOS software.

### Prerequisites

Before performing this task, you must create a Tcl script that can run on Cisco IOS software. The Tcl script may be transferred to internal flash memory using any file system that the Cisco IOS file system (IFS) supports, including TFTP, FTP, and rcp. The Tcl script may also be sourced from a remote location.

### SUMMARY STEPS

1. `enable`
2. `tclsh`
3. Enter the Tcl source command with the filename and path.
4. `exit`

---

### Command or Action

<table>
<thead>
<tr>
<th>Step 12</th>
<th><code>snmp_setany community-string oid type val [oid2 type2 val2...]</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td><code>Router(tcl)# snmp_setany private 1.3.6.1.2.1.1.5.0 -d TCL-SNMP_TEST</code></td>
</tr>
<tr>
<td>Purpose</td>
<td>(Optional) Retrieves current values of specified variables from a MIB table and then performs a set request on the variables.</td>
</tr>
<tr>
<td></td>
<td>- Use the <code>community-string</code> argument to specify the SNMP community from which the values of objects will be retrieved and then set.</td>
</tr>
<tr>
<td></td>
<td>- Use the <code>oid</code> argument to specify the object ID(s) to retrieve and set.</td>
</tr>
<tr>
<td></td>
<td>- Use the <code>type</code> argument to specify the type of object to retrieve and set.</td>
</tr>
<tr>
<td></td>
<td>- Use the <code>val</code> argument to specify the value of the object to be retrieved and then set.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 13</th>
<th><code>exit</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td><code>Router(tcl)# exit</code></td>
</tr>
<tr>
<td>Purpose</td>
<td>Exits Tcl configuration mode and returns to privileged EXEC mode.</td>
</tr>
</tbody>
</table>
## Configuration Examples for Cisco IOS Scripting with Tcl

This section provides the following configuration examples:

- Example: Tcl Script Using the show interfaces Command, page 14
- Example: Tcl Script for SMTP Support, page 15
- Example: Tcl Script for SNMP MIB Access, page 17

### Example: Tcl Script Using the show interfaces Command

Using the Tcl regular expression engine, scripts can filter specific information from `show` commands and present it in a custom format. The following is an example of filtering the `show interfaces` output and creating a comma-separated list of BRI interfaces on the router:

```tcl
proc get_bri {} {
    set check ""
    set int_out [exec "show interfaces"]
    foreach int [regexp -all -line -inline "(^BRI\[0-9]/\[0-9])" $int_out] {
        if {![string equal $check $int]} {
            if {([info exists bri_out])} {
                if {([info exists bri_out])} {
                    append bri_out "," $int
                    if {([info exists bri_out])} {
                        set bri_out "$int"
                    } else {
                        set bri_out $int
                    }
                } else {
                    set check $int
                }
            } else {
                set check "$int"
            }
        }
    }
    return bri_out
}
```

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Step</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>enable</code></td>
<td>1</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Enter your password if prompted.</td>
</tr>
<tr>
<td><code>tclsh</code></td>
<td>2</td>
<td>Enables the interactive Tcl shell and enters Tcl configuration mode.</td>
</tr>
<tr>
<td><code>source slot0:test.tcl</code></td>
<td>3</td>
<td>Commands entered in Tcl configuration mode are sent first to the interactive Tcl interpreter. If the command is not a valid Tcl command, it is then sent to the CLI parser.</td>
</tr>
<tr>
<td><code>exit</code></td>
<td>4</td>
<td>Exits Tcl configuration mode and returns to privileged EXEC mode.</td>
</tr>
</tbody>
</table>

### Command or Action Purpose

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><code>enable</code></td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Enter your password if prompted.</td>
</tr>
<tr>
<td>2</td>
<td><code>tclsh</code></td>
<td>Enables the interactive Tcl shell and enters Tcl configuration mode.</td>
</tr>
<tr>
<td>3</td>
<td><code>source slot0:test.tcl</code></td>
<td>Commands entered in Tcl configuration mode are sent first to the interactive Tcl interpreter. If the command is not a valid Tcl command, it is then sent to the CLI parser.</td>
</tr>
<tr>
<td>4</td>
<td><code>exit</code></td>
<td>Exits Tcl configuration mode and returns to privileged EXEC mode.</td>
</tr>
</tbody>
</table>
Example: Tcl Script for SMTP Support

