Yuma yangcli Manual

YANG-Based Unified Modular Automation Tools

NETCONF Over SSH Client

Version 2.0

Last Updated: July 20, 2011

Table Of Contents

Yuma yangcli Manual

1 Preface

1.1 Legal Statements

1.2 Additional Resources

This document assumes you have successfully set up the software as described in the printed document:

Yuma Installation Guide

Other documentation includes:

Yuma Quickstart Guide

Yuma User Manual

Yuma netconfd Manual

Yuma yangdiff Manual

Yuma yangdump Manual

Yuma Developer Manual

To obtain additional support you may join the yuma-users group on sourceforge.net and send email to this e-mail address:

yuma-users@lists.sourceforge.net

The SourceForge.net Support Page for Yuma can be found at this WEB page:

http://sourceforge.net/projects/yuma/support

There are several sources of free information and tools for use with YANG and/or NETCONF.

The following section lists the resources available at this time.

1.2.1 WEB Sites

- **Netconf Central**
  - http://www.netconfcentral.org/
  - Yuma Home Page
    - Free information on NETCONF and YANG, tutorials, on-line YANG module validation and documentation database
- **Yuma SourceFource OpenSource Project**
  - http://sourceforge.net/projects/yuma/

    - Download Yuma source and binaries; project forums and help
- **Yang Central**
    - http://www.yang-central.org
    - Free information and tutorials on YANG, free YANG tools for download
- **NETCONF Working Group Wiki Page**
    - http://trac.tools.ietf.org/wg/netconf/trac/wiki
    - Free information on NETCONF standardization activities and NETCONF implementations
- **NETCONF WG Status Page**
    - http://tools.ietf.org/wg/netconf/
    - IETF Internet draft status for NETCONF documents
- **libsmi Home Page**
    - http://www.ibr.cs.tu-bs.de/projects/libsmi/
    - Free tools such as smidump, to convert SMIv2 to YANG

1.2.2 Mailing Lists

- **NETCONF Working Group**
    - http://www.ietf.org/html.charters/netconf-charter.html
    - Technical issues related to the NETCONF protocol are discussed on the NETCONF WG mailing list. Refer to the instructions on the WEB page for joining the mailing list.
- **NETMOD Working Group**
    - http://www.ietf.org/html.charters/netmod-charter.html
    - Technical issues related to the YANG language and YANG data types are discussed on the NETMOD WG mailing list. Refer to the instructions on the WEB page for joining the mailing list.

1.3 Conventions Used in this Document

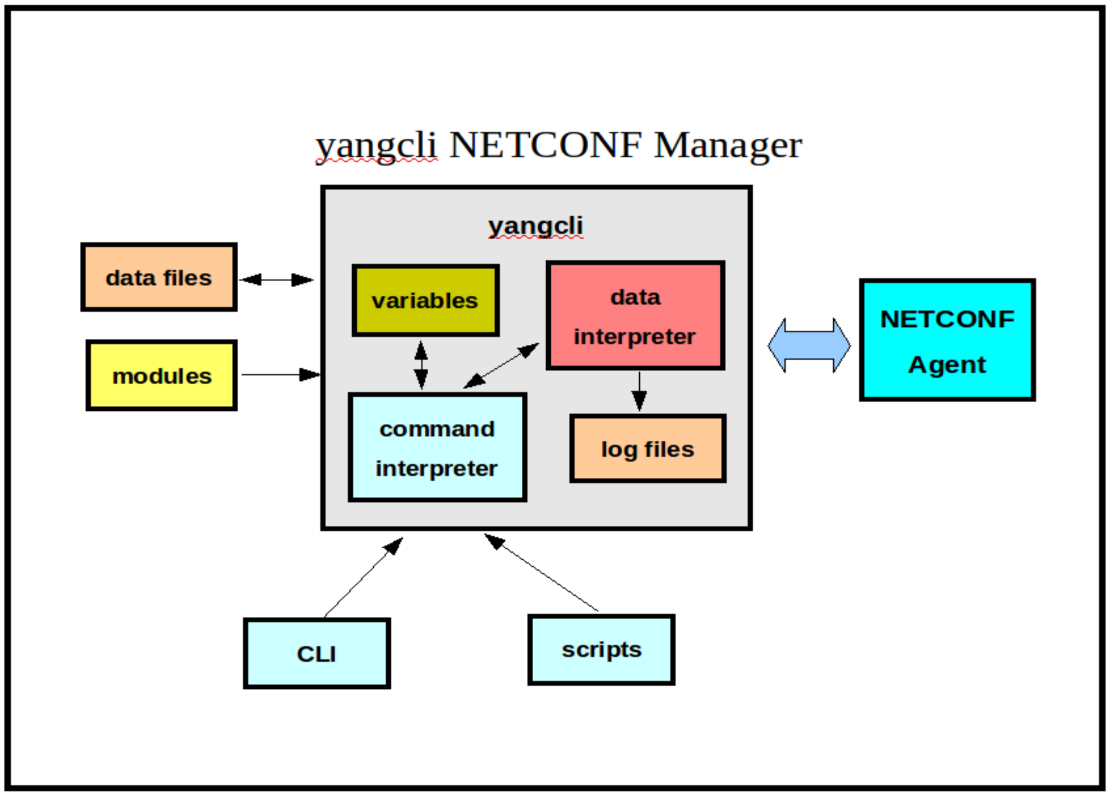
The following formatting conventions are used throughout this document:

Documentation Conventions

Convention	Description
--foo	CLI parameter foo
<foo>	XML parameter foo
foo	**yangcli** command or parameter
$FOO	Environment variable FOO
$$foo	**yangcli** global variable foo
`some text`	Example command or PDU
some text	Plain text

2 yangcli User Guide

Program Components



2.1 Introduction

The **yangcli** program is a NETCONF over SSH client application. It is driven directly by YANG modules, and provides a simple but powerful application interface for any YANG file. There is no configuration required at all to use any YANG file, although there are some YANG extensions that will be utilized if present.

2.1.1 Features

The **yangcli** client has the following features:

- Automatic support for all NETCONF protocol operations, including several 'short-hand' commands for the most common operations, like <edit-config> and <get-config>.
- Automated database locking, unlocking and error cleanup, using the high-level **get-locks** and **release-locks** commands.
- Automatic, standards-based, server schema synchronization, using the YANG module capability URI information in the <hello> PDU, and the <get-schema> operation:

    - For each session, the exact view of the server schema definition tree is created, based on the module capability:
        - module namespace
        - module name
        - module revision date
        - enabled features
        - names of any modules that contain deviations for this module
    - The help text and parameter validation for each session will be tailored to the capabilities advertised by the server.
    - Parses each potential matching YANG file to make sure the module name, revision date, and namespace URI value are all exactly the same as the values in the module capability URI.
- Understands all NETCONF protocol capabilities, and complex hard-wired logic simplifies protocol usage, and allows high-level commands to pick appropriate defaults for many RPC operation parameters.
- Load any YANG module at boot-time or run-time and start using it immediately.
- Full concurrent support for multiple revisions of the same module.
- Supports NETCONF notifications, including :interleave capability.
- Full XPath 1.0 and subtree filtering support.
- Automatic support for all YANG language mechanisms, including extensions.
- Any YANG **<rpc>** operation is automatically available as a **yangcli** command.
- Uses YANG files directly as input, so no pre-processing or configuration needed to use a new module.
- Can be called from **unix** scripts in 'batch-mode' to automatically establish a NETCONF session, issue a command or invoke a yangcli script, close the session, and return the results.
- Extensive interactive shell environment, including user variables, file variables, smart parameter set completion, and a simple scripting environment for automatic operation.
- Automatic, context-sensitive (tab key) command-line completion.
- Full support for XPath, instance-identifier, leafref, and identityref parameters.
- Automatic, context-sensitive help system, based completely on YANG files and using the exact modules supported by the current NETCONF session, if connected.
- Full, customizable command line editing, using **emacs** by default, but **vi** or a custom set of keystroke bindings are also supported.
- Command line history and command recall.
- Store and recall command line history files for later use.
- Automatic NETCONF session management, including support for all YANG extensions to the <capability> element.
- Automatic recognition and support for all NETCONF 'capability' related operations.
- Automatic support for all YANG additions to the NETCONF protocol, such as the **insert** operation
- Unlimited nested scripts with up to 10 parameters each can automate testing and other management tasks

2.1.2 Starting yangcli

The current working directory in use when **yangcli** is invoked is important. It is most convenient to run **yangcli** from a work directory, rather than the installation directory or within the module library.

The **yangcli** program can be invoked several ways:

- To get the current version and exit:

```
yangcli --version
```

- To get program help and exit:

```
yangcli --help
yangcli --help --brief
yangcli --help --full
```

- To start an interactive session with the default parameters:

```
yangcli
```

- To start an interactive session with a new log file:

```
yangcli --logfile=mylogfile
```

- To start an interactive session and append to an existing log file:

```
yangcli --logfile=mylogfile --log-append
```

- To get parameters from a configuration file:

```
yangcli --config=~/yangcli.conf
```

- To begin to connect to a server upon startup, provide the **--server** parameter. The **connect** command will be started upon startup and the user will be prompted to enter the rest of the mandatory parameters to the **connect** command.

```
yangcli server=myserver.example.com
```

- To connect to a server and automatically connect without any interactive interruption, enter the **--server**, **--user**, and **--password** parameters. A session startup will be attempted right away, using these parameters. Any optional parameters for the **connect** command (**--port** or **--timeout**) may be entered as well. All parameters can be entered from a config file, and/or the command line.

```
yangcli --server=myserver.example.com \
        --user=andy --password=yangrocks
```

- To automatically connect to a server, run a script in non-interactive mode, and then remain connected to the server, add the **--run-script** parameter to the connection parameters. The **--runpath** parameter can also be entered, if needed.

```
yangcli --server=myserver.example.com \
        --user=andy --password=yangrocks \
        --run-script=mytestscript
```

- To automatically connect to a server, run a script in non-interactive mode, and then exit the program, add the **--batch-mode** and **--run-script** parameters to the connection parameters. The **--runpath** parameter can also be entered, if needed.

```
yangcli --server=myserver.example.com \
        --user=andy --password=yangrocks \
        --run-script=mytestscript --batch-mode
```

- To automatically connect to a server, and run a single command instead of a script, and then exit, use the **--run-command** parameter instead of the **--run-script** parameter. The **--batch-mode** parameter can be left out to remain in the current session (in interactive mode) after the command is invoked.

```
yangcli --server=myserver.example.com \
        --user=andy --password=yangrocks \
        --batch-mode --run-command="sget /system"
```

2.1.3 Stopping yangcli

To terminate the **yangcli** program, use the **quit** command.

The control-C character sequence will also cause the program to be terminated.

2.1.4 Statements

The **yangcli** script interpreter accepts several types of statements:

yangcli Statements

type	description	example
command	invoke a local command and/or send an <rpc> to the server	sget /system
variable assignment	set a user variable to some value	$system = sget /system
file assignment	set the contents of a file to some value	@save.txt = $system

variable deletion	delete a user variable or clear a system variable	$system =

A command can be as simple like 'get' or complex, like 'edit-config'.

A variable assignment sets the specified user or system variable to the right hand side of the expression. An expression has many forms, including the result from a local command or a remote NETCONF operation.

If a string that matches a command is used as the assignment value, then it must be entered in quotes (single or double). For example, the **$$default-operation** system configuration variable accepts enumeration values which also match RPC commands:

```
yangcli> $$default-operation = 'merge'
```

A file assignment is essentially the same as a variable assignment, except the right hand side of the expression is saved in the specified file, instead of a user variable.

To delete a user variable, leave the right hand side of the expression empty (or just whitespace).

Note: More statement types will be available in the next release.

2.1.5 Commands

The yangcli program has several built-in commands, defined in **yangcli.yang**, **yuma-netconf.yang,** and **notifications.yang**.

The YANG **rpc** statement is used to define the syntax and behavior of each command.

There are 2 types of yangcli commands:

- **local**: the command is executed within the yangcli application, and can be invoked at any time.
- **remote**: the command is executed on the remote server, and is only available when a NETCONF session is active. Any YANG **rpc** statement that **yangcli** does not recognize as a local command is treated as a remote command available on the server.

Local Commands

command	description
cd	Change the current working directory
connect	Connect to a server and start a NETCONF session
elif	Start an 'else-if' conditional block
else	Start an 'else' conditional block
end	End an 'if' or 'while' block
eval	Evaluate an XPath expression
eventlog	View or clear the notification event log
fill	Fill a user variable
help	Get context-sensitive help
history	Manage the command history buffer

if	Start an 'if' conditional block
list	List modules, objects, or other properties of the session
log-debug	Log a debug message
log-error	Log an error message
log-info	Log an info message
log-warn	Log a warning message
mgrload	Load a YANG file into the client only
pwd	Print the current working directoty
quit	Exit the program
recall	Recall a line from the command history buffer
run	Run a script
show	Show variables and objects currently available
start-timer	Start a script timer
stop-timer	Stop a script timer and display the elapsed time
while	Start a 'while' conditional loop block

The following table shows the standard NETCONF protocol operations that are directly available for use, depending on the capabilities of the server.

