Cisco IOS Shell

First Published: March 25, 2011
Last Updated: March 30, 2011

The Cisco IOS Shell (IOS.sh) feature provides shell scripting capability to the Cisco IOS command-line-interface (CLI) environment. Cisco IOS.sh enhances the process of controlling and configuring an IOS router using the CLI by including, variable substitution, paths, conditional statements, loops, pipes, and so on to enhance the user experience of Cisco IOS CLI users.

Finding Feature Information

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.sh" section on page 15.

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

Contents

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

Prerequisites for Cisco IOS.sh

- Cisco IOS Release 15.1(4)M, 15.1(2)S, and later releases.
- Cisco IOS.sh must be configured and enabled to use the Cisco IOS.sh features and functions on your router.

Restrictions for Cisco IOS.sh

If Cisco IOS.sh is not enabled, the Cisco IOS.sh features and functions are not available on your router.

Information About Cisco IOS.sh

The Cisco IOS.sh feature is designed to be familiar to users. This is done by implementing a control language that is similar in many ways to various open source CLI interfaces. A system administrator with a UNIX background can easily understand and use the new Cisco IOS.sh features, and an experienced Cisco IOS CLI user can easily learn and use these features as enhancements. Cisco IOS.sh is also mostly backward compatible with the existing Cisco IOS CLI, with a few obvious exceptions. This means that CLI commands that are entered on the router will probably continue to work as before. However, users should be aware that some commands may need to be invoked differently if users want to take advantage of the Cisco IOS.sh.

How to Enable Cisco IOS.sh

You can turn Cisco IOS.sh processing on the terminal by using the **terminal shell** command. However shell processing feature is only on while the terminal is running. Once the terminal is turned off, shell processing is off. When the **terminal shell** command is used, shell processing is not visible in the running configuration because it is only on the terminal level and is not in the configuration level. It is convenient to use the **terminal shell** command at the terminal level to quickly access the Cisco IOS.sh man commands. To enable shell processing and access all its functions, it is recommended that you use the **shell processing full** command.

The Cisco IOS.sh feature can be enabled in the following ways:

- Terminal Option, page 2
- Configuration Option, page 3

Terminal Option

Users have options to enable and disable Cisco IOS.sh environment in a given terminal.

To enable shell processing on the terminal and to enable trace enter the following:

```
Router> enable
Router# terminal shell
Router# terminal shell trace
```

To disable shell processing on the terminal enter the following:

```
Router# terminal no shell
```

When you disable the shell environment, it does not destroy anything. To enable shell and turn Cisco IOS.sh back on, enter the following:

```
Router# terminal shell
Router# terminal shell trace
```

Configuration Option

Users have options to enable and disable the Cisco IOS.sh environment in the global configuration mode using the **shell processing full** command. This is the recommended option. The shell processing command without the **full** keyword enables the default behavior of Cisco IOS.sh to be avialble. But to enable shell processing and access all its functions, use the **shell processing full** command.

```
Router> enable
Router# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)# shell processing full
```

To disable shell processing, use the no form of the command:

```
Router(config)# no shell processing
```

To enable shell and turn Cisco IOS.sh back on, enter the following:

```
Router(config)# shell processing full
```

Using Cisco IOS.sh

The Cisco IOS.sh feature is integrated into the existing IOS CLI environment. This feature must be enabled either in configuration, or using a terminal command before being able to access all its functions, see "How to Enable Cisco IOS.sh" section on page 2. See the "Prerequisites for Cisco IOS.sh" procedure on page -2 section for the required release information.

Note Some releases may have a limited version of the commands described in this document.

After the Cisco IOS.sh is enabled, users can perform the following tasks:

- Defining and using environment variables
- Using control constructs to automate repetitive tasks
- Creating and using Cisco IOS.sh functions
- Using a new set of built-in function, that provide various text processing facilities.
- Using extended pipelines to use the output of one command as input for another one.
- Evaluating logical and arithmetic expressions for tests and variable setting.
- Using online manual pages describing these changes.

Each of these features are described, with examples, in the following sections.

