Cisco IOS Scripting with Tcl

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).

Feature History for Cisco IOS Scripting with Tcl

Release	Modification
12.3(2)T	This feature was introduced.
12.3(7)T	Support was added for accessing SNMP MIB objects using Tcl.
12.2(25)S	This feature was integrated into Cisco IOS Release 12.2(25)S.
12.2(33)SXH	This feature was integrated into Cisco IOS Release 12.2(33)SXH.

Contents

Prerequisites for Cisco IOS Scripting with Tcl

- Familiarity with Tcl programming and Cisco IOS commands is assumed.

Cisco Systems

Corporate Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

- Tcl commands can be executed from the Tcl configuration mode using the Cisco IOS CLI. Tcl configuration mode, like global 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 1 Tcl Command Options That Behave Differently in Cisco IOS Software

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

Table 1 Tcl Command Options That Behave Differently in Cisco IOS Software (continued)

Command	Keyword	Argument	Supported	Comments
history	**!***n*		Partially	The **!***n* shortcut does not work in Cisco IOS software. Use the **history** Tcl command with the **redo** *n* keyword.
load			No	When the CLI **load** command is used, an error message stating “dynamic loading not available on this system” is displayed.

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

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. 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.

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 4 (required)
- Using the Tcl Shell to Access SNMP MIB Objects, page 7 (optional)
- Running Predefined Tcl Scripts, page 11 (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.

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

Command	Description
ios_config	Runs a Cisco IOS CLI configuration command.
log_user	Toggles Tcl command output under Tcl configuration mode.
typeahead	Writes text to the router standard input (stdin) buffer file.

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 the required Tcl command language syntax.
8. **ios_config "***cmd***" "***cmd-option***"**

9. **exec "***exec-cmd***"**
10. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	(Optional) Enters global configuration mode. • Perform Step 2 through Step 5 if you are using encoding files, an initialization script, or both.
Step 3	**scripting tcl encdir** *location-url* **Example:** `Router(config)# scripting tcl encdir tftp://10.18.117.23/enctcl/`	(Optional) Specifies the default location of external encoding files used by the Tcl **encoding** command.
Step 4	**scripting tcl init** *init-url* **Example:** `Router(config)# scripting tcl init ftp://user:password@172.17.40.3/tclscript/initfiles3.tcl`	(Optional) Specifies an initialization script to run when the Tcl shell is enabled.
Step 5	**exit** **Example:** `Router(config)# exit`	(Optional) Exits global configuration mode and returns to privileged EXEC mode.
Step 6	**tclsh** **Example:** `Router# tclsh`	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 then sent to the CLI parser.

	Command or Action	Purpose
Step 8	**ios_config "***cmd***" "***cmd-option***"** **Example:** `Router(tcl)# ios_config "interface Ethernet 2/0" "no keepalive"`	(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. • 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.
Step 9	**exec "***exec-cmd***"** **Example:** `Router(tcl)# exec "show interfaces"`	(Optional) Executes Cisco IOS CLI EXEC mode commands from a Tcl script by specifying the Tcl command **exec** with the CLI commands. • In this example, interface information for the router is displayed.
Step 10	**exit** **Example:** `Router(tcl)# exit`	Exits Tcl configuration mode and returns to privileged EXEC mode.

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"

