Tool Command Language (TCL) on the Cisco Router.

Introduction.

TCL is a scripting language available on the Cisco router. Although there are a number of articles on the Internet I thought it might be interesting to illustrate various elements of the facility. Cisco use it in the voice arena to provide fax to email (app_faxmail_onramp.2.0.1.3.tcl) and automated attendants (its-CISCO.2.0.1.0.tcl). It can be used to automate processes, format output, perform SNMP tasks and change the configuration. Below are various references I have used.

References

<table>
<thead>
<tr>
<th>Descriptions</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction to TCL</td>
<td><a href="http://www.tcl.tk/man/tcl8.5/tutorial/tcltutorial.html">http://www.tcl.tk/man/tcl8.5/tutorial/tcltutorial.html</a></td>
</tr>
<tr>
<td>Examples</td>
<td><a href="http://www.netcraftsmen.net/welcher/papers/iostcl01.html">http://www.netcraftsmen.net/welcher/papers/iostcl01.html</a></td>
</tr>
<tr>
<td>Regular Expression</td>
<td><a href="http://www.nil.si/ipcorner/EnhanceIOSUI">http://www.nil.si/ipcorner/EnhanceIOSUI</a> (last page)</td>
</tr>
<tr>
<td>Reachability example</td>
<td><a href="http://www.internetworkexpert.com/resources/tclshrsh.htm">http://www.internetworkexpert.com/resources/tclshrsh.htm</a></td>
</tr>
<tr>
<td>TCL for configuration</td>
<td><a href="http://www.nil.com/ipcorner/ConfigReplacement">http://www.nil.com/ipcorner/ConfigReplacement</a></td>
</tr>
<tr>
<td>Regular Expression</td>
<td><a href="http://www.regular-expressions.info/quickstart.html">http://www.regular-expressions.info/quickstart.html</a></td>
</tr>
</tbody>
</table>

Invoking TCL.

TCL statements can be stored in flash or run interactively. My examples are interactive. To invoke TCL use tclsh and to terminate type exit.

First display the TCL version.
Router#tclsh
Router(tcl)#info patchlevel
8.3.4

Note version 8.3.4 may not support some of the newer facilities such as lazy matching described later.
Commands, directories, files and loops.

Now display the available commands.

Router(tcl)#info commands
tell socket subst open eof pwd glob list exec pid snmp_getone time eval lrange
tcl_trace fblocked lsearch gets case lappend proc break variable llength return linsert
snmp_getid error catch clock info split array if log_user fconfigure concat join
lreplace snmp_setany source fcopy global switch snmp_getbulk update close cd for
file append format read package set binary namespace scan seek while flush after
vwait snmp_getnext typeahead uplevel continue hostname ios_config foreach rename
fileevent regexp upvar unset encoding expr load interp history puts incr lindex
lsort string

So let us list all the files on flash:
Router#tclsh
Router(tcl)#foreach line [glob *] {puts $line}
Produces a list of all the files in flash:

[glob *] returns a list of all the files. [menu.xml text 7921]
foreach line assigns each element of the list to the variable line and then runs the
commands within {}. puts $line simply prints out the value of the variable line.

Router(tcl)#foreach line [glob *] {puts $line}
menu.xml
text
7921

You can also loop incrementally.

Router(tcl)#for {set i 0} {$i < 10} {incr i} {
    puts "I inside loop: $i"
}
    I inside loop: 0
    I inside loop: 1
    I inside loop: 2
    ...
    I inside loop: 9
SNMP, writing files and configuration.

Here is an example of a SNMP read to determine the number of interfaces (71) (.iso.org.dod.internet.mgmt.mib-2.interfaces.ifNumber)

Router(tcl)#snmp_getone public 1.3.6.1.2.1.2.1.0
   {<obj oid='ifNumber.0' val='71'/>}

And then retrieve all the interface names (.iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable.ifEntry.ifDescr)