Standard NETCONF Commands

command	description
cancel-commit	Cancel the current confirmed commit procedure
close-session	Close the current NETCONF session
commit	Make the candidate database be the running config
copy-config	Copy an entire NETCONF database
create-subscription	Start receiving NETCONF notifications
delete-config	Delete an entire NETCONF database
discard-changes	Discard any edits in the candidate database
edit-config	Alteration of the target database
get	Filtered retrieval of state data and running config
get-config	Filtered retrieval of any NETCONF database
get-schema	Get a data model definition file from the server
kill-session	Force close another NETCONF session
lock	Lock a NETCONF database that is currently unlocked
unlock	Unlock a NETCONF database that is currently locked
validate	Validate the contents of a NETCONF database

The following **yangcli** commands are available for simplified access to standard NETCONF operations

Custom NETCONF Commands

command	description
create	Invoke an <edit-config> create operation
delete	Invoke an <edit-config> delete operation
get-locks	Lock all the databases with the <lock> operation
insert	Invoke an <edit-config> YANG insert operation
merge	Invoke an <edit-config> merge operation
release-locks	Unlock all the locked databases with the <unlock> operation
remove	Invoke an <edit-config> remove operation
replace	Invoke an <edit-config> replace operation
save	Save the current edits on the server in NV-storage
sget	Invoke a <get> operation with a subtree filter
sget-config	Invoke a <get-config> operation with a subtree filter
xget	Invoke a <get> operation with an XPath filter
xget-config	Invoke a <get-config> operation with an XPath filter

The following table shows the extended NETCONF protocol operations that are available on the **netconfd** server only.

Extended netconfd Commands

command	description
get-my-session	Retrieve session customization parameters
load	Load a module into the server.
no-op	No operation.
restart	Restart the server.
set-log-level	Set the server logging verbosity level.
set-my-session	Set session customization parameters.
shutdown	Shutdown the server.

2.1.6 Variables

The **yangcli** program utilizes several types of user-accessible variables. These variables can be listed with the '**show vars**' command and other commands as well.

A variable reference consists of 1 or 2 dollar signs ('$'), followed immediately by a valid identifier string (e.g., **$$global-log** or **$local-log**).

Variables can be 1 or more characters in length, and follow the YANG rules for identifier names. The first character must be a letter, 'A' to 'Z', or 'a' to 'z'. The 2nd to last characters can be a letter 'A' to 'Z', or 'a' to 'z', number ('0' to '9'), an underscore ('_'), a dash ('-'), or a period ('.') character.

There are 4 categories of parameters supported:

1. Read-only system variables
2. Read-write system variables
3. Read-write global user variables
4. Read-write local user variables

It is an error if a variable is referenced (in the right-hand-side of a statement) that does not exist.

The first 3 types are **global variables**, which means that they are available to all run-levels of all scripts. The last type, called a **local variable,** is only visible to the current run-level of the current script (or interactive shell). Refer to the following section for more details on run levels.

Variable Syntax

syntax	description	example
$$<variable-name>	Left hand side: set the global variable to some value	$$saved_get = get
$$<variable-name>	Right hand side: access the value of a global variable	fill --target=\ $$mytarget
$<variable-name>	Left hand side: set the local variable to some value	$myloglevel = \ $$log-level
$<variable-name>	Right hand side: access the value of any variable with this name (try local, then global)	$myuser = $user

The following table shows the **unix** environment variables that are available as read-only global variables in **yangcli**. These variables are set once when the program is started, and cannot be used in the the left hand side of an assignment statement.

Read-only system variables

variable	description
$$HOME	the **HOME** environment variable
$$HOSTNAME	the **HOST** or **HOSTNAME** environment variable
$$LANG	the **LANG** environment variable
$$PWD	the **PWD** environment variable, when **yangcli** was invoked
$$SHELL	the **SHELL** environment variable
$$USER	the **USER** environment variable

$$YUMA_DATAPATH	the **YUMA_DATAPATH** environment variable
$$YUMA_HOME	the **YUMA_HOME** environment variable
$$YUMA_MODPATH	the **YUMA_MODPATH** environment variable
$$YUMA_RUNPATH	the **YUMA_RUNPATH** environment variable

The following table shows the CLI configuration parameters that can be read or changed (but not deleted). If a particular parameter was not set during program invocation, then the associated variable will contain the empty string.

Read-write system variables

variable	description
$$alt-names	the **–alt-names** configuration parameter
$$autocomp	the -**-autocomp** configuration parameter
$$autoload	the **--autoload** configuration parameter
$$baddata	the **--baddata** configuration parameter
$$default-module	the **--default-module** configuration parameter
$$default-operation	the <default-operation> parameter for the <edit-config> operation
$$display-mode	the **--display-mode** configuration parameter
$$error-option	the default <error-option> parameter for the <edit-config> operation
$$fixorder	the **--fixorder** configuration parameter
$$indent	the **–indent** configuration parameter
$$log-level	the **--log-level** configuration parameter
$$match-names	the **–match-names** configuration parameter
$$optional	the **--optional** configuration parameter
$$server	the **--server** configuration parameter
$$test-option	the <test-option> parameter for the <edit-config> operation
$$time-rpcs	the **–time-rpcs** configuration parameter
$$timeout	the **--timeout** configuration parameter
$$use-xmlheader	the **–use-xmlheader** configuration parameter
$$user	the **--user** configuration parameter
$$with-defaults	the **--with-defaults** configuration parameter

Read-write global user variables

If a unrecognized global variable (e.g., $$foo) is used in the left-hand side of an assignment statement, then a global user variable will be created with that name. If the global user variable already exists, then its value will be overwritten.

Read-write local user variables

If a local variable (e.g., $foo) is used in the left-hand side of an assignment statement, then a local user variable will be created with that name. If the local user variable already exists, then its value will be overwritten. If the variable is created within a script (i.e., run-level greater than zero), then it will be deleted when the script exits.

2.1.7 Files

File contents can be used in **yangcli** statements, similar to user variables.

A file reference consist of the 'at-sign' character ('@') followed immediately by a valid file specification.

```
      @foo.yang = get-schema --identifier=foo --version=””
--format=”YANG”

      mgrload --module=foo
```

If the file extension is “**.yang**”, “**.log**”, “**.txt**”, or “**.text**”, then the value (or command output) will be saved, and yangcli will strip off the outermost XML (if needed) to save the requested file as a pure text file. Otherwise, the file will be saved in XML format. The display mode set by the user can affect XML output. If the display mode i s'xml-nons' then XML without namespace (xmlns) attributes will be generated instead of normal XML.

Note: The **--display-mode** configuration parameter, and **$$display-mode** system variable, only affect the output and display of data in the yangcli program. NETCONF protocol messages are always sent in 'xml' mode.

Files may also be used as parameter replacements, within a command.

```
      $saved_get = get --filter=@filter.xml --with-defaults=explicit
```

It is an error if the file referenced does not exist or cannot be read.

2.1.8 Scripts

Any command can be entered at the interactive shell, or stored in a script file, and invoked with the '**run**' command. Scripts are simply text files, and the extension used does not matter.

There are no formal templates for scripts, like there are for RPC operations, at this time. Instead, positional parameters can be passed to any script.

The parameters named **--P1** to **--P9** allow up to 9 parameters to be passed to any script. Within each script, the numbered parameters '**$1**' to '**$9**' are available, and contain the value that was passed as the corresponding ---Pn” parameter when calling the script.

If a line contains only optional whitespace, followed by the pound sign character '#', then the line is treated as a comment. These lines will be skipped.

If an error occurs during a command within a script, or an <rpc-error> is received from the server during a NETCONF session, then the running script will be canceled, and if this was a nested script, then all parent scripts will also be canceled.

Script Example:

```
run connect --P1=andy --P2==localhost --P3=yangrocks

// connect script
# start a NETCONF session
$user = $1
$server = $2
$password = $3

connect --user=$user --server=$server --password=$password
```

Run Levels

The **run** command can appear in a script.

When **yangcli** starts up, either in interactive mode or in batch mode, the script interpreter is at run level zero. Each time a **run** command is invoked, either at the command line or within a script currently being invoked, a new run level with the next higher value is assigned. Local variables are only visible within the current run level.

A maximum of 512 run levels are supported in **yangcli**.

Scripts can be called recursively, but a stop condition needs to be used to break the recursion (e.g., call the script from within a conditional code block.

Conditional Command Blocks

The 'if', 'elif', 'else', and 'end' commands are used to create an 'if command sequence'.

Any other commands that appear between these commands are part of a conditional command block.

These blocks can be nested. The current conditional state is inherited, so an if command sequence within a false conditional block will not be executed. A block can contain zero or more command lines,

These command work exactly like an 'if' expression within a program language such as Python. Note that indentation is not significant, but it may be used to make scripts more readable.

The 'if' command starts a new if-command sequence. It is followed by a conditional command block. This block ends when an 'elif', 'else', or 'end' command within the same if command block is encountered.

At most, only one conditional code block within the if command sequence will be executed. Once a conditional command block has been exectuted, any remaining blocks will be skipped.

All user and system variables that are available to the current script run level can be used within the XPath expressions for determining the conditional block state (true or false).

Conditional Command Loop Blocks

The 'while' and 'end' commands are used to create an 'while loop sequence'.

Any other commands that appear between these commands are part of a conditional command loop block.

These blocks can be nested. The current conditional state is inherited, so a while loop sequence within a false conditional block will not be executed. A block can contain zero or more command lines,

The loop condition can be a constant expression. The maxloops parameter will prevent infinite looping, and can be utilized to use the while loop sequence as a simple 'for' loop, iterating a specific number of times.

All user and system variables that are available to the current script run level can be used within the XPath expressions for determining the conditional block state (continue or terminate loop).

Sample Script

The following script does not do any useful work.

It is provided to demonstrate some simple constructs.

```
$x = 0
while '$x < 2'
  # this is a comment
  log-info 'start 1'
  $x = eval '$x + 1'
  $y = 0
  while '$y < 4'
    log-info 'start 2'
    $y = eval '$y + 1'
    if "module-loaded('test')"
      log-info 'module test loaded'
    elif '$x > 1'
      log-info 'x>1'
    elif "feature-enabled('test3', 'feature1')"
      log-info 'feature1'
    else
      log-info 'else'
    end
    log-info 'end 2'

  end
  log-info 'end 1'
end
if "feature-enabled('test5', 'feature-foo')"
  log-info 'feature-foo'
  run add-foo-parms
end
```

2.1.9 Configuration Parameter List

The following configuration parameters are used by **yangcli**. Refer to the CLI Reference for more details.

yangcli CLI Parameters

parameter	description
--alt-names	Controls whether alternate names will be checked for UrlPath searches.
--autocomp	Controls whether partial commands are allowed or not.
--autohistory	Controls whether th command line history buffer will be automatically loaded at startup and saved on exit.
--autoload	Controls whether modules used by the server will be loaded automatically, as needed.
--bad-data	Controls how bad data about to be sent to the server is handled.
--batch-mode	Indicates the interactive shell should not be used.
--config	Specifies the configuration file to use for parameters.
--datapath	Sets the data file search path.
--default-module	Specifies the default module to use to resolve identifiers.
--deviation	Species one or more YANG modules to load as deviations.`
--display-mode	Specifies how values should be displayed.
--echo-replies	Controls whether RPC replies will be displayed in the log output, if log-level >= 'info'
--feature-disable	Leaf list of features to disable
--feature-enable	Specifies a feature that should be enabled
--feature-enable-default	Specifies if a feature should be enabled or disabled by default
--fixorder	Controls whether PDUs are changed to canonical order before sending them to the server.
--force-target	Controls whether the candidate or running configuration datastore will be used as the default edit target, when both are supported by the server.
--help	Get program help.
--help-mode	Adjust the help output (--brief or --full).
--indent	Specifies the indent count to use when writing data.
--log	Specifies the log file to use instead of STDOUT.
--log-append	Controls whether a log file will be reused or overwritten.
--log-level	Controls the verbosity of logging messages.
--match-names	Match mode to use for UrlPath searches
--modpath	Sets the module search path.
--module	Specifies one or more YANG modules to load upon startup.
--ncport	Specifies the NETCONF server port number to use in

	the **connect** command.
--password	Specifies the password to use in the **connect** command.
--private-key	Contains the file path specification for the file containing the client-side private key.
--protocols	Controls which NETCONF protocol versions will be enabled
--public-key	Contains the file path specification for the file containing the client-side public key.
--runpath	Sets the executable file search path.
--run-command	Specifies the command to run at startup time.
--run-script	Specifies the script to run at startup time.
--server	Specifies the server address to use in the **connect** command.
--subdirs	Specifies whether child sub-directories should be searched when looking for files.
--time-rpcs	Measure the round-trip time of each <rpc> request and <rpc-reply> at the session level.
--timeout	Specifies the timeout to use in the **connect** command.
--use-xmlheader	Specifies how file result variables will be written for XML files. Controls whether the XML preamble header will be written or not.
--user	The default user name for NETCONF sessions.
--version	Prints the program version and exits.
--warn-idlen	Controls how identifier lengths are checked.
--warn-linelen	Controls how line lengths are checked.
--warn-off	Suppresses the specified warning number.
--yuma-home	Specifies the **$YUMA_HOME** project root to use when searching for files.