The following Tcl script is useful for sending e-mail messages from a router.

```tcl
##
## Place required comments here!!!
##
package provide sendmail 2.0

# Sendmail procedure for Support
namespace eval ::sendmail {
    namespace export initialize configure sendmessage sendfile
    array set ::sendmail::sendmail {
        smthost    mailhub
        from        ""
        friendly    ""
    }
    proc configure {} {}
    proc initialize {smtphost from friendly} {
        variable sendmail
        if {[string length $smtphost]} then {
            set sendmail(smtphost) $smtphost
        }
        if {[string length $from]} then {
            set sendmail(from) $from
        }
        if {[string length $friendly]} then {
            set sendmail(friendly) $friendly
        }
    }
    proc sendmessage {toList subject body {tcl_trace 0}} {
        variable sendmail
        set smthost $sendmail(smtphost)
        set from $sendmail(from)
        set friendly $sendmail(friendly)
        if {$trace} then {
            puts stdout "Connecting to $smtphost:25"
        }
        set sockid [socket $smtphost 25]
        ## DEBUG
        set status [catch {
            puts $sockid "HELO $smtphost"
            flush $sockid
            set result [gets $sockid]
```
if ($trace) then {
    puts stdout "HELO $smtphost\n\t$result"
}

puts $sockid "MAIL From:<$from>"
flush $sockid
set result [gets $sockid]
if ($trace) then {
    puts stdout "MAIL From:<$from>\n\t$result"
}

foreach to $toList {
    puts $sockid "RCPT To:<$to>"
    flush $sockid
}

set result [gets $sockid]
if ($trace) then {
    puts stdout "RCPT To:<$to>\n\t$result"
}

puts $sockid "DATA "
flush $sockid
set result [gets $sockid]
if ($trace) then {
    puts stdout "DATA \n\t$result"
}

puts $sockid "From: $friendly <$from>"
foreach to $toList {
    puts $sockid "To:<$to>"
}
puts $sockid "Subject: $subject"
puts $sockid "\n"
foreach line [split $body "\n"] {
    puts $sockid "$line"
}

puts $sockid "."
puts $sockid 'QUIT'
flush $sockid
set result [gets $sockid]
if ($trace) then {
    puts stdout "QUIT\n\t$result"
}

} result]
catch {close $sockid }
if ($status) then {
    return -code error $result
}
return
}

proc sendfile {toList filename subject {tcl_trace 0}} {
    set fd [open $filename r]
sendmessage $toList $subject [read $fd] $trace
Example: Tcl Script for SNMP MIB Access

Using the Tcl shell, Tcl commands can perform actions on MIBs. The following example shows how to set up the community access strings to permit access to SNMP. Public access is read-only, but private access is read-write. The following example shows how to retrieve a large section of a table at once using the `snmp_getbulk` Tcl command extension.

Two arguments, `non-repeaters` and `max-repetitions`, must be set when an `snmp_getbulk` command is issued. The `non-repeaters` argument specifies that the first N objects are to be retrieved with a simple `snmp_getnext` operation. The `max-repetitions` argument specifies that up to M `snmp_getnext` operations are to be attempted to retrieve the remaining objects.

In this example, three bindings—sysUpTime (1.3.6.1.2.1.1.2.0), ifDescr (1.3.6.1.2.1.2.2.1.2), and ifType (1.3.6.1.2.1.2.2.1.3)—are used. The total number of variable bindings requested is given by the formula N + (M * R), where N is the number of non-repeaters (in this example 1), M is the max-repetitions (in this example 5), and R is the number of request objects (in this case 2, ifDescr and ifType). Using the formula, 1 + (5 * 2) equals 11; and this is the total number of variable bindings that can be retrieved by this `snmp_getbulk` request command.