Router(tcl)#snmp_getbulk public 0 71 1.3.6.1.2.1.2.2.1.2
   {<obj oid='ifDescr.1' val='FastEthernet0/0'/>}
   {<obj oid='ifDescr.2' val='Serial0/0'/>}
   {<obj oid='ifDescr.3' val='FastEthernet0/1'/>}
   {<obj oid='ifDescr.4' val='Serial0/1'/>}
   {<obj oid='ifDescr.5' val='Null0'/>}
   {<obj oid='ifDescr.6' val='E1 1/0'/>}

Let us write a file

Router(tcl)#set outfile [ open "listing.out" w]
file7
Router(tcl)#foreach line [glob *] {puts $outfile $line}

Router(tcl)#close $outfile

Router(tcl)#exit
Router#more listing.out
Display filename [/listing.out]?
menu.xml
text
7921

Let us change the configuration.

Router(tcl)#puts [ios_config "Interface FastEthernet2/0" "shut"]
*Jun 23 23:36:06: %LINK-5-CHANGED: Interface FastEthernet2/0, changed state to administratively down
Router(tcl)#puts [ios_config "Interface FastEthernet2/0" "no shut"]
*Jun 23 23:36:18: %LINK-3-UPDOWN: Interface FastEthernet2/0, changed state to up
Pattern matching and lists.

TCL has a pattern matching facility. This allows you to break down text into interesting elements. This is achieved through a regular expression. () indicate a sub-match and w+ means a alphabetic string of one or more characters.

```
set text "the cat sat on the mat"
set result [regexp {((w+) (w+) (w+) (w+))} $text A b c d e]
$A will contain the amount of text matched.
puts $A
the cat sat on the
puts $a
the
puts $b
cat
```

It is also possible to return the elements in a list. The first element is a list of the matched elements and the other elements are the individual sub-matches.

```
Router(tcl)#puts [regexp -inline {((w+) (w+) (w+) (w+) (w+))} $text]
{the cat sat on the mat} the cat sat on the mat.
```

Alternative to regular expression is the split command.

```
set text [exec "show ip int brief"]
foreach line [split $text "n"] {puts $line}
```

```
<table>
<thead>
<tr>
<th>Interface</th>
<th>IP-Address</th>
<th>OK?</th>
<th>Method</th>
<th>Status</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Async1</td>
<td>192.168.3.3</td>
<td>YES</td>
<td>NVRAM</td>
<td>down</td>
<td>down</td>
</tr>
<tr>
<td>GigabitEthernet0/0</td>
<td>10.64.40.10</td>
<td>YES</td>
<td>NVRAM</td>
<td>up</td>
<td>up</td>
</tr>
<tr>
<td>Service-Engine0/0</td>
<td>10.79.3.253</td>
<td>YES</td>
<td>TFTP</td>
<td>up</td>
<td>up</td>
</tr>
<tr>
<td>GigabitEthernet0/1</td>
<td>unassigned</td>
<td>YES</td>
<td>NVRAM</td>
<td>up</td>
<td>up</td>
</tr>
<tr>
<td>GigabitEthernet0/1.1</td>
<td>10.76.1.1</td>
<td>YES</td>
<td>NVRAM</td>
<td>up</td>
<td>up</td>
</tr>
<tr>
<td>GigabitEthernet0/1.2</td>
<td>10.76.0.1</td>
<td>YES</td>
<td>NVRAM</td>
<td>up</td>
<td>up</td>
</tr>
<tr>
<td>GigabitEthernet0/1.4</td>
<td>10.79.3.253</td>
<td>YES</td>
<td>NVRAM</td>
<td>up</td>
<td>up</td>
</tr>
<tr>
<td>GigabitEthernet0/1.9</td>
<td>192.168.1.63</td>
<td>YES</td>
<td>NVRAM</td>
<td>up</td>
<td>up</td>
</tr>
</tbody>
</table>
```

Elements of the list can be accessed using lindex. Element zero is the interface name.

```
Router(tcl)#foreach line [split $text "n"] {puts [lindex $line 0]}
```

```
Interface
Async1
GigabitEthernet0/0
Service-Engine0/0
GigabitEthernet0/1
GigabitEthernet0/1.1
GigabitEthernet0/1.2
GigabitEthernet0/1.4
GigabitEthernet0/1.9
```
Summary

I hope you have enjoyed this introduction. Please read the references and try the various facilities yourself.