2.2 Invoking Commands

Commands can be entered with parameters:

- in one continuous line

```
merge target=/toaster/toastControl --value=”down”
```

- in several lines (using line continuation)

```
merge target=/toaster/toastControl \
 --value=”down”
```

- interactively prompted for each parameter

```
merge
(will be prompted for target and value)
```

- a combination of the above

```
merge target=/toaster/toastControl
(will be prompted for value)
```

When a command is entered, and the yangcli script interpreter is running in interactive mode (**--batch-mode** not active), then the user will be prompted for any missing mandatory parameters.

If the **--optional** parameter is present (or the **$$optional** system variable is set to 'true'), then the user will be prompted for any optional parameters that are missing.

A command has the basic form:

<name (QName)> <parameter (any YANG type)>*

The command name is a qualified name matching the name used in the YANG rpc statement. This is followed by zero or more parameters (in any order).

A command parameter has the same syntax as a CLI configuration parameter.

The command name syntax is described below.

An un-escaped end-of-line character ('enter key') terminates command line input.

2.2.1 Command Prompt

The **yangcli** command prompt changes, depending on the context.

Idle Mode:

If the script interpreter is idle and there is no NETCONF session active, then the prompt is simply the program name:

```
yangcli>
```

If the script interpreter is idle and there is a NETCONF session active, then the prompt is the program name, followed by ' <user>@<server>', depending on the parameters used in the **connect** command:

```
yangcli andy@myserver>
```

Continuation Mode:

If a backslash, end-of-line sequence ended the previous line, the prompt will simply be the word 'more' indented 3 spaces:

```
yangcli andy@myserver> get \
   more>
```

The 'more>' prompt will continue if the new line also ends in with an escaped end-of-line. When a new line is finally terminated, all the fragments are spliced together and delivered as one command line.

Note: context-sensitive command completion is not available in this mode.

Choice mode:

If a partial command has been entered in interactive mode, and the script interpreter needs to prompt for a YANG 'choice' parameter, then a list of numbered cases will be presented, and the prompt will be the same as it was before (depending on whether a NETCONF session is active or not), except a colon character (':'), followed by the command name, will be added at the end. As long as parameters for the same command are being entered (i.e., prompted for child nodes within a selected case, the command name will be appended to the prompt.

```
yangcli andy@myserver> sget

Enter a number of the selected case statement:

  1: case varref:
       leaf varref
  2: case from-cli:
       leaf target
       leaf optional
       anyxml value

Enter choice number [1 - 2]:
yangcli andy@myserver:sget>
```

Parameter mode:

If a partial command has been entered in interactive mode, and the script interpreter needs to prompt for a leaf or leaf-list, then the parameter name will appear in angle brackets ('<' and '>').

```
Filling mandatory case /sget/input/from/from-cli:
Enter string value for leaf <target>
yangcli andy@myserver:sget>
```

If the '**ncx:password**' extension is part of the YANG definition for the leaf or leaf-list, then the characters entered at the prompt in this mode will not be echoed, and they will not be saved in the command history buffer. Any default value will not be printed either. Instead, 4 asterisks '****' will be printed, even though the correct value will be used in the command.

If a default value is available, it will appear in square brackets ('[' and ']'). In this case, entering 'return' will not be interpreted as an empty string, but rather the default value that was presented.

```
yangcli> connect

Enter string value for leaf <user> [andy]
yangcli:connect>

Enter string value for leaf <server> [myserver]
yangcli:connect>

Enter string value for leaf <password> [****]
yangcli:connect>
Enter uint16 value for leaf <port> [830]

yangcli:connect>

Enter uint32 value for leaf <timeout> [30]

yangcli:connect>
```

Note: After a NETCONF session is terminated for any reason, the connection parameters will be remembered , and presented as defaults the next time the connect command is entered.

False Conditional Block Mode

If a conditional command (if, elif, else, or while command) is active, but the conditional expression is false, then any commands defined within that conditional block will not be executed. If this occurs in interactive mode instead of a script, the string '[F]' will be added to the command prompt. Note that the 'help' and 'log-info' commands do not get executed in the following example:

```
yangcli> if 0

yangcli[F]> help

yangcli[F]> log-info 'this will not get printed'

yangcli[F]> end

yangcli>
```

2.2.2 Command Name

The command name can be entered with or without an XML prefix:

```
yangcli andy@myserver> nc:get
yangcli andy@myserver> get
```

If there is a prefix (e.g., 'nc:get'), then it needs to be one of the XML prefixes in use at the time. Use the 'show modules' command to see the modules and prefixes in use. The YANG prefix will usually be the same as the XML prefix, but not always.

XML prefixes are required to be unique, so if any 2 YANG modules pick the same prefix, then 1 of them has to be changed for XML encoding purposes.

If the **--default-module** configuration parameter (or **$$default-module** system variable) is set, it will be used to resolve a command name without any prefix, if it is not a NETCONF or **yangcli** command.

An error message will be printed if the command entered cannot be found in any YANG module, or if there are multiple commands that match the same string.

2.2.3 ncx:default-parm Extension

Each command may define a default parameter, by placing an '**ncx:default-parm**' extension in the rpc input section in the YANG rpc statement. This extension allows less typing in **yangcli** to accomplish the same thing.

If the script interpreter encounters a string in the command line that is not a recognized parameter for that command, and there is a default parameter defined, then the string will be used as a value for the default parameter.

For example, the 'show' parameter is the default parameter for the 'history' command, so both of these commands will be interpreted to mean the same thing:

```
      history --show=10
      history 10
```

Note: the default parameter does not allow the wrong form of a parameter type to be entered, to accept the default for that parameter. For example, the 'show' parameter above has a default value of '25':

```
      # this is the same as history show=25
      history

      # this is an error, not the same as the above
      history show
```

The following table shows the default parameters that are available at this time.

Default Parameters

command	default parameter
cd	dir

connect	server
create	target
elif	expr
else	expr
if	expr
eventlog	show
fill	target
help	command
history	show
insert	target
load	module
log-debug	msg
log-error	msg
log-info	msg
log-warn	msg
merge	target
mgrload	module
recall	index
replace	target
run	script
set-log-level	log-level
sget	target
sget-config	target
xget	select
xget-config	select
while	expr

2.2.4 Parameter Mode Escape Commands

There are 4 escape sequence commands that can be used while entering parameters.

They all begin with the question mark character ('?'), and end with the 'Enter' key.

Control key sequences are not used because that would interfere with command line editing keys.

Parameter mode escape sequences

escape sequence	description
?	Print some help text
??	Get all available help text

?s	Skip this parameter
?c	Cancel this command

Note: If the current parameter is considered hidden (**ncx:hidden** extension used in the YANG definition), then no help will be available for the parameter, even though it is accessible. This also apples to the **help** command. Any command or parameter designated as **ncx:hidden** will be treated as an unknown identifier, and no help will be given.

Note: Skipping mandatory nodes with the '?s' command is affected by the **--bad-data** configuration parameter and **$$bad-data** system variable. An error, warning, or confirmation check may occur. Refer to the CLI Reference for more details.

Note: If there are any YANG defined values (e.g., enumeration, bits, default-stmt) available for the current parameter, then pressing the tab key will list the full or partial completions available.

2.2.5 Using XPath Expressions

There are some command parameters, such as the **--target** parameter for the **create** command, that accept XPath absolute path expressions.

If prefixes are present, then they must match the set of XML prefixes in use at the time. Use the **show modules** command to see the current set of prefixes.

If prefixes are not used, then the first available matching XML namespace will be used instead.

If the starting forward slash character ('/') is missing, then it will be added.

```
# these are all the same value
yangcli:fill> system
yangcli:fill> /system
yangcli:fill> /sys:system
```

It is important to remember 2 simple rules to avoid common errors in XPath expressions:

1. String constants must be quoted with single quote characters.
   The expression [name=fred] is not the same as [foo='fred'].
   The former compares the 'name' node value to the 'fred' node value.
   The latter compares the 'name' node value to the string 'fred'.
2. The double quote character ('”') is not allowed in XPath **--select** parameter expressions because the expression will be sent to the server inside a double-quoted string.

If an incomplete XPath absolute path expression is entered, and the script interpreter is in interactive mode, then the user will be prompted to fill in any missing mandatory nodes or key leafs.

```
# complete form of ifMtu leaf
yangcli:fill> /interfaces/interface[name='eth0']/ifMtu
# incomplete form of ifMtu leaf

yangcli:fill> /interfaces/interface/ifMtu

Filling key leaf <name>:
Enter string value:
```

The **--select** parameter for the **xget** and **xget-config** commands accepts full XPath expressions. The expression must yield a node-set result in order to be used as a filter in NETCONF **get** and **get-config** operations.

One of the simplest XPath filters to use is the **descendant-or-self** filter ('//<expr>').

For example, this command will retrieve all instances of the 'ifMtu' leaf:

```
xget //ifMtu
```

When interface (or any list) entries are returned by netconfd, they will contain the the entire path back to the root of the YANG module, not just the specified node. Also, any key leafs within a list are added. This is only done if the XPath expression is missing any predicates for key leafs.

This is different than XPath 1.0 as used in XSLT. Since NETCONF **get** and **get-config** operations return complete XML instance documents, not node-sets, the ancestor nodes and naming nodes need to be added.

```
# reply shown with --display-mode=plain
data {
   interfaces {
      interface eth0 {
          name eth0
          ifMtu 1500
      }
      interface eth1 {
          name eth1
          ifMtu 1518
      }
   }
}
```

2.2.6 Special Parameter Handling

Some special handling of YANG data structures is done by the script interpreter.

Containers

Some parameters, such as the **--source** and **--target** parameters in many commands, are actually represented as a container with a single child -- a choice of several leaf nodes. In this situation, just the name of the desired leaf node can be entered (when in idle mode), as the 'contents' of the container parameter.

```
sget-config /system source=candidate
```

Choices and Cases

If a parameter name exact-match is not found, and a partial match is attempted, then choice and case node names will be ignored, and not cause a match.

Since these nodes never appear in the XML PDUs they are treated as transparent nodes (wrt/ parameter searches) unless they are specified with their full name.

Parameters that are a choice of several nodes, similar to above, except without a parent container node, (e.g., **--help-mode**) can be omitted. The accessible child nodes within the case nodes can be entered directly (e.g., **sget --target** parameter).

```
# this is not allowed because 'help-mode' is not complete
yangcli> help --command=help --help-mo=brief

# this is allowed because 'help-mode' is complete,
# even though help-mode is a choice and 'brief' is
# an empty leaf
yangcli> help help help-mode=brief

# choice and case names are transparent when
# searching for parameter names, so the
# following command is the same as above
yangcli> help help brief
```

Lists and Leaf-Lists

When filling a data structure and a descendant node is entered, which is a YANG list or leaf-list, then multiple entries can be entered. After the node is filled in, there will be a prompt (Y/N, default no) to add more list or leaf-list entries.

Binary Data Type

The YANG binary data type is supported. Parameters of this type should be entered in plain text and they will be converted to binary format.

2.2.7 Command Completion

The 'tab' key is used for context-sensitive command completion:

- If no command has been started, a list of available commands is printed
- If a partial command is entered, a list of commands which match the characters entered so far is printed
- If a command is entered, but no parameters, then a list of available parameters is printed
- If a command is entered, and the cursor is within a command name, then a list of parameters which match the characters entered so far is printed
- If a command is entered, and the cursor is after a command name, but not within a value string, then a list of available parameters is printed

- If a command is entered, and the cursor is within a command value, then a list of possible values which match the characters entered so far is printed. Note that not all data types support value completion at this time.
- If no values are available, but a default value is known, that value will be printed

Command list example: no NETCONF session is active:

```
yangcli> <hit tab key>
cd       fill     history  mgrload  quit     run
connect  help     list     pwd      recall   show
```

Command list example: NETCONF session is active

```
yangcli andy@myserver.example.com> <hit tab key>
cd                     get-schema             recall
close-session          help                   replace
commit                 history                restart
connect                insert                 run
copy-config            kill-session           save
create                 list                   sget
create-subscription    load                   sget-config
delete                 load-config            show
delete-config          lock                   shutdown
discard-changes        merge                  unlock
edit-config            mgrload                validate
fill                   no-op                  xget
get                    pwd                    xget-config
get-config             quit
```

2.2.8 Command Line Editing

The command line parser is based on **libtecla**, a freely available library.

The home page is located here:

http://www.astro.caltech.edu/~mcs/tecla/

The complete user guide for configuring **libtecla** is located here:

http://www.astro.caltech.edu/~mcs/tecla/tecla.html

If the file **$HOME/.teclarc** exists, then **libtecla** will use it to configure the key bindings.