Cisco IOS.sh Variables

An Cisco IOS.sh variable is a way to simplify typing tasks and use long text or functions in a short symbol. Cisco IOS.sh variables can be created in EXEC mode. There cannot be any spaces between the variable name and the equal sign.

Creating a Variable

To create a variable name, use NAME=value with no space:

```
Router# VAR1=value1
Router# VAR2=value2
```

To retrieve the configuration state of the Cisco IOS.sh variables:

```
Router# show shell environment
VAR1=value1
VAR2=value2
```

To remove a variable, simply assign an empty value to the variable:

```
Router# VAR1=
```

Cisco IOS.sh variables are associated with a user's login session, so the Cisco IOS.sh variable set by one user is not visible to others.

Variable Substitution in CLI

A variable reference is done by using the "$" character in front of the name of the variable:

```
Router# abc=123
Router# echo $abc
123
```

Note There is no "shell processing" in startup-config and running-config.

If a CLI must use the "$" character, the usage must be enclosed in single quotes, or prefixed by a back slash character (\$variable):

```
Router# echo 'Here is a string with $xyz in it'
Here is a string with $xyz in it
```

In addition to user-defined variables, there are several other variables that are used to control the Cisco IOS.sh environment, and to get the result of commands executed by the Cisco IOS.sh. The following are some of the other variables:

- PATH
- ?
- prc_change_mode
- prc_change_type
- prc_error_code
- prc_failure_type
- prc_ha_sync

Path

Cisco IOS.sh supports an environment variable called PATH. It contains a percent character (%) separated list of names and directories, which tell the Cisco IOS.sh where to look when executing commands.

Elements that can be included in the PATH variable are:

- CLI
- User-defined functions
- Built-in functions
- User-defined file system directories

The order of the elements on the PATH variable indicates the order of places that the Cisco IOS.sh looks for commands. For example:

```
"PATH=%CLI%;%Userfunctions%;%Builtins%;tmpsys:/;disk0:".
```

In this case, the order of the command lookup is in the following order:

1. If the entered command matches an IOS CLI, then the matched IOS CLI will be invoked.
2. If the entered command matches a user-defined Cisco IOS.sh function, then this user-defined Cisco IOS.sh function will be invoked.
3. If the entered command matches a built-in function, then this built-in function will be invoked.
4. If the entered command matches a script in “tmpsys:lib/tcl” directory (is a virtual file system mapped to a part of the IOS image), the script will be invoked.
5. If the entered command is found in disk0:, the script will be executed.

PRC Variables

There are five built-in variables that are set on command completion. These variables indicate the status of the command that was just executed.

- prc_change_mode
- prc_change_type
- prc_error_code
- prc_failure_type
- prc_ha_sync

The PRC variables contain the value of the PRC, as documented by IOS CLI documentation. In addition, the “$?” variable contains zero (0) on successful execution, and non-zero on failure. The use of the question mark is a historical precedent. Unfortunately, a question mark cannot be typed at the IOS prompt unless it is preceded by the IOS escape character 'Ctrl-V'.

Cisco IOS.sh Control Constructs

Control constructs are used to conditionally or iteratively execute sets of CLIs. There are three reserved keywords that invoke control constructs

- if
- for

- while

“if” Statements

An “if” statement is used to conditionally execute sets of statements. The “if” loop syntax is as follows:

```
if <test>
  statements
elif <test>
  statements
else
  statements
fi
```

The “elif” and “else” commands are optional, and there can be multiple “elif” statements. The “end” for “if” statements is “fi” which is “if” backwards. This is done for historical reasons.

There are a set of conditional tests available for tests, which are prefixed by either “[[“ or “((“. (See conditional testing below).

In addition, built-in functions or CLI commands can be used as the test, and the PRC of the command will be used to determine TRUE or FALSE.

“for” Statements

For loop syntax:

A “for” loop is a control construct that loops over a given set of values, and executes a set of statements with a variable set to each of the values in turn. Here is a simple example.

```
for x in 1 2 3 4 5
do
    echo $x
done
```

The loop will print out each of the values 1,2,3,4,5 in turn.