Sample results for the individual variables include a retrieved value of sysUpTime.0 being 1336090, where the unit is in milliseconds. The retrieved value of ifDescr.1 (the first interface description) is FastEthernet0/0, and the retrieved value of ifType.1 (the first interface type) is 6, which corresponds to the ethernetCsmacd type.

```
snmp-server community public RO
snmp-server community private RW

tclsh
snmp_getbulk public 1 5 1.3.6.1.2.1.1.2.0 1.3.6.1.2.1.2.2.1.2 1.3.6.1.2.1.2.2.1.3
  (obj oid='sysUpTime.0' val='1336090'/>)
  (obj oid='ifDescr.1' val='FastEthernet0/0'/>)
  (obj oid='ifType.1' val='6'/>)
  (obj oid='ifDescr.2' val='FastEthernet1/0'/>)
  (obj oid='ifType.2' val='6'/>)
  (obj oid='ifDescr.3' val='Ethernet2/0'/>)
  (obj oid='ifType.3' val='6'/>)
  (obj oid='ifDescr.4' val='Ethernet2/1'/>)
  (obj oid='ifType.4' val='6'/>)
  (obj oid='ifDescr.5' val='Ethernet2/2'/>)
  (obj oid='ifType.5' val='6'/>)
```

The following example shows how to retrieve the sysDescr.0, sysObjectID.0, sysUpTime.0, sysContact.0, sysName.0, and sysLocation.0 variables—in this example shown as system.1.0, system.2.0, system.3.0, system.4.0, system.5.0, and system.6.0—from the SNMP entity on the router using the `snmp_getid` Tcl command extension.

```
tclsh
snmp_getid public
  (obj oid='system.1.0' val='Cisco Internetwork Operating System Software
Cisco IOS(tm) 7200 Software (C7200-IK9S-M), Experimental Version 12.3(20030507:225511)
[geotpi2itd1 124]
Copyright (c) 1986-2003 by Cisco Systems, Inc.
Compiled Wed 21-May-03 16:16 by engineer'/>)
  (obj oid='system.2.0' val='products.223'/>)
  (obj oid='sysUpTime.0' val='6664317'/>)
  (obj oid='system.4.0' val='1-800-553-2447 - phone the TAC'/>)
```
The following example shows how to retrieve a set of individual variables from the SNMP entity on the router using the **snmp_getnext** Tcl command extension:

```
snmp_getnext public 1.3.6.1.2.1.1.1.0 1.3.6.1.2.1.1.2.0
{<obj oid='system.2.0' val='products.223'/>}
{<obj oid='sysUpTime.0' val='6683320'/>}
```

The following example shows how to retrieve a set of individual variables from the SNMP entity on the router using the **snmp_getone** Tcl command extension:

```
snmp_getone public 1.3.6.1.2.1.1.1.0 1.3.6.1.2.1.1.2.0
{<obj oid='system.1.0' val='Cisco Internetwork Operating System Software
Cisco IOS(tm) 7200 Software (C7200-IK9S-M), Experimental Version 12.3(20030507:225511)
[geotpi2itd1 124]
Copyright (c) 1986-2003 by Cisco Systems, Inc.
Compiled Wed 21-May-03 16:16 by engineer'/>}
{<obj oid='system.2.0' val='products.223'/>}
```

The following example shows how to change something in the configuration of the router using the **snmp_setany** Tcl command extension. In this example, the hostname of the router is changed to TCLSNMP-HOST.

```
tclsh
snmp_setany private 1.3.6.1.2.1.1.5.0 -d TCLSNMP-HOST
{<obj oid='system.5.0' val='TCLSNMP-HOST'/>}
```
# Additional References

The following sections provide references related to the Cisco IOS Scripting with Tcl feature.