Ethernet 0 is up, line protocol is up
  Hardware is MCI Ethernet, address is 0000.0c00.750c (bia 0000.0c00.750c)
  Internet address is 10.108.28.8, subnet mask is 255.255.255.0
  MTU 1500 bytes, BW 10000 Kbit, DLY 100000 usec, rely 255/255, load 1/255
  Encapsulation ARPA, loopback not set, keepalive set (10 sec)
  ARP type: ARPA, ARP Timeout 4:00:00
  Last input 0:00:00, output 0:00:00, output hang never
  Last clearing of "show interface" counters 0:00:00
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  Five minute input rate 0 bits/sec, 0 packets/sec
  Five minute output rate 2000 bits/sec, 4 packets/sec
     1127576 packets input, 447251251 bytes, 0 no buffer
     Received 354125 broadcasts, 0 runts, 0 giants, 57186* throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     5332142 packets output, 496316039 bytes, 0 underruns
     0 output errors, 432 collisions, 0 interface resets, 0 restarts
.
.
.
```

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

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

Command	Description
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*...]

Table 3 Cisco IOS Custom Tcl Command Extensions for SNMP MIB Access (continued)

Command	Description
snmp_getone	Retrieves a set of individual variables from the SNMP entity on the router. This command is similar to the SNMP **getone** command. The syntax is in the following format: **snmp_getone** *community-string oid* [*oid2 oid3...*]
snmp_setany	Retrieves the current values of the specified variables and then performs a set request on the variables. This command is similar to the SNMP **setany** command. The syntax is in the following format: **snmp_setany** *community-string oid type val* [*oid2 type2 val2...*] • Use the *type* argument to specify the type of object to retrieve. The *type* can be one of the following: – **-i**—Integer. A 32-bit number used to specify a numbered type within the context of a managed object. For example, to set the operational status of a router interface, 1 represents up and 2 represents down. – **-u**—Unsigned32. A 32-bit number used to represent decimal values in the range from 0 to $2^{32} - 1$ inclusive. – **-c**—Counter32. A 32-bit number with a minimum value of 0 and a maximum value of $2^{32} - 1$. When the maximum value is reached, the counter resets to 0 and starts again. – **-g**—Gauge. A 32-bit number with a minimum value of 0 and a maximum value of $2^{32} - 1$. The number can increase or decrease at will. For example, the interface speed on a router is measured using a gauge object type. – **-o**—Octet string. An octet string—in hex notation—used to represent physical addresses. – **-d**—Display string. An octet string—in text notation—used to represent text strings. – **-ipv4**—IP version 4 address. – **-oid**—Object ID. • Use the *val* argument to specify the value of object ID(s) to retrieve.

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

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	(Optional) Enters global configuration mode. • Perform Step 2 through Step 5 Perform Step 2 through Step 5 if you are using encoding files, an initialization script, or both.
Step 3	**scripting tcl encdir** *location-url* **Example:** `Router(config)# scripting tcl encdir tftp://10.18.117.23/enctcl/`	(Optional) Specifies the default location of external encoding files used by the Tcl **encoding** command.
Step 4	**scripting tcl init** *init-url* **Example:** `Router(config)# scripting tcl init ftp://user:password@172.17.40.3/tclscript/initfiles3.tcl`	(Optional) Specifies an initialization script to run when the Tcl shell is enabled.
Step 5	**exit** **Example:** `Router(config)# exit`	(Optional) Exits global configuration mode and returns to privileged EXEC mode.
Step 6	**tclsh** **Example:** `Router# tclsh`	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.

	Command or Action	Purpose
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: sysDesrc.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.

	Command or Action	Purpose
Step 12	**snmp_setany** *community-string oid type val* [*oid2 type2 val2...*] **Example:** `Router(tcl)# snmp_setany private 1.3.6.1.2.1.1.5.0 -d TCL-SNMP_TEST`	(Optional) Retrieves current values of specified variables from a MIB table and then performs a set request on the variables. • Use the *community-string* argument to specify the SNMP community from which the values of objects will be retrieved and then set. • Use the *oid* argument to specify the object ID(s) to retrieve and set. • Use the *type* argument to specify the type of object to retrieve and set. • Use the *val* argument to specify the value of the object to be retrieved and then set.
Step 13	**exit** **Example:** `Router(tcl)# exit`	Exits Tcl configuration mode and returns to privileged EXEC mode.

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**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**tclsh** **Example:** `Router# tclsh`	Enables the interactive Tcl shell and enters Tcl configuration mode.
Step 3	Enter the Tcl source command with the filename and path. **Example:** `Router(tcl)# source slot0:test.tcl`	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.
Step 4	**exit** **Example:** `Router(tcl)# exit`	Exits Tcl configuration mode and returns to privileged EXEC mode.

Configuration Examples for Cisco IOS Scripting with Tcl

This section contains the following configuration examples:

Tcl Script Using the show interfaces Command: Example

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** command output and creating a comma-separated list of BRI interfaces on the router:

```
tclsh
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]} {
                append bri_out "," $int
             } else {
                 set bri_out $int
             }
             set check $int
        }
    }
```

```
    return $bri_out
}
```

Tcl Script for SMTP Support: Example

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

```
##
## 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 {
        smtphost    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 smtphost $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
```

```
        return
    }

}
```

Tcl Script for SNMP MIB Access: Examples

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'/>}
 {<obj oid='system.5.0' val='c7200.myCompany.com'/>}
 {<obj oid='system.6.0' val='Bldg 24, San Jose, CA'/>}
```

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 additional information related to the Cisco IOS Scripting with Tcl feature.