By default, **libtecla** uses **emacs** key bindings. There is no need for any further **libtecla** configuration if **emacs** mode is desired.

In order to use the **vi** editor key bindings, the **$HOME/.teclarc** file must exist, and it must contain the following line:

```
edit-mode vi
```

Custom key bindings are also available. Refer to the **libtecla** documentation for more details on command line editing key customization.

The control key sequence (^F == control key and f key at the same time). The letter is not case-sensitive, so ^F and ^f are the same command.

The alt key sequence (M-f == alt key and f key at the same time). The letter is not case-sensitive, so M-F and M-f are the same command.

The following table shows the the most common default key bindings:

Common editing key bindings

command	description
^F	cursor right
^B	cursor-left
^A	beginning of line
^E	end of line
^U	delete line
M-f	forward-word
M-b	backward word
^P	up history
^N	down history

2.2.9 Command History

Each command line is saved in the command history buffer, unless a password is being entered in parameter mode.

By default, the previous history line (if any) will be shown if the ^P key is pressed.

By default, the next history line (if any) will be shown if the ^N key is pressed.

In addition, the **history** command can be used to control the command line buffer further. This command has 4 sub-modes:

- **show**: show maximum of N history entries (default is 25)
- **clear**: clear the history buffer
- **save**: save the history buffer to a file
- **load**: load the history buffer from a file

By default, the command line history buffer is loaded when the program starts, and saved when the program exits. This behavior can be turned off by setting the **--autohistory** configuration parameter to 'false'.

The '!' character is special when editing commands. If the first character is '!', and it is followed by a number or a non-zero character, the line will be interpreted as a recall request:

- yangcli> !42   == recall command number 42 (same as recall 42)
- yangcli> !get   == recall the most recent command line starting with 'get'

Refer to the Command Reference section for more details on the **history** command.

2.2.10 Command Responses

The command output and debugging messages within yangcli is controlled by the current log level (error, warn, info, debug, debug2, debug3).

If a command is executed by the script interpreter, then a response will be printed, depending on the log level value.

If the log level is 'info' or higher, and there were no errors and no response, then the string 'OK' is printed.

```
      yangcli> $foo = 7
         OK
      yangcli>
```

If the log-level is set to 'error' or 'warn', then the 'OK' messages will be suppressed.

If the log level is set to 'debug' or higher, then responses from the server will be echoed to the log (file or STDOUT). The current display mode will be used when printing data structures such as <rpc-error> and <notification> element contents.

If an error response is received from the server, it will always be printed to the log.

```
yangcli andy@myserver> create /system

Filling container /system:
RPC Error Reply 5 for session 8:

rpc-reply {
   rpc-error {
      error-type application
      error-tag access-denied
      error-severity error
      error-app-tag limit-reached
      error-path /nc:rpc/nc:edit-config/nc:config/sys:system
      error-message 'max-access exceeded'
   }
}

yangcli andy@myserver>
```

Refer the the **--log-level** parameter in the CLI Reference for more details.

2.3 NETCONF Sessions

The **yangcli** program can be connected to one NETCONF server at a time.

Run multiple instances of yangcli to control multiple agents at once.

Use the connect command to start a NETCONF session.

This section explains how to use **yangcli** to manage a NETCONF server, once a session is established.

When a NETCONF session starts up, a <capability> exchange is done, and the server reports exactly what content it supports. This information is used extensively to customize the session, and report errors and warnings for the session.

2.3.1 Connection Startup Screen

If the **--log-level** is set to 'info' of higher, then a startup screen will be displayed when a NETCONF session is started. It contains:

- startup banner
- client session ID
- server session ID
- protocol capabilities supported by the server
  - Includes revision-date of supported module
- YANG modules supported by the server
  - Includes any features and deviations supported in the module
- Enterprise specific capabilities supported by the server
- Default target database (<candidate> or <running>)
- Save operation mapping for the server
- **with-defaults** reporting capability reported by the server

The following example shows a typical startup screen connecting to the **netconfd** server:

```
NETCONF session established for andy on myserver

Client Session Id: 1
Server Session Id: 8

Server Protocol Capabilities
   Protocol Version: RFC 4741
   candidate:1.0
   rollback-on-error:1.0
   validate:1.0
   xpath:1.0
   notification:1.0
   interleave:1.0
   with-defaults:1.0
   netconf-monitoring:1.0
   schema-retrieval:1.0

Server Module Capabilities
   ietf-inet-types@2009-11-10
   ietf-netconf-monitoring@2009-11-20
   ietf-with-defaults@2009-07-01
      Features:
         with-defaults
   ietf-yang-types@2009-11-10
   nc-notifications@2008-07-14
   notifications@2008-07-14
   test@2009-12-26
      Features:
         feature1
         feature3
         feature4
   yuma-app-common@2010-01-25
   yuma-interfaces@2009-11-21
   yuma-mysession@2009-08-11
   yuma-nacm@2009-11-21
   yuma-ncx@2009-12-21
   yuma-proc@2009-11-21
   yuma-system@2009-12-27
   yuma-types@2010-01-25
```

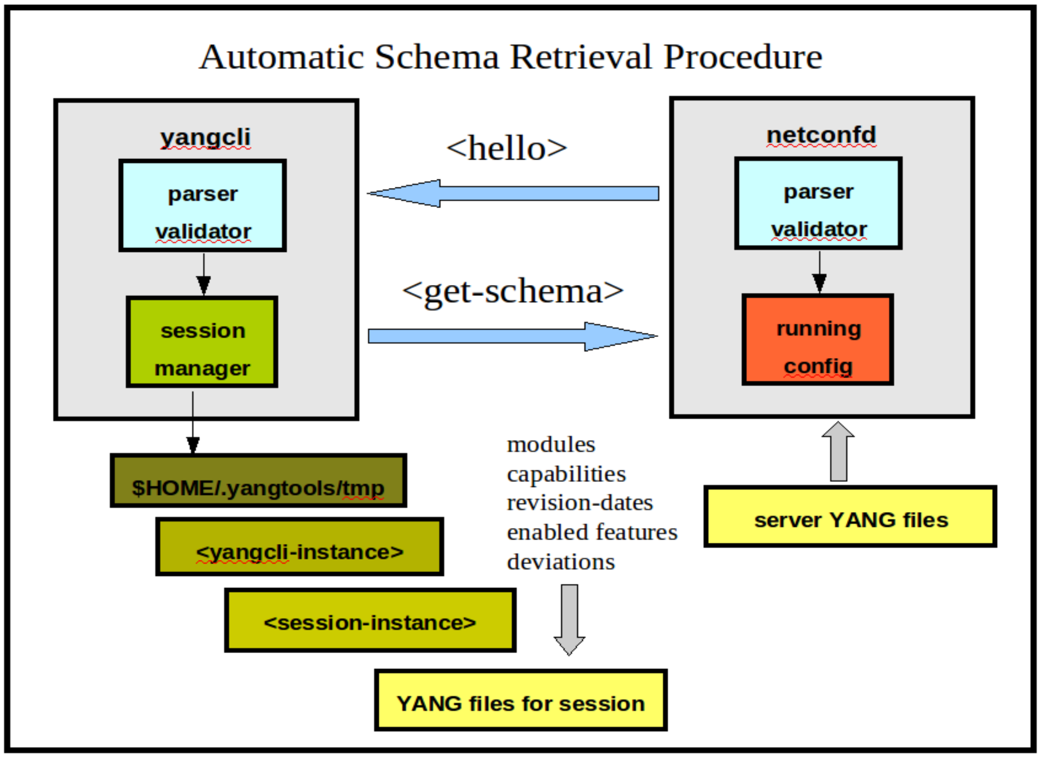
```
Server Enterprise Capabilities
   None

Default target set to: <candidate>
Save operation mapped to: commit
Default with-defaults behavior: explicit
Additional with-defaults behavior: trim:report-all

Checking server Modules...

yangcli andy@myserver>
```

2.3.2 Server Tailored Context



While a NETCONF session is active, the set of available YANG modules will be set to the modules that the server is using, if the **--autoload** configuration parameter is enabled.

If the :schema-retrieval capability is also available on the server, then the <get-schema> operation will be attempted for any YANG module specified in the <hello> message capabilities, but not available to the **yangcli** program.

When the server module capabilities are analyzed by the **yangcli** client, the entire YANG module search path is checked for the specific module advertised in the capability. All the modules are partially parsed to check the actual namespace and revision date values. The following fields must exactly match in order for yangcli to use a local YANG module, if **--autoload**=true.

- module name
- module revision date (if any)
- module namespace

If the namespace URI value is different, it indicates that there is either a bug in one of the conflicting YANG modules, or that two different naming authorities picked the same module name. In this case, a warning will be generated during session initialization.

Any data returned from the server for YANG modules not currently available will be treated as a YANG 'anyxml' node, instead of the (unknown) YANG data type.

If the module contains YANG features that are not advertised in the <capabilities> exchange, then those data definitions will not be available (by default) for use in **yangcli** commands.

If the module contains an object with a 'when' statement, and the 'when' XPath expression evaluates to 'false', then that data definition will not be available (by default) for use in **yangcli** commands.

The **help** command will be tailored to the modules, capabilities, features, and module deviations reported by the server in <capability> exchange.

2.3.3 Retrieving Data

There are 6 commands available to retrieve generic data (i.e., an arbitrary subset of an entire NETCONF database):

command	description
get	raw NETCONF <get> operation
get-config	raw NETCONF <get-config> operation
sget	high-level subtree filter, using the <get> protocol operation
sget-config	high-level subtree filter, using the <get-config> protocol operation
xget	high-level XPath filter, using the <get> protocol operation
xget-config	high-level XPath filter, using the <get-config> protocol operation

All the high-level retrieval operations support the **$$with-defaults** system variable. The <with-defaults> parameter will be added the the NETCONF PDU if this variable is set to a value other than 'none' (the default). This system variable will be used as the default if not entered directly.

```
sget /system --with-defaults=$$with-defaults
```

This parameter can also be specified directly, each time the command is used.

```
xget-config //ifMtu --with-defaults=trim
```

The **$$bad-data** system variable is used to control how invalid operations and data are sent to the server. The **xget** and **xget-config** commands are affected by this parameter. If the **:xpath** capability was not advertised by the server when the session started, an error or warning may occur if these commands are used.

If any data is received that **yangcli** does not understand, then a warning message will be printed and the data will be treated as if it was defined with the YANG '**anyxml**' data type.

2.3.4 Modifying Data

The following commands are available to modify generic data (i.e., an arbitrary subset of an entire NETCONF database):

command	description
commit	raw NETCONF <commit> operation
create	high-level <edit-config> operation, with nc:operation='create'
delete	high-level <edit-config> operation, with nc:operation='delete'
delete-config	raw NETCONF <delete-config> operation
discard-changes	raw NETCONF <discard-changes> operation
edit-config	raw NETCONF <edit-config> operation
fill	fill a variable for re-use in other operations
insert	high-level <edit-config> operation, with YANG insert operation extensions
lock	lock a NETCONF database
merge	high-level <edit-config> operation, with nc:operation='merge'
replace	high-level <edit-config> operation, with nc:operation='replace'
save	High level save operation, depending on the default target (candidate or running)
unlock	unlock a NETCONF database

All the high-level editing operations use the **--target** parameter reported by the server when the session started up. If the server did not report the **:candidate** or **:writable-running** capabilities, then there will be no writable target, and an error will occur if these commands are entered.

All the high-level editing operations support the **$$default-operation** system variable. The <default-operation> parameter will be added the the NETCONF <edit-config> PDU if this variable is set to a value other than 'not-used'. The default is the enumeration 'none', which means do not use any default operation, and only use the explicit **nc:operation** attribute.

All the high-level editing operations support the **$$test-option** system variable. The <test-option> parameter will be added the the NETCONF <edit-config> PDU if this variable is set to a value other than 'none' (the default). This system variable will be used as the default if not entered directly.

```
replace /interfaces/interface[name='eth0']/ifMtu \
      --test-option=$$test-option \
      --value=$newvalue
```

This parameter can also be specified directly, each time the command is used.

```
$newvalue = 1518

replace /interfaces/interface[name='eth0']/ifMtu \
      --test-option=test-only \
      --value=$newvalue
```

All the high-level retrieval operations support the **$$error-option** system variable. The <error-option> parameter will be added the the NETCONF <edit-config> PDU if this variable is set to a value other than 'none' (the default). This system variable will be used as the default if not entered directly.

```
replace /interfaces/interface[name='eth0']/ifMtu \\
      --error-option=$$error-option \
      --value=1518
```

This parameter can also be specified directly, each time the command is used.

```
replace /interfaces/interface[name='eth0']/ifMtu \
      --error-option=rollback-on-error \
      --value=1518
```

The high level **save** command is mapped to other commands, depending on the capabilities reported by the server.

save command

capabilities	real command(s)
:candidate	commit
:writable-running	<none>
:startup	copy-config --source=running \ --target=startup

2.3.5 Using Notifications

The **create-subscription** command is used to start receiving notifications.