## Related Documents

<table>
<thead>
<tr>
<th>Related Topic</th>
<th>Document Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Embedded Syslog Manager</td>
<td>Embedded Syslog Manager module</td>
</tr>
<tr>
<td>Network Management commands (including Tcl and logging commands): complete command syntax, defaults, command mode, command history, usage guidelines, and examples</td>
<td><em>Cisco IOS Network Management Command Reference</em></td>
</tr>
</tbody>
</table>

## Standards

<table>
<thead>
<tr>
<th>Standards</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.</td>
<td>—</td>
</tr>
</tbody>
</table>

## MIBs

<table>
<thead>
<tr>
<th>MIBs</th>
<th>MIBs Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>No new or modified MIBs are supported by this feature, and support for existing MIBs has not been modified by this feature.</td>
<td>To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: <a href="http://www.cisco.com/go/mibs">http://www.cisco.com/go/mibs</a></td>
</tr>
</tbody>
</table>

## RFCs

<table>
<thead>
<tr>
<th>RFCs</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature.</td>
<td>—</td>
</tr>
</tbody>
</table>
Technical Assistance

<table>
<thead>
<tr>
<th>Description</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies.</td>
<td><a href="http://www.cisco.com/cisco/web/support/index.html">http://www.cisco.com/cisco/web/support/index.html</a></td>
</tr>
<tr>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.</td>
<td></td>
</tr>
</tbody>
</table>
Feature Information for Cisco IOS Scripting with Tcl

Table 4 lists the release history for this feature.

Not all commands may be available in your Cisco IOS software release. For release information about a specific command, see the command reference documentation.

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which Cisco IOS and Catalyst OS software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Note

Table 4 lists only the Cisco IOS software release that introduced support for a given feature in a given Cisco IOS software release train. Unless noted otherwise, subsequent releases of that Cisco IOS software release train also support that feature.

Table 4 Feature Information for Cisco IOS Scripting with Tcl

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cisco IOS Scripting with Tcl</td>
<td>12.3(2)T</td>
<td>The Cisco IOS Scripting with Tcl feature provides the ability to run Tcl version 8.3.4 commands from the Cisco IOS command-line interface.</td>
</tr>
<tr>
<td></td>
<td>12.3(7)T</td>
<td></td>
</tr>
<tr>
<td></td>
<td>12.2(25)S</td>
<td></td>
</tr>
<tr>
<td></td>
<td>12.2(33)SXH</td>
<td></td>
</tr>
<tr>
<td></td>
<td>12.2(33)SRC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>12.2(33)SB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cisco IOS XE 3.1.0SG</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The following commands were introduced or modified:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>scripting tcl encdir, scripting tcl init, scripting tcl low-memory, tclquit, tclsh</td>
</tr>
<tr>
<td>TCL SNMP MIB Access</td>
<td>12.3(7)T</td>
<td>The Tcl SNMP MIB Access feature introduces a set of UNIX-like SNMP commands to make access to Simple Network Management Protocol (SNMP) MIB objects easier.</td>
</tr>
<tr>
<td></td>
<td>12.2(25)S</td>
<td></td>
</tr>
<tr>
<td></td>
<td>12.2(33)SXH</td>
<td></td>
</tr>
<tr>
<td></td>
<td>12.2(33)SRC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>12.2(33)SB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cisco IOS XE 3.1.0SG</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The following section has more information:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Using the Tcl Shell to Access SNMP MIB Objects, page 10.</td>
</tr>
<tr>
<td>TCL UDP and VRF support</td>
<td>15.1(1)T</td>
<td>The Tcl UDP and VRF feature provides support for UDP sockets in IOS Tcl.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The following commands were introduced or modified:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>fconfigure, socket, udp_open, udp Peek</td>
</tr>
</tbody>
</table>
Glossary

ESM—Embedded Syslog Manager.
IVR—Interactive Voice Response.
MIB—Management Information Base.
Tcl—Tool Command Language.

Note
See *Internetworking Terms and Acronyms* for terms not included in this glossary.