“while” Statements

A 'while" statement is used the same test syntax as an 'if' statement, and executes the statements in the following block until the test returns FALSE. An example is

```
let x=0
while (( x++ < 10 )); do
  echo $x
done
```

This will print out the numbers from 1 to 10

Loop Modifiers

Users can exit a 'while' or 'for' loop early by using a 'break' statement. Here is an example.

```
n=0
while true; do
    let n++
     if [[ $n -le 10 ]]; then
        echo $n
    else
break;
    fi
done
```

In addition to 'break', the 'continue' statement can be used to stop executing statements in the loop, and restart the loop. A 'while' statement that is interrupted by a 'continue' will simply stop executing statements in the body of the loop and start the 'while' statement over again. A 'for' statement that is interrupted by a continue will also start over, except that it will assign the next element in the list of values before doing so.

```
for i in 0 1 2 3 4 5 6
do
    statements1      #Executed for all values of ''I'', up to a disaster-condition if any.
    statements2
    if interface ethernet$i/0
    then
      statements
    else
      continue
    fi
    statements3
done
```

Note "Continue" and 'break' statements have no effect outside of a loop.

Cisco IOS.sh Conditional Expressions

Cisco IOS.sh evaluates conditional expressions (with each operator and operand as a separate argument) as part of 'if/elif' and 'while' statements. The result of these are interpreted by those commands in a somewhat counter-intuitive way:

0 for True

1 for False

The return value of any command can be displayed using the Cisco IOS.sh variable '$?'.

Note In order to enter "?" on the command line without invoking online help, you will need to prefix the '?' with Ctrl-V.

The two built-in test facilities are '[[' and '(('. The first is for evaluating logical tests, and the second is for evaluating numerical expression tests.

Logical Tests

Logical testing is invoked in 'if' or 'while' statements using the

'[[' operator. Here is an example.

```
if [[ "abc" = $foo ]]; then
   echo '$foo is equal to abc'
fi
```

If the variable '$foo' contains the string 'abc', then it will print the message.

There are quite a few different logical tests that can be used within

the '[[' to ']]'.

The following tests can be used on files in the file system:

-a <filename>	- return TRUE if file exists
-e <filename>	- same as -a
-d <dirname>	- TRUE if arg is a directory
-f <filename>	- TRUE if file is a regular file
-x <filename>	- TRUE if file is executable
-r <filename>	- TRUE if file is readable by user
-w <filename>	- TRUE if file is writable by user

The following tests can be used on strings:

-z <string>	- TRUE if string is 0 length
-n <string>	- TRUE if string is not 0 length
<string> <= <string>	- TRUE if first is lexographically less than second
<string> == <regex>	- TRUE if regex matches ALL of string
<string> != <regex>	- TRUE if regex does not match string
<string> =~ <regex>	- TRUE if regex matches anything in string vars $v0-$v9 are set to contain matches that are enclosed in parenthesis.
<expr> && <expr>	- TRUE if both expressions are non-null
<expr> || <expr>	- TRUE if either expression is non-null

Additionally, there are some numerical tests that can be used here. These are intended to be used with variables that contain integers (floating point numbers are not supported), and will give unexpected results if given strings instead of numbers.

<varname>++	- postincrement a variable
<varname>--	- postdecrement a variable

<expr> * <expr>	- TRUE if multiplication result is non-zero
<expr> / <expr>	- TRUE if division is non-zero, and denominator is non-zero
<expr> + <expr>	- TRUE if sum is non-zero
<expr> - <expr>	- TRUE if difference is non-zero
<nexpr> -lt <nexpr>	- TRUE if first numeric expr is < second
<nexpr> -le <nexpr>	- TRUE if first numeric expr is <= second

<nexpr> -eq <nexpr>	- TRUE if numeric expressions are equal
<nexpr> -ge <nexpr>	- TRUE if first numeric expr is >= second
<nexpr> -gt <nexpr>	- TRUE if first numeric expr is > second

Arithmetic Tests

The '((' can be used in tests. It has a different set of possibilities that can be performed than the '[[ ]]' syntax.