The **netconfd** server will include a <sequence-id> element in any notification that is saved in the replay buffer. This unsigned integer can be used to help debug notification filters (i.e., if non-consecutive <sequence-id> values are received, then the notification was filtered, or dropped due to access control policy).

If any replay notifications are desired, then the **--startTime** parameter must be included. At the end of the stored notifications, the server will send the <replayComplete> event. This notification type is not saved, and will not be found in the server replay buffer, if replay is supported by the server. The **netconfd** server will not include a <sequence-id> element in this notification type.

If the notification subscription should stop at a certain time, then the **--stopTime** parameter must be included. At the end of the stored notifications, the server will send the <replayComplete> event, followed by the <notificationComplete> event. . This notification type is not saved, and will not be found in the server replay buffer, if replay is supported by the server. The **netconfd** server will not include a <sequence-id> element in this notification type.

Notifications are printed to the log, using the current **$$display-mode** system variable setting, when and if they are received.

Notifications are also logged. Use the **eventlog** command to access the notifications stored in the event log.

2.3.6 Configuration Parameters That Affect Sessions

The **--server**, **--user**, and **--password** configuration parameters can be used to start a NETCONF session automatically at startup time. The connect command will only be attempted at startup time if the **--server** parameter is present.

If all 3 of these parameters are present at startup time, then no interactive prompting for additional optional parameters will be done. Instead the connect command will be invoked right away.

During normal operation, the **--optional** configuration parameter, or the **$$optional** system variable, can be used to control interactive prompting for optional parameters.

The **--server** parameter is saved in the **$$server** system variable, which can be overwritten at any time. If set, this will be used as the initial default value for the **--server** parameter in the **connect** command.

The **--fixorder** configuration parameter can be used to control XML PDU ordering. If set to 'true', then the PDU will be reordered (if needed),to use the canonical order, according to the YANG specification. If 'false', the order parameters are entered at the command line will be their NETCONF PDU order as well. The default is 'true'. To send the server incorrectly ordered data structures on purposes, set this parameter to 'false'.

The **--user** parameter is saved in the **$$user** system variable, which can be overwritten at any time. If set, this will be used as the initial default value for the **--user** parameter in the **connect** command.

The **--with-defaults** configuration parameter, or the **$$with-defaults** system variable, can be used to set the default value for the 'with-defaults' parameter extension for the NETCONF **get**, **get-config**, and **copy-config** protocol operations. The default is 'none'.

The **--error-option** configuration parameter, or the **$$error-option** system parameter, can be used to set the default value for the **--error-option** parameter for the NETCONF edit-config protocol operation. The default is 'none'.

The **--test-option** configuration parameter, or the **$$test-option** system parameter, can be used to set the default value for the **--test-option** parameter for the NETCONF edit-config protocol operation. The default is 'none'.

The **--bad-data** configuration parameter, or the **$$bad-data** system variable, can be used to control how **yangcli** handles parameter values that are known to be invalid, or usage of optional protocol operations that the current session does not support. The default value is 'check'. To use **yangcli** in a testing mode to send the server incorrect data on purpose, set this parameter to 'warn' or 'ignore'.

2.3.7 Trouble-shooting NETCONF Session Problems

If the NETCONF session does not start for any reason, one or more error messages will be printed, and the prompt will indicate 'idle' mode. This section assumes that the server is **netconfd**, and these debugging steps may not apply to all NETCONF agents.

If the NETCONF session does not start:

- make sure the server is reachable
  - try to 'ping' the server and see if it responds
- make sure the SSH server is responding
  - try to login in to the server using normal SSH login on port 22
- make sure a firewall is not blocking TCP port 830
  - try to connect to the NETCONF server using the **--port=22** option
- make sure the netconf-subsystem is configured correctly in /etc/ssh/sshd_config
  there should be the proper configuration commands for NETCONF to work. For example, the **netconfd** server configuration might look like this:
-

```
Port 22
Port 830
Subsystem    netconf      /usr/sbin/netconf-subsystem
```

- make sure the **netconfd** server is running. Use the unix 'ps' command, or check the **netconfd** log file, to make sure it is running.

```
ps -alx | grep netconf
(look for 1 'netconfd and N 'netconf-subsystem' -- 1 for
 each active session)
```

- make sure the user name is correct
  - This must be a valid user name on the system
- make sure the password is correct
  - This must be the valid password (in /etc/passwd or /etc/shadow) for the specified user name

If the NETCONF session stops responding:

- make sure the server is still reachable
  - try to 'ping' the server and see if it responds
- make sure the SSH server is still responding
  - try to login in to the server using normal SSH login on port 22

If the NETCONF server is not accepting a certain command:

- make sure the command (or parameters used in the command) is actually supported by the server.
  - There may be features, when statements, or deviation statements that indicate the server does not support the command or one or more of its parameters.
- make sure that access control configured on the server is not blocking the command. The error-tag should be 'access-denied' in this case.

If the NETCONF server is not returning the expected data in a <get> or <get-config> protocol operation::

- Make sure all the parameters are supported by the server
    - the **:xpath** capability must be advertised by the server to use the 'select' attribute in the <get> or <get-config> operations
    - the **:with-defaults** capability must be advertised by the server to use the <with-defaults> parameter
- if using a filter, try to retrieve the data without a filter and see if it is there
- make sure that access control configured on the server is not blocking the retrieval. There will not be any error reported in this case. The server will simply skip over any unauthorized data, and leave it out of the <rpc-reply>.
- set the logging level to debug2 or higher, and look closely at the PDUs being sent to the server. Set the display mode to a value other than 'plain' to make sure the correct namespaces are being used in the request.

If an <edit-config> operation is failing unexpectedly:

- make sure that access control configured on the server is not blocking the request. The error-tag should be 'access-denied' in this case.
- make sure an unsupported parameter or parameter value is not used
    - <test-option> is not supported unless the **:validate** capability is advertised by the server
    - <error-option> = 'rollback-on-error' is not supported unless the **:rollback-on-error** capability is advertised by the server
- if the request contains an edit to a nested data structure, make sure the parent data structure(s) are in place as expected. The <default-operation> parameter is set to 'none' in the high level editing operations, so any data 'above' the edited data must already exist.
- set the logging level to debug2 or higher, and look closely at the PDUs being sent to the server. Set the display mode to a value other than 'plain' to make sure the correct namespaces are being used in the request.

2.4 Command Reference

This section describes all the **yangcli** local and remote commands built-in when using the **netconfd** server.

There may be more or less commands available, depending on the YANG modules actually loaded at the time.

The specific NETCONF capabilities needed are listed for each remote command. No capabilities are ever needed for a local command.

It is possible that other agents will support protocol operations that **netconfd** does not support. If yangcli has the YANG file available for the module, then it can be managed with the high level commands. Low-level commands can still be used with external data (e.g., @mydatafile.xml).

Any YANG rpc statement can be used as a remote **yangcli** command. Refer to the server vendor documentation for details on the protocol operations, database contents, and notification definitions that they support.

2.4.1 CD

The **cd** command is used to change the current working directory.

cd command

Command type:	local
Default parameter:	dir
Min parameters:	1
Max parameters:	1
Return type:	status
YANG file:	yangcli.yang

Command Parameters:

- **dir**
  - type: string
  - usage: mandatory
  - default: none
  - The '**dir**' string must contain a valid directory specification

Positive Response:

- the new current working directory is printed

Negative Response:

- an error message will be printed describing the error

Usage:

```
      yangcli> cd ~/modules

      Current working directory is /home/andy/modules

      yangcli> cd --dir=$YUMA_HOME

      Current working directory is
/home/andy/swdev/yuma/trunk/netconf

      yangcli>
```

2.4.2 CLOSE-SESSION

The **close-session** command is used to terminate the current NETCONF session. A NETCONF server should always accept this command if it is valid, and not reject it due to access control enforcement or if the server is in notification delivery mode.

close-session command

Command type:	remote
Default parameter:	none
Min parameters:	0
Max parameters:	0
Return type:	status
YANG file:	yuma-netconf.yang

Command Parameters:

- none

Positive Response:

- the session is terminated and the command prompt is changed to indicate idle mode

Negative Response:

- an <rpc-error> message will be printed describing the error

Usage:

```
yangcli andy@myserver> close-session

RPC OK Reply 2 for session 10:

yangcli>
```

Reference:

- RFC 4741, section 7.8

2.4.3 COMMIT

The **commit** command is used to save the edits in the <candidate> database into the <running> database. If there are no edits it will have no effect.

commit command

Command type:	remote
Default parameter:	none
Min parameters:	0
Max parameters:	2
Return type:	status
YANG file:	yuma-netconf.yang
Capabilities needed:	:candidate
Capabilities optional:	:confirmed-commit

Command Parameters:

- **confirmed**
  - type: empty
  - usage: optional
  - default: none
  - capabilities needed: :confirmed-commit
  - This parameter requests a confirmed commit procedure. The server will expect another **commit** command before the **confirm-timeout** time period expires.
- **confirm-timeout**
  - type: uint32 (range: 1 .. max)
  - usage: optional
  - default: 600 seconds
  - capabilities needed: :confirmed-commit
  - This is the number of seconds to request before the timeout.
    The '**confirmed**' leaf must also be present for this parameter to have any affect.
- **persist**
  - type: string
  - usage: optional
  - persist ID used to start or extend the confirmed commit procedure
- **persist-id**
  - type: string
  - usage: optional
  - persist ID used to conform a previously started the confirmed commit procedure

Positive Response:

- the session is terminated and the command prompt is changed to indicate idle mode

Negative Response:

- an <rpc-error> message will be printed describing the error

Usage:

```
yangcli andy@myserver> commit

RPC OK Reply 5 for session 10:

yangcli andy@myserver>
```

Reference:

- RFC 4741, section 8.3.4

2.4.4 CONNECT

The **connect** command is used to start a session with a NETCONF server.

If there already is a NETCONF session active, then an error message will be printed and the command will not be executed.

connect command

Command type:	remote
Default parameter:	server
Min parameters:	3
Max parameters:	8
Return type:	status
YANG file:	yangcli.yang

Command Parameters:

- **server**
  - type: inet:ip-address (string containing IP address or DNS name
  - usage: mandatory
  - default: previous server used, if any, will be presented as the default, but not used automatically
  - This parameter specifies the server address for the session.
- **password**
  - type: string (ncx:password)
  - usage: mandatory
  - default: previous password used, if any, will be presented as the default, but not used automatically
  - This parameter specifies the password string to use to establish the session. It will not be echoed in parameter mode or saved in the command history buffer.
- **port**
  - type: uint16
  - usage: optional
  - default: 830
  - This parameter specifies the TCP port number that should be used for the session.
- **timeout**
  - type: uint32 (0 = no timeout, otherwise the number of seconds to wait)
  - usage: optional
  - default: set to the **$$timeout** system variable, default 30 seconds
  - This parameter specifies the number of seconds to wait for a response from the server before giving up. The session will not be dropped if a remote command times out, but any late response will be dropped. A value of zero means no timeout should be used, and **yangcli** will wait forever for a response.
- **user**

  - type: string
  - usage: mandatory
  - default: previous user name used, if any, will be presented as the default, but not used automatically
  - This parameter specifies the user name to use for the session. This must be a valid user name on the NETCONF server.
- **protocols**
  - type: bits (netconf1.0 netconf1.1)
  - usage: optional
  - default: **--protocols** configuration parameter setting
  - Specifies which NETCONF protocol versions to enable. Overrides **–protocols** configuration parameter.
- **private-key**
  - type: string
  - usage: optional
  - default: --private-key configuration parameter setting
  - Specifies the SSH private key file to use. Overrides –private-key configuration parameter.
- **public-key**
  - type: string
  - usage: optional
  - default: **--public-key** configuration parameter setting
  - Specifies the SSH public key file to use. Overrides **–public-key** configuration parameter.

Positive Response:

- the session is started and the prompt changes to include the 'user@server' string.

Negative Response:

- One or more error messages will be printed. Refer to the section on trouble-shooting NETCONF Session problems for more details.

Usage:

```
yangcli> connect myserver user=andy password=yangrocks

<startup screen printed>

yangcli andy@myserver>
```

2.4.5 COPY-CONFIG

The **copy-config** command is used to copy one entire NETCONF database to another location.

Not all possible parameter combinations will be supported by every server. In fact, the NETCONF protocol does not require any parameters to be supported unless the **:startup** or **:url** capabilities is supported by the server.