Related Documents

Related Topic	Document Title
Embedded Syslog Manager	*Embedded Syslog Manager* feature document, Release 12.3(2)T
Command references for Cisco IOS Release 12.2	Cisco IOS Command References, Release 12.2
Command references for Cisco IOS Release 12.3 T	Cisco IOS Command References, Release 12.3 T

Standards

Standards	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	—

MIBs

MIBs	MIBs Link
No new or modified MIBs are supported by this feature, and support for existing MIBs has not been modified by this feature.	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFCs	Title
No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature.	—

Technical Assistance

Description	Link
Technical Assistance Center (TAC) home page, containing 30,000 pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/public/support/tac/home.shtml

Command Reference

This section documents new commands.

- **scripting tcl encdir**
- **scripting tcl init**
- **tclsh**

scripting tcl encdir

To specify the default location of external encoding files used by the Tool Command Language (Tcl) shell, use the **scripting tcl encdir** command in global configuration mode. To remove the default location, use the **no** form of this command.

scripting tcl encdir *location-url*

no scripting tcl encdir

Syntax Description

location-url	URL used to access external encoding files used by the Tcl shell.

Defaults

Tcl does not use external encoding files.

Command Modes

Global configuration

Command History

Release	Modification
12.3(2)T	This command was introduced.
12.2(25)S	This command was integrated into Cisco IOS Release 12.2(25)S.

Usage Guidelines

Character strings in Tcl are encoded using 16-bit Unicode characters. Different operating system interfaces or applications can generate character strings using other encoding methods. Use the **scripting tcl encdir** command to configure a location URL for the external Tcl character encoding files to support the Tcl **encoding** command.

Tcl contains only a few character sets within the Tcl shell. Additional characters sets are loaded, as needed, from external files.

Examples

The following example shows how to specify a default location for external encoding files to be used by the Tcl shell:

```
Router# config terminal
Router(config)# scripting tcl encdir tftp://10.18.117.23/file2/
```

Related Commands

Command	Description
scripting tcl init	Specifies an initialization script for the Tcl shell.
tclsh	Enables the Tcl shell and enters Tcl configuration mode.

scripting tcl init

To specify an initialization script for the Tool Command Language (Tcl) shell, use the **scripting tcl init** command in global configuration mode. To remove the initialization script, use the **no** form of this command.

scripting tcl init *init-url*

no scripting tcl init

Syntax Description		
	init-url	URL used to access the initialization script to be used by the Tcl shell.

Defaults

Tcl does not run an initialization script.

Command Modes

Global configuration

Command History

Release	Modification
12.3(2)T	This command was introduced.
12.2(25)S	This command was integrated into Cisco IOS Release 12.2(25)S.

Usage Guidelines

Use the **scripting tcl init** command when you want to predefine Tcl procedures to run in an initialization script. The initialization script runs when the Tcl shell is enabled and saves manual sourcing of the individual scripts.

Examples

The following example shows how to specify an initialization script to run when the Tcl shell is enabled:

```
Router# config terminal
Router(config)# scripting tcl init ftp://user:password@172.17.40.3/tclscript/initfile3.tcl
```

Related Commands

Command	Description
scripting tcl encdir	Specifies the default location of external encoding files used by the Tcl shell.
tclsh	Enables the Tcl shell and enters Tcl configuration mode.

tclsh

To enable the interactive Tool Command Language (Tcl) shell and to enter Tcl configuration mode, use the **tclsh** command in privileged EXEC mode.

tclsh

Syntax Description

This command has no arguments or keywords.

Defaults

The Tcl shell is disabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(2)T	This command was introduced.
12.2(25)S	This command was integrated into Cisco IOS Release 12.2(25)S.

Usage Guidelines

Use the **tclsh** command when you want to run Tcl commands from the Cisco IOS command-line interface (CLI). When the interactive Tcl shell is enabled and Tcl configuration mode is entered, Tcl commands can be entered line by line or a predefined Tcl script can be run. 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. 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.

Use the Cisco IOS CLI **exit** command or the Tcl **tclquit** command to disable the use of the Tcl shell and return to privileged EXEC mode.

Examples

The following example shows how to enable the interactive Tcl shell:

```
Router# tclsh
Router(tcl)#
```

Related Commands

Command	Description
exit	Exits from the current configuration mode to the next highest configuration mode.
scripting tcl encdir	Specifies the default location of external encoding files used by the Tcl shell.
scripting tcl init	Specifies an initialization script for the Tcl shell.

Glossary

ESM—Embedded Syslog Manager.

IVR—Interactive Voice Response.

MIB—Management Information Base.

SNMP—Simple Network Management Protocol.

Tcl—Tool Command Language.

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