In the '(( ))' syntax, variables can be referenced without the '$' prefix. Here is the list of tests that are available. All of the following expressions expect numeric values. If the result is non-zero, it will evaluate to TRUE.

<varname>	- evaluates to the variable value
<varname>++	- postincrement
<varname>--	- postdecrement
!<expr>	- logical NOT
~<expr>	- bitwise negation
<expr> * <expr>	- multiplication
<expr> / <expr>	- division
<expr> + <expr>	- addition
<expr> - <expr>	- subtraction
<expr> < <expr>	- comparison. evaluates to 0 or 1
<expr> <= <expr>	- comparison. evaluates to 0 or 1
<expr> == <expr>	- comparison. evaluates to 0 or 1
<expr> >= <expr>	- comparison. evaluates to 0 or 1
<expr> > <expr>	- comparison. evaluates to 0 or 1
<expr> & <expr>	- bitwise AND
<expr> | <expr>	- bitwise OR
<expr> ^ <expr>	- bitwise XOR
<expr> << <expr>	- left shift
<expr> >> <expr>	- right shift
<expr> && <expr>	- logical AND
<expr> || <expr>	- logical OR
<varname> = <expr>	- assignment
<varname> += <expr>	- assign var + expr to var
<varname> -= <expr>	- assign var - expr to var
<varname> *= <expr>	- assign var * expr to var
<varname> /= <expr>	- assign var / expr to var
<varname> <<= <expr>	- left shift value of var by expr bits, assign to var
<varname> >>= <expr>	- right shift value of var by expr bits, assign to var

(<expr>)   - value of expression

Parenthesis can be used to group expressions. However, parenthesis must be quoted or escaped in order to be used

Back Quotation Mark Support

Cisco IOS.sh has a syntax that can be used to enable the output of a command to be used as Cisco IOS.sh input. Any text between backquote characters will be interpreted as a command, and inserted as if it was typed in at the console. Here is an example:

```
Router# for xx in `interface Ethernet`; do echo $xx; done
Ethernet0/0
Ethernet0/1
Ethernet0/2
Ethernet0/3
Ethernet1/0
Ethernet1/1
Ethernet1/2
Ethernet1/3
r100-isis#
```

The 'interface' command lists the set of interfaces that match the supplied pattern. When we put it in back quotes, the matching interfaces are inserted into the command line. The result is seen by the Cisco IOS.sh as follows.

```
Router# for xx in Ethernet0/0 Ethernet0/1 Ethernet0/2 (etc...)
```

Pipes and Redirections

The IOS CLI has a facility to 'pipe' text from a command into a set of programs that can filter or redirect the output. The Cisco IOS.sh has expanded this facility to support more than one 'pipe' command on a line of input. The commands are executed concurrently, and print whatever the final command in the 'pipeline' prints.

```
Router# show version | grep '^C'
Cisco IOS Software, 7200 Software (C7200-P-M), Experimental Version 12.2(20090611:002213)
Copyright (c) 1986-2010 by Cisco Systems, Inc.
Compiled Tue 16-Jun-10 14:23 by janedoe
Cisco 7204VXR (NPE225) processor (revision A) with 114688K/16384K bytes of memory.
Configuration register is 0x0
```

This example prints all lines that begin with uppercase C.

Cisco IOS.sh Functions

IOS.sh functions enable you to define a group of commands that can be parameterized by command line arguments. The arguments passed to the IOS.sh function will be substituted for $1, $2, and so on within the body of the function

A simple example would be a function to set a description for an interface by using the Cisco IOS.sh **function** command:

```
Router# function set_interface_description () {
    configure terminal
```

```
        interface $1
            description $2
    end
}
```

This command can then be executed on the command line as follows:

```
Router# set_interface_description 'eth0/0' 'this is a description'
```

This will change eth0/0's description to 'this is a description'

Another example is enabling configuration logging:

```
Router# function config_logging() {
configure terminal
 archive
  logging $1
  record rc
end
}
```

Then, execute the following.

```
Router# config_logging enable
```

When the **config_logging** Cisco IOS.sh command is executed with the parameter enable, logging of configuration is enabled. Any of the facilities of Cisco IOS.sh can be used in a shell function.