If the server supports the :startup capability, then it must support:

```
yangcli andy@myserver> copy-config source=running target=startup
```

This is the standard way to save a snapshot of the current running configuration in non-volatile storage, if the server has a separate startup database. If not, the server will automatically save any changes to the running configuration to non-volatile storage.

copy-config command

Command type:	remote
Default parameter:	none
Min parameters:	2
Max parameters:	3
Return type:	status
YANG file:	yuma-netconf.yang
Capabilities needed:	none
Capabilities optional:	:candidate :startup :url :with-defaults

Command Parameters:

- **source**
  - type: container with 1 of N choice of leafs
  - usage: mandatory
  - default: none
  - This parameter specifies the name of the source database for the copy operation.
  - container contents: 1 of N:
    - **candidate**
      - type: empty
      - capabilities needed: :candidate
    - **running**
      - type: empty
      - capabilities needed: none
    - **startup**

            - type: empty
            - capabilities needed: startup
        - **config:**
            - type: container (in-line configuration data)
            - capabilities needed: none
        - **url**
            - type: yang:uri
            - capabilities needed: :url, and the scheme used in the URL must be specified in the :url capability 'schemes' parameter
            - To enter this parameter, the interactive mode must be used. The shorthand mode (source=url) cannot be used, since this parameter contains a string.
- **target**
    - type: container with 1 of N choice of leafs
    - usage: mandatory
    - default: none
    - This parameter specifies the name of the target database for the copy operation.
    - container contents: 1 of N:
        - **candidate**
            - type: empty
            - capabilities needed: :candidate
        - **running**
            - type: empty
            - capabilities needed: :writable-running (still optional to implement)
                - **netconfd** does not support this mode
        - **startup**
            - type: empty
            - capabilities needed: startup
        - **url**
            - type: yang:uri
            - capabilities needed: :url, and the scheme used in the URL must be specified in the :url capability 'schemes' parameter.
            - To enter this parameter, the interactive mode must be used. The shorthand mode (target=url) cannot be used, since this parameter contains a string.
- **with-defaults**
    - type: enumeration (none report-all report-all-tagged trim explicit)
    - usage: optional
    - default: none
    - capabilities needed: with-defaults

- ◦ This parameter controls how nodes containing only default values are copied to the target database.

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Usage:

```
      yangcli andy@myserver> copy-config source=candidate

      Enter a number of the selected case statement:

        1: case candidate:
             leaf candidate
        2: case running:
             leaf running
        3: case startup:
             leaf startup
        4: case url:
             leaf url

      Enter choice number [1 - 4]:
      yangcli andy@myserver:copy-config> 4

      Filling optional case /copy-config/input/target/config-
source/url
      Enter string value for leaf <url>:
      yangcli andy@myserver:copy-config> file://configs/myconfig.xml

      RPC OK Reply 12 for session 10:
      yangcli andy@myserver>
```

Reference:

- RFC 4741, section 7.3

2.4.6 CREATE

The **create** command is a high-level <edit-config> operation. It is used to create some new nodes in the default target database.

A target node is specified (in 1 of 2 ways), and then the data structure is filled in. Only mandatory nodes will be filled in unless the **$$optional** system variable is set to 'true'.

Refer to the **fill** command for more details on interactive mode data structure completion.

create command

Command type:	remote
Default parameter:	target
Min parameters:	1

Max parameters:	5
Return type:	status
YANG file:	yangcli.yang
Capabilities needed:	:candidate or :writable-running

Command Parameters:

- **choice 'from'** (not entered)
  - type: choice of case 'varref' or case 'from-cli'
  - usage: mandatory
  - default: none
  - This parameter specifies the where **yangcli** should get the data from, for the create operation. It is either a user variable or from interactive input at the command line.
    - **varref**
      - type: string
      - usage: mandatory
      - default: none
      - This parameter specifies the name of the user variable to use for the target of the create operation. The parameter must exist (e.g., created with the **fill** command) or an error message will be printed.
    - **case from-cli** (not entered)
      - **target**
        - type: string
        - usage: mandatory
        - default: none
        - This parameter specifies the database target node of the create operation. This is an **ncx:schema-instance** string, so any instance identifier, or absolute path expression, or something in between, is accepted.
      - **urltarget**
        - type: string
        - usage: optional
        - default: none
        - This parameter specifies the database target node of the create operation. This is an **UrlPath** string.
      - **optional**
        - type: empty
        - usage: optional
        - default: controlled by **$$optional** system variable
        - This parameter controls whether optional nodes within the target will be filled in. It can be used to override the **$$optional** system variable, when it is set to 'false'.

  - **value**
    - type: anyxml
    - usage: mandatory
    - default: none
    - This parameter specifies the value that should be used for the contents of the **target** parameter. If it is entered, then the interactive mode prompting for missing parameters will be skipped, if this parameter is complete (or all mandatory nodes present if the **$$optional** system variable is 'false'). For example, if the target is a leaf, then specifying this parameter will always cause the interactive prompt mode to be skipped.
- **timeout**
  - type: uint32 (0 = no timeout, otherwise the number of seconds to wait)
  - usage: optional
  - default: set to the **$$timeout** system variable, default 30 seconds
  - This parameter specifies the number of seconds to wait for a response from the server before giving up. The session will not be dropped if a remote command times out, but any late response will be dropped. A value of zero means no timeout should be used, and yangcli will wait forever for a response.

System Variables:

- **$$default-operation**
  - The <default-operation> parameter for the <edit-config> operation will be derived from this variable.
- **$$error-option**
  - The <error-option> parameter for the <edit-config> operation will be derived from this variable
- **$$test-option**
  - The <test-option> parameter for the <edit-config> operation will be derived from this variable

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Usage:

```
yangcli andy@myserver> create varref=myvar

RPC OK Reply 10 for session 10:

yangcli andy@myserver> create /nacm/rules/data-rule \
(user will be prompted to fill in the data-rule contents)

RPC OK Reply 11 for session 10:

yangcli andy@myserver> create \
             target=/nacm/rules/data-rule[name='test
rule']/comment \
             value="this test rule is temporary. Do not remove!"
      (no user prompting; <edit-config> request sent right away)

      RPC OK Reply 12 for session 10:
      yangcli andy@myserver>
```

Reference:

- RFC 4741, section 7.2

2.4.7 CREATE-SUBSCRIPTION

The create-subscription command is used to start receiving notifications from the server.

The :notification capability must be supported by the server to use this command.

Unless the :interleave capability is also supported by the server, then only the **close-session** command can be used while notifications are being delivered.

create-subscription command

Command type:	remote
Default parameter:	none
Min parameters:	0
Max parameters:	4
Return type:	status
YANG file:	notifications.yang
Capabilities needed:	:notification
Capabilities optional:	:interleave

Command Parameters:

- **stream**
  - type: string
  - usage: optional
  - default: 'NETCONF'
  - This parameter specifies the name of the notification stream for this subscription request. Only the 'NETCONF' stream is mandatory to implement. Any other stream contains vendor-specific content, and may not be fully supported, depending on the stream encoding.
- **filter**
  - type: anyxml (same as the <get> or <get-config> filter parameter)
  - usage: optional
  - default: none

    - This parameter specifies a boolean filter that should be applied to the stream. This is the same format as the standard <filter> element in RFC 4741, except that instead of creating a subset of the database for an <rpc-reply> PDU, the filter is used as a boolean test to send or drop each notification delivered from the server.
        - If any nodes are left in the 'test response', the server will send the entire notification.
        - If the result is empty after the filter is applied to the “test response”, then the server will not send the notification at all.
        - It is possible that access control will either cause the a notification to be dropped entirely, or may be pruned and still delivered. The standard is not clear on this topic. The **netconfd** server will prune any unauthorized payload from an eventType, but if the <eventType> itself is unauthorized, the entire notification will be dropped.
- **startTime**
    - type: yang:date-and-time
    - usage: optional
    - default: none
    - This parameter causes any matching replay notifications to be delivered by the server, if notification replay is supported by the server. A notification will match if its <eventTime> value is greater or equal to the value of this parameter.
    - After all the replay notifications are delivered, the server will send a <replayComplete> eventType, indicating there are no more replay notifications that match the subscription request.
- **stopTime**
    - type: yang:date-and-time
    - usage: optional (not allowed unless startTime is also present)
    - default: none
    - This parameter causes any matching replay notifications to be delivered by the server, if notification replay is supported by the server. A notification will match if its <eventTime> value is less than the value of this parameter.
        - This parameter must be greater than the **startTime** parameter, or an error will be returned by the server.
        - If this parameter is used, then the entire subscription will stop after this specified time, even if it is in the future. The <notificationComplete> eventType will be sent by the server when this event occurs.
        - If this parameter is not used (but **startTime** is used), then the server will continue to deliver 'live' notifications after the <replayComplete> eventType is sent by the server.

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Usage:

```
      yangcli andy@myserver> create-subscription
```

```
RPC OK Reply 13 for session 10:

yangcli andy@myserver> create-subscription \
      startTime=2009-01-01T00:00:00Z

RPC OK Reply 14 for session 10:

yangcli andy@myserver>
```

Reference:

- RFC 5277, section 2.1.1

2.4.8 DELETE

The **delete** command is a high-level <edit-config> operation. It is used to delete an existing subtree in the default target database.

A target node is specified, and then any missing key leafs (if any) within the data structure are filled in. If the target is a leaf-list, then the user will be prompted for the value of the leaf-list node to be deleted.

Refer to the **fill** command for more details on interactive mode data structure completion.

delete command

Command type:	remote
Default parameter:	target
Min parameters:	1
Max parameters:	1
Return type:	status
YANG file:	yangcli.yang
Capabilities needed:	:candidate or :writable-running

Command Parameters:

- **target**
  - type: string
  - usage: optional (urltarget or target must be present)
  - default: none
  - This parameter specifies the database target node of the delete operation. This is an **ncx:schema-instance** string, so any instance identifier, or absolute path expression, or something in between, is accepted.
- **urltarget**
  - type: string
  - usage: optional
  - default: none

    - This parameter specifies the database target node of the delete operation. This is an **UrlPath** string.

System Variables:

- **$$default-operation**
    - The <default-operation> parameter for the <edit-config> operation will be derived from this variable.
- **$$error-option**
    - The <error-option> parameter for the <edit-config> operation will be derived from this variable
- **$$optional**
    - Controls whether optional descendant nodes will be filled into the **target** parameter contents
- **$$test-option**
    - The <test-option> parameter for the <edit-config> operation will be derived from this variable

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Usage:

```
        yangcli andy@myserver> delete /nacm/rules/data-rule \
        (user will be prompted to fill in the data-rule 'name' key 
leaf)

        RPC OK Reply 15 for session 10:

        yangcli andy@myserver> delete \
                target=/nacm/rules/data-rule[name='test rule']/comment
        (no user prompting; <edit-config> request sent right away)

        RPC OK Reply 16 for session 10:
        yangcli
```

Reference:

- RFC 4741, section 7.2

2.4.9 DELETE-CONFIG

The **delete-config** command is used to delete an entire NETCONF database.

Not all possible **target** parameter values will be supported by every server. In fact, the NETCONF protocol does not require that any database be supported by this operation.

If the server supports the :url capability, then it may support deletion of local file databases in this manner.:

delete-config command

Command type:	remote
Default parameter:	none
Min parameters:	1
Max parameters:	1
Return type:	status
YANG file:	yuma-netconf.yang
Capabilities needed:	none
Capabilities optional:	:candidate :startup :url

Command Parameters:

- **target**
  - type: container with 1 of N choice of leafs
  - usage: mandatory
  - default: none
  - This parameter specifies the name of the target database for the delete operation.
  - container contents: 1 of N:
    - **startup**
      - type: empty
      - capabilities needed: startup
      - a server may support this target
    - **url**
      - type: yang:uri
      - capabilities needed: :url, and the scheme used in the URL must be specified in the :url capability 'schemes' parameter.
      - To enter this parameter, the interactive mode must be used. The shorthand mode (target=url) cannot be used, since this parameter contains a string.
      - a server may support this parameter

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Usage:

```
yangcli andy@myserver> delete-config target=startup

RPC OK Reply 17 for session 10:

yangcli andy@myserver>
```

Reference:

- RFC 4741, section 7.4

2.4.10 DISCARD-CHANGES

The **discard-changes** command is used to delete any edits that exist in the <candidate> database, on the NETCONF server. The server will only accept this command if the :candidate capability is supported. If the <candidate> database is locked by another session, then this request will fail with an 'in-use' error.

discard-changes command

Command type:	remote
Default parameter:	none
Min parameters:	0
Max parameters:	0
Return type:	status
YANG file:	yuma-netconf.yang
Capabilities needed:	:candidate

Command Parameters: none

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Usage:

```
yangcli andy@myserver> discard-changes

RPC OK Reply 18 for session 10:

yangcli andy@myserver>
```

Reference:

- RFC 4741, section 8.3.4.2

2.4.11 EDIT-CONFIG

The edit-config command allows a subset of a NETCONF database on the server to be changed.

If the server supports the **:url** capability, then it may support editing of local file databases.

If the server supports the :candidate capability, then it will allow edits to the <candidate> database.

If the server supports the :writable-running capability, it will support edits to the <running> database.

It is not likely that a server will support the <candidate> and <running> database as targets at the same time, since changes to the <running> configuration would not be reflected in the <candidate> database, while it was being edited by a different session.

edit-config command

Command type:	remote
Default parameter:	none
Min parameters:	2
Max parameters:	5
Return type:	status
YANG file:	yuma-netconf.yang
Capabilities needed:	:candidate or :writable-running
Capabilities optional:	:url :rollback-on-error :validate

Command Parameters:

- **default-operation**
  - type: enumeration (merge replace none)
  - usage: optional
  - default: merge
  - This parameter specifies which edit operation will be in effect at the start of the operation, before any **nc:operation** attribute is found.
    - The high-level edit operations provided by **yangcli** will set this parameter to 'none'. This is the safest value, since only subtrees that have an explicit **nc:operation** attribute in effect can possibly be altered by the command.
    - If the value is 'merge', then any missing nodes in the database will be automatically created as needed.
    - If the value is 'replace', then the target database will be pruned to match the edits, as needed. Only the data from the **config** parameter will remain if this value is used. (Use with extreme caution).

- **error-option**
  - type: enumeration (stop-on-error continue-on-error rollback-on-error
  - usage: optional
  - default: stop-on-error
  - This parameter specifies what the server should do when an error is encountered.
    - The rollback-on-error value is only allowed if the **:rollback-on-error** capability is supported by the server.
    - The standard is not clear what continue-on-error really means. It is suggested that this value not be used. It is possible that the server will validate all input parameters before making any changes, no matter how this parameter is set.
- **choice edit-content** (not entered)
  - **config**
    - type: anyxml
    - usage: mandatory
    - default: none
    - This parameter specifies the subset of the database that should be changed. This is the most common way to edit a NETCONF server database, since it is mandatory to support by all agents.
  - **url**
    - type: yang:uri
    - capabilities needed: **:url,** and the scheme used in the URL must be specified in the **:url** capability 'schemes' parameter.
    - To enter this parameter, the interactive mode must be used. The shorthand mode (target=url) cannot be used, since this parameter contains a string.
- **target**
  - type: container with 1 of N choice of leafs
  - usage: mandatory
  - default: none
  - This parameter specifies the name of the target database for the edit operation.
  - container contents: choice: 1 of N:
    - **candidate**
      - type: empty
      - capabilities needed: :candidate
    - **running**
      - type: empty
      - capabilities needed: :writable-running
- **test-option**
  - type: enumeration (test-then-set set test-only)
  - usage: optional
  - default: set

- This parameter specifies how the server should test the **edit-content** parameter before using it.
  - If the value is 'set' (normal case), the server will apply validation tests as needed for the individual data structures being edited
  - The value 'test-then-set' is only allowed if the :validate capability is supported by the server. The server will test if the entire database will be valid after the edits are made, before making any changes to the candidate configuration.
    - This mode is very resource intensive. Set this parameter to 'set' for better performance, and run the validation tests manually with the **validate** command.
  - The value 'test-only' is not supported by all agents. It will be in the next version of the NETCONF protocol, but is non-standard at this time.
    - Use this value to check if a specific edit should succeed or not, allowing errors to be corrected before altering the database for real.

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Usage:

```
yangcli andy@myserver> edit-config target=candidate \
       default-operation=merge \
       test-option=test \
       error-option=stop-on-error \
       config=@myconfig.xml

RPC OK Reply 19 for session 10:

yangcli andy@myserver>
```

Reference:

- RFC 4741, section 7.2

2.4.12 ELIF

The **elif** command is used to define a conditional command block after an if command.

This command must be entered within the same script as the if command, when used within a script. It can be used zero or more times within an if command sequence.

The **expr** parameter is used to specify the XPath expression to test if the elif conditional block is true or false. A false block will be skipped and a true block will be executed. The command prompt will contain the string '[F]' while inside a false conditional block in interactive mode. This expression string should be entered with quotes to avoid misinterpreting any whitespace or special characters.

The **docroot** parameter (if present) specifies the XML document that the 'expr' parameter should be evaluated against. This is optional, since only XPath path expressions need to refer to a document.

Even if the 'expr' expression is true, the conditional block will only be executed if no conditional block in the if command sequence has already been executed.

```
if command
....
[elif command]
....
[elif-command]
...
[else command]
...
end command
```

elif command

Command type:	local
Default parameter:	expr
Min parameters:	1
Max parameters:	2
Return type:	status
YANG file:	yangcli.yang

Command Parameters:

- **expr**
  - type: XPath expression string
  - usage: mandatory
  - default: none
  - This parameter specifies the XPath expression to determine if the following commands are within a true or false conditional block.
- **docroot**
  - type: anyxml
  - usage: optional (typically a variable reference is used)
  - default: none
  - This parameter specifies the XML document that should be used if the expr XPath expression references any path nodes.

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if errors are detected locally.

- elif without a previous if command will cause an error
- elif following an 'else' command will cause an error
- invalid XPath expression or invalid docroot reference will cause an error

Usage:

```
yangcli andy@myserver> elif expr='$x > 4'

yangcli andy@myserver>
```

2.4.13 ELSE

The **else** command is used to define a final conditional command block after an if command.

This command must be entered within the same script as the if command, when used within a script. It can be used zero or one time within an if command sequence.

The conditional command block following the else command will only be executed if no conditional block has already been executed in the same if command sequence.

```
if command
....
[elif command]
....
[elif-command]
...
[else command]
...
end command
```

else command

Command type:	local
Default parameter:	none
Min parameters:	0
Max parameters:	0
Return type:	status
YANG file:	yangcli.yang

Command Parameters: none

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if errors are detected locally.
    - else without a previous if command will cause an error

Usage:

```
yangcli andy@myserver> else

yangcli andy@myserver>
```

2.4.14 END

The **end** command is used to terminate a conditional command block after an if command block, or after a 'while' command.

This command must be entered within the same script as the if or while command, when used within a script.

if command

....

[elif command]

....

[elif-command]

...

[else command]

...

end command

while command

....

end command

end command

Command type:	local
Default parameter:	none
Min parameters:	0

Max parameters:	0
Return type:	status
YANG file:	yangcli.yang

Command Parameters: none

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if errors are detected locally.
  - else without a previous if command will cause an error

Usage:

```
yangcli andy@myserver> end

yangcli andy@myserver>
```

2.4.15 EVAL

The **eval** command is used to evaluate an XPath expression..

The **expr** parameter is used to specify the XPath expression to evaluate.   This expression string should be entered with quotes to avoid misinterpreting any whitespace or special characters.

The **docroot** parameter (if present) specifies the XML document that the 'expr' parameter should be evaluated against. This is optional, since only XPath path expressions need to refer to a document.

eval command

Command type:	local
Default parameter:	expr
Min parameters:	1
Max parameters:	2
Return type:	data
YANG file:	yangcli.yang

Command Parameters:

- **expr**
  - type: XPath expression string
  - usage: mandatory

  - default: none
  - This parameter specifies the XPath expression to determine if the following commands are within a true or false conditional block.
- **docroot**
  - type: anyxml
  - usage: optional (typically a variable reference is used)
  - default: none
  - This parameter specifies the XML document that should be used if the expr XPath expression references any path nodes.

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if errors are detected locally.
  - elif without a previous if command will cause an error
  - elif following an 'else' command will cause an error
  - invalid XPath expression or invalid docroot reference will cause an error

Output:

- **data**
  - type: anyxml
  - This element will contain the result from the XPath expression. A node-set result will produce a complex element return value, and all other XPath result types will produce a string return value.

Usage:

```
      yangcli andy@myserver> $x = eval '$x + 1'

      yangcli andy@myserver> $sysname = eval '//sysName' 
docroot=$backup
```

2.4.16 EVENTLOG

The **eventlog** command is used to view or clear all or part of the notification event log. This log will be empty if no well-formed notifications have been received from any server.

The **eventlog show** command is used to display some event log entries.

The **eventlog clear** command is used to delete some event log entries.

If no parameters are entered, it is the same as entering 'eventlog show=-1'.

The event log is not automatically emptied when a session terminates, in case the session was dropped unexpectedly. New entries will be appended to the event log as new sessions and/or subscriptions are started.

eventlog command

Command type:	local
Default parameter:	show
Min parameters:	0
Max parameters:	3
Return type:	status
YANG file:	yangcli.yang

Command Parameters:

- **choice eventlog-action** (not entered):
  - type: choice of case 'show-case' or leaf 'clear'
  - usage: optional
  - default: show=-1 is used as the default if nothing entered
  - This parameter specifies the event log action that should be performed.
    - **clear**
      - type: int32 (-1 to clear all entries; 1 to max to delete N entries)
      - usage: optional
      - default: -1
      - This parameter specifies the maximum number of event log entries to be deleted, starting from the oldest entries in the event log. The value -1 means delete all the entries. Otherwise the value must be greater than zero, and up to that many entries will be deleted.
    - **case show-case** (not entered)
      - **choice help-mode** (not entered) (default choice is 'normal')
        - **brief**
          - type: empty
          - usage: optional
          - default: none
          - This parameter specifies that brief documentation mode should be used. The event log index, sequence ID, and <eventType> will be displayed in this mode.
        - **normal**
          - type: empty
          - usage: optional
          - default: none
          - This parameter specifies that normal documentation mode should be used. The event log index, <eventTime>, sequence ID, and <eventType> will be displayed in this mode.
        - **full**
          - type: empty

        - usage: optional
        - default: none
        - This parameter specifies that full documentation mode should be used. The event log index, <eventTime>, sequence ID, and <eventType> will be displayed in this mode. In addition, the entire contents of the notification PDU will be displayed, using the current **$$display-mode** setting.
    - **show**
        - type: int32 (-1 for all, 1 to max for N entries)
        - usage: optional
        - default: -1
        - This parameter specifies the number of event log entries that should be displayed. The value '-1' indicates all the entries should be displayed. Otherwise, the value must be greater than zero, indicating the number of entries to display.
    - **start**
        - type: uint32
        - usage: optional
        - default: 0
        - This parameter specifies the start position in the event log to start displaying entries. The first entry is number zero. Each time the event log is cleared, the numbering restarts.

System Variables:

- $$display-mode
    - The log entries printed when help-mode='full' are formatted according to the current value of this system variable.

Positive Response:

- the event log entries are printed or cleared as requested

Negative Response:

- An error message will be printed if errors are detected.

Usage:

```
yangcli andy@myserver> eventlog show=5 start=3

 [3]    [2009-07-10T02:21:10Z] (4)      <sysSessionStart>
 [4]    [2009-07-10T02:23:14Z] (5)      <sysSessionEnd>
 [5]    [2009-07-10T02:23:23Z] (6)      <sysSessionStart>
 [6]    [2009-07-10T02:24:52Z] (7)      <sysConfigChange>
 [7]    [2009-07-10T02:24:57Z] (8)      <sysSessionEnd>

yangcli andy@myserver>
```

2.4.17 FILL

The **fill** command is used to create a user variable for reuse in other commands.

It is used in an assignment statement to create a variable from various sources.

If it is not used in an assignment statement, then the result will not be saved, so the command will have no effect in this case.

The value contents will mirror the subtree within the NETCONF database indicated by the target parameter. If not completely provided, then missing descendant nodes will be filled in interactively, by prompting for each missing node.

fill command

Command type:	local
Default parameter:	target
Min parameters:	2
Max parameters:	3
Return type:	data
YANG file:	yangcli.yang

Command Parameters:

- **optional**
  - type: empty
  - usage: optional
  - default: controlled by **$$optional** system variable
  - This parameter controls whether optional nodes within the target will be filled in. It can be used to override the **$$optional** system variable, when it is set to 'false'.
- **target**
  - type: string
  - usage: mandatory
  - default: none
  - This parameter specifies the database target node of the create operation. This is an **ncx:schema-instance** string, so any instance identifier, or absolute path expression, or something in between, is accepted.
- **value**
  - type: anyxml
  - usage: mandatory
  - default: none
  - This parameter specifies the content to use for the filled variable.
    - If this parameter is not entered, then the user will be prompted interactively to fill in the **target** data structure.
    - If a string is entered, then the target value being filled must be a leaf or leaf-list.
    - If a variable reference is entered, then it will be used as the content, if the target value being filled is a leaf or a leaf-list.
    - If the target value is a complex object, then the referenced variable must also be a complex object of the same type.

- An error will be reported if the global or local variable does not reference the same object type as the target parameter.

System Variables:

- $$optional
  - Controls whether optional descendant nodes will be filled into the **target** parameter contents

Positive Response:

- OK

Negative Response:

- An error message will be printed if errors are detected.

Output:

- data
  - type: anyxml
  - The data structure will mirror the requested target object.
  - The variable (if any) will retain the target object name and namespace so it can be used in other operations more easily. In the example below, the **$my_interface** local variable will have the module name 'interfaces' and name 'interface', when used in other commands such as **create** or **merge**.

Usage:

```
yangcli> $my-interface = fill \
       target=/interfaces/interface optional
       (user will be prompted to fill in all fields
        of the <interface> element)
   OK

yangcli>
```

2.4.18 GET

The **get** command is used to retrieve data from the server.

get command

Command type:	remote
Default parameter:	none
Min parameters:	0
Max parameters:	2
Return type:	data
YANG file:	yuma-netconf.yang

Command Parameters:

- **filter**
  - type: anyxml
  - usage: optional
  - default: none
  - This parameter specifies a boolean filter that should be applied to the stream. Any data in the <running> database (or non-config data) that does not match the filter will be left out of the <rpc-reply> response.
    - If no filter is used, the server will return the entire <running> database and all non-config data as well. This could be a lot of data, depending on the server.
    - If the result is empty after the filter is applied to the available data, then the server will send an empty <data> element in the <rpc-reply>
    - It is possible that access control will cause the <rpc-reply> to be pruned. The **netconfd** server will silently prune any unauthorized payload from the <rpc-reply>.
- **with-defaults**
  - type: enumeration (none report-all report-all-tagged trim explicit)
  - usage: optional
  - default: none
  - capabilities needed: with-defaults
  - This parameter controls how nodes containing only default values are returned in the <rpc-reply>.