Built-in Cisco IOS.sh Functions

The built-in Cisco IOS.sh functions are system-defined shell functions, and are supplied along with the software image. These built-in functions supply various utilities that can be used along with Cisco IOS.sh to manipulate information. The following is the set of Cisco IOS.sh built-in functions that are currently available.

Table 1 Built-in Cisco IOS.sh Functions

Built-in Function	Description
cat	Output data from a pipe or file to the terminal
cut	Edit piped output
echo	Echo arguments to the terminal
false	Return false in while or if expressions, and set the result
fetch	Return values from the configuration database
for	Cisco IOS.sh for loops
grep	Search for regular expressions in piped output or files
head	Print the first lines in the input
interface	Print interfaces that match the argument
let	Evaluate a numeric expression, and set the result
man	Print information for built-ins
more	Page piped output to the terminal
nl	Number the lines in the input

Table 1 Built-in Cisco IOS.sh Functions (continued)

Built-in Function	Description
null	Ignore the input
printf	Output formatted data to the terminal
read	Read input into variables
set_oper	Set operational values
sleep	Pause execution of the terminal
sort	Sort the input
tail	Print the tail of the input
true	Return true in while or if expressions, and set the result
uname	Print system information
wc	Count lines, words, and chars

To access the built-in Cisco IOS.sh functions enter the **man** EXEC command in the following order:

```
Router# enable
Router# configure terminal
Router(config)# shell processing full
Router(config)# exit
Router# man
((              evaluate a numeric test expression
IOS.sh          The IOS shell
[[              evaluate a logical test expression
cat             output data from a pipe or file to the terminal
compatibility   compatibility information on IOS.sh
control         control constructs in IOS.sh
cut             edit piped output
echo            echo arguments to the terminal
expressions     usage of expressions in IOS.sh
false           return false in while or if expressions, and set the result
fetch           return values from the configuration database
for             IOS.sh for loops
functions       IOS.sh function facility
grep            search for regular expressions in piped output or files
head            print the first lines in the input
if-else         IOS.sh if command
interface       print interfaces that match the argument
let             evaluate a numeric expression, and set the result
man             print information for builtins
more            page piped output to the terminal
nl              number the lines in the input
null            ignore the input
printf          output formatted data to the terminal
quoting         IOS.sh quoting facility
read            read input into variables
scripting       how to script the IOS CLI
set_oper        set operational values
sleep           pause execution of the terminal
sort            sort the input
tail            print the tail of the input
true            return true in while or if expressions, and set the result
uname           print system information
variables       usage of variables in IOS.sh
wc              count lines, words, and chars
while           iterate while an expression is TRUE
```

To access individual built-in commands, enter the command from the above list preceded by the **man** EXEC command, for example, **man <built-in command name>**. The details of the **true** built-in command is displayed below:

```
Router# man true
NAME
    true - return true

SYNOPSIS
    true

DESCRIPTION
    The 'true' program always returns true.
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	*Cisco IOS Master Commands List, All Releases*

Standards

Standard	Title
None	—

MIBs

MIB	MIBs Link
• None	To locate and download MIBs for selected platforms, Cisco software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
None	—

Technical Assistance

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

Feature Information for Cisco IOS.sh

Table 2 lists the release history for this feature.

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which 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 2 lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Table 2 Feature Information for Cisco IOS.sh

Feature Name	Releases	Feature Information
Cisco IOS Shell	15.1(4)M 15.1(2)S	The Cisco IOS.sh feature enhances the process of controlling and configuring an IOS router using the CLI by including, variable substitution, paths, conditional statements, loops, pipes, and so on to enhance the user experience of Cisco IOS CLI users. The following sections provide information about this feature: • Prerequisites for Cisco IOS.sh, page 2 • Information About Cisco IOS.sh, page 2 • How to Enable Cisco IOS.sh, page 2 • Using Cisco IOS.sh, page 3 The following commands were introduced: **shell environment load**, **shell environment save**, **shell init**, **shell processing**, **show shell environment**, **show shell functions, terminal shell**.