Positive Response:

- the server returns <data>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Output:

- **data**
  - type: anyxml
  - This element will contain the requested data from the **<**running> database, or non-config data from the server instrumentation.

Usage:

```
yangcli andy@myserver> get

RPC Data Reply 20 for session 10:

rpc-reply {
    data {
       …. data returned by the server
    }
}
```

```
yangcli andy@myserver> get filter=@myfilter.xml

RPC Data Reply 21 for session 10:

rpc-reply {
    data {
    }
}

(the previous response will occur if the filter did not
 match anything or the server access control filtered the
 entire response)

yangcli andy@myserver>
```

Reference:

- RFC 4741, section 7.7

2.4.19 GET-CONFIG

The **get-config** command is used to retrieve configuration data from the server.

get-config command

Command type:	remote
Default parameter:	none
Min parameters:	1
Max parameters:	3
Return type:	data
YANG file:	yuma-netconf.yang

Command Parameters:

- **filter**
  - type: anyxml
  - usage: optional
  - default: none
  - This parameter specifies a boolean filter that should be applied to the stream. Any data in the <running> database (or non-config data) that does not match the filter will be left out of the <rpc-reply> response.
    - If no filter is used, the server will return the entire <running> database and all non-config data as well. This could be a lot of data, depending on the server.
    - If the result is empty after the filter is applied to the available data, then the server will send an empty <data> element in the <rpc-reply>
    - It is possible that access control will cause the <rpc-reply> to be pruned. The **netconfd** server will silently prune any unauthorized payload from the <rpc-reply>.

- **source**
  - type: container with 1 of N choice of leafs
  - usage: mandatory
  - default: none
  - This parameter specifies the name of the source database for the retrieval operation.
  - container contents: 1 of N:
    - **candidate**
      - type: empty
      - capabilities needed: :candidate
    - **running**
      - type: empty
      - capabilities needed: none
    - **startup**
      - type: empty
      - capabilities needed: startup
    - **url**
      - type: yang:uri
      - capabilities needed: :url, and the scheme used in the URL must be specified in the :url capability 'schemes' parameter
      - To enter this parameter, the interactive mode must be used. The shorthand mode (source=url) cannot be used, since this parameter contains a string.
- **with-defaults**
  - type: enumeration (none report-all report-all-tagged trim explicit)
  - usage: optional
  - default: none
  - capabilities needed: with-defaults
  - This parameter controls how nodes containing only default values are returned in the <rpc-reply>.

Positive Response:

- the server returns <data>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Output:

- **data**
  - type: anyxml
  - This element will contain the requested data from the **source** database.

Usage:

```
yangcli andy@myserver> $my-config = get-config target=running

RPC Data Reply 22 for session 10:

rpc-reply {
    data {
       …. entire database returned by the server
    }
}

yangcli andy@myserver> @saved-config.xml = get-config \
      filter=@myfilter.xml  \
      target=candidate

rpc-reply {
    data {
      … data requested by the filter
    }
}

yangcli andy@myserver>
```

Reference:

- RFC 4741, section 7.1

2.4.20 GET-LOCKS

The **get-locks** command is a high-level wrapper for the <lock> operation. It is used to lock all the databases (<running> plus <candidate> and/or <startup> if they exist). If all the locks cannot be obtained, then release all the locks that were obtained (all-or-nothing).

The entire time to wait for a lock in use is set with the lock-timeout parameter.

The retry-interval parameter is used when the <lock> operation fails with a 'lock-denied' error-tag, because some other session has the lock.

If the <candidate> cannot be locked for another reason, a <discard-changes> operation will be attempted to clear any leftover edits.

Normally, the errors received while attempting to acquire locks are not printed to the log, like normal commands. Instead, if $$log-level system parameter is set to 'debug2' or 'debug3', then these messages will be printed.

get-locks command

Command type:	remote
Default parameter:	none
Min parameters:	0
Max parameters:	3
Return type:	status
YANG file:	yangcli.yang

Command Parameters:

- **lock-timeout**
  - type: uint32 (seconds)
  - usage: optional
  - default: 120 seconds (2 minutes)
  - This parameter specifies how long to wait for a lock that is in use by another session.
- **retry-interval**
  - type: uint32 (seconds)
  - usage: optional
  - default: 2 seconds
  - This parameter specifies how long to wait to retry for a lock.
- **cleanup**
  - type:boolean
  - usage: optional
  - default: true
  - This parameter controls whether the 'release-locks' command will be called automatically if the entire set of required locks cannot be granted.

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Usage:

```
yangcli andy@myserver> get-locks lock-timeout=0

Sending <lock> operations for get-locks...

RPC OK Reply 6 for session 23:

RPC OK Reply 7 for session 23:

get-locks finished OK
yangcli andy@myserver>
```

2.4.21 GET-MY-SESSION

The **get-my-session** command is used to retrieve the session customization parameters from the server. It is only supported by the **netconfd** server.

The session indent amount, session line size, and default behavior for the with-defaults parameter can be controlled at this time.

get-my-session command

Command type:	remote
Default parameter:	none
Min parameters:	0
Max parameters:	0
Return type:	data
YANG file:	mysession.yang
Capabilities needed:	none

Command Parameters:

- none

Positive Response:

- the server returns <indent>, <linesize>, and <with-defaults> elements

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Output:

- **indent**
  - type: uint32 (range 0 to 9)
  -
- **linesize**
  - type: uint32
  - This parameter specifies the desired line length for the session.
- **with-defaults**
  - type: enumeration (none report-all trim explicit)
  - This parameter specifies the desired default with-defaults filtering behavior for the session.
  -

```
yangcli andy@myserver> get-my-session
RPC Data Reply 25 for session 10:
rpc-reply {
    data {
      indent 3
      linesize 72
      with-defaults report-all
    }
}

yangcli andy@myserver>
```

2.4.22 GET-SCHEMA

The **get-schema** command is used to retrieve data model definition files from the server. This is part of the NETCONF monitoring draft. The server must support the **:schema-retrieval** capability to use this command.

If the server reports a module or module version that **yangcli** cannot find in its local module library, then an error message will be printed. The **get-schema** command can then be used to retrieve the missing module from the server.

The **ietf-netconf-monitoring.yang** module includes a list of the schema supported by the server, which can be retrieved from a server that supports this module, such as **netconfd**.

```
yangcli andy@myserver> sget /netconf-state/schemas
```

The preceding command will return a <schemas> container with several <schema> child nodes. One example entry is shown below:

```
schemas {
    schema system 2009-06-04 YANG {
       identifier system
       version 2009-06-04
       format YANG
       namespace http://netconfcentral.org/ns/yuma-system
       location NETCONF
    }
}
```

The <identifier>, <version> and <format> leafs can be used as the corresponding parameter values for the **get-schema** command. See the example below for more details.

get-schema command

Command type:	remote
Default parameter:	none
Min parameters:	3
Max parameters:	3
Return type:	data
YANG file:	ietf-netconf-monitoring.yang
Capabilities needed:	:schema-retrieval

Command Parameters:

- **identifier**
  - type: string
  - usage: mandatory
  - default: none

  - This parameter specifies the name of the module to retrieve.
    - Do not use any path specification of file extension; just the module name is entered.
    - The name is case-sensitive, and must be specified exactly as defined.
- **version**
  - type: string
  - usage: mandatory
  - default: none
  - This parameter specifies the version of the module to retrieve.
    - For YANG modules, this will be the most recent revision date string defined in a module revision statement.
    - If any version is acceptable, or if the specific version is not known, then use the empty string.
- **format**
  - type: enumeration (XSD YANG YIN RNG)
  - usage: mandatory
  - default: none
  - This parameter specifies the format of the module to be retrieved.
    - XSD: W3C REC REC-xmlschema-1-20041028
    - YANG: RFC 6020
    - YIN: RFC 6020
    - RNG: ISO/IEC 19757-2
    - **netconfd** only supports the YANG and YIN formats

Positive Response:

- the server returns <data>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Output:

- data
  - type: anyxml
  - **yangcli** will strip off this XML container if the command result is being saved to a text file. Only the YANG contents will be saved instead.

Usage:

```
yangcli andy@myserver> @notifications.yang = get-schema \
       identifier=notifications \
       version=2009-06-04 \
       format=YANG

RPC Data Reply 24 for session 10:
```

```
rpc-reply {
    data {
       …. entire notifications.yang contents
    }
}

(after retrieval, the module can be loaded locally
 with the mgrload command)

yangcli andy@myserver> mgrload notificications.yang

   OK

yangcli andy@myserver>
```

Reference:

- draft-ietf-netconf-monitoring-06.txt

2.4.23 HELP

The help command is used to print documentation to STDOUT.

If no session is active, then only help for the local commands and the standard NETCONF commands will be available.

If a NETCONF session is active, then the documentation shown will attempt to exactly match the capabilities of the server.

If additional (i.e., augment generated) parameters are available, then they will be shown in the command output. If the server does not implement some parameters (e.g., feature not supported) then these parameters will not be shown in the command output.

If the server has modified an object with deviation statements, then the altered object will be shown.

The **ncx:hidden** extension suppresses the **help** command. If this extension is present in the YANG definition associated with the request, then no help will be available for that object or command.

help command

Command type:	local
Default parameter:	command
Min parameters:	3
Max parameters:	3
Return type:	data
YANG file:	ietf-netconf-monitoring.yang
Capabilities needed:	:schema-retrieval

Command Parameters:

- **choice helptype** (not entered)
  - **command**
    - type: string

    - usage: mandatory
    - default: none
    - This parameter specifies the name of the command for which documentation is requested
  - **commands**
    - type: empty
    - usage: mandatory
    - default: none
    - This parameter will request documentation for all available commands
  - **notification**
    - type: string
    - usage: mandatory
    - default: none
    - This parameter specifies the name of the notification for which documentation is requested
  - **object**
    - type: string
    - usage: mandatory
    - default: none
    - This parameter specifies the name of the NETCONF database object for which documentation is requested.
      - Only top level objects are supported by this command.
      - Documentation for the entire object subtree will be printed, if the object is a container, choice, or list.
      - Documentation for nested objects is only available in parameter mode, using the escape commands for help ('?') and full help ('??')
  - **type**
    - type: string
    - usage: mandatory
    - default: none
    - This parameter specifies the name of the YANG typedef for which documentation is requested
    - Only top-level typedefs are supported by this command. Local typedefs within groupings, containers, or lists are not exportable in YANG.
- **choice help-mode** (not entered) (default choice is 'normal')
  - **brief**
    - type: empty
    - usage: optional
    - default: none
    - This parameter specifies that brief documentation mode should be used.

- **normal**
  - type: empty
  - usage: optional
  - default: none
  - This parameter specifies that normal documentation mode should be used.
- **full**
  - type: empty
  - usage: optional
  - default: none
  - This parameter specifies that full documentation mode should be used.

Positive Response:

- the server prints the requested help text

Negative Response:

- An error message will be printed if errors are detected.

Usage:

```
yangcli> help help full

help
   Print the yangcli help text
   input
      default parameter: command
      choice helptype
         leaf command [NcxIdentifier]
            Show help for the specified command,
            also called an RPC method

         leaf commands [empty]
            Show info for all local commands

         leaf notification [NcxIdentifier]
            Show help for the specified notification

         leaf object [NcxIdentifier]
            Show help for the specified object

         leaf type [NcxIdentifier]
            Show help for the specified type

      choice help-mode
         leaf brief [empty]
            Show brief help text

         leaf normal [empty]
            Show normal help text

         leaf full [empty]
            Show full help text

yangcli andy@myserver> help notification sysConfigChange

notification sysConfigChange
```

```
   Generated when the <running> configuration is changed.
   leaf userName [string]

   leaf sessionId [SessionId]
      range: 1..max

   leaf remoteHost [ip-address]

   leaf target [string]

   leaf operation [EditOperationType] [d:merge]
      enum values: merge replace create delete

yangcli andy@svnserver>
```

2.4.24 HISTORY

The **history** command is used to show, clear, load, or save the command line history buffer.

Use the **recall** command to recall a previously executed command line, after getting the line number from the **history show** command.

All lines entered will be saved in the history buffer except an **ncx:password** value entered in parameter mode.

When **yangcli** starts, the command line history buffer is empty. If a history file was previously stored with the **history save** command, then it can be recalled into the buffer with the **history load** command.

The **history clear** command is used to delete the entire command line history buffer.

The numbering sequence for commands, starts from zero and keeps incremented until the program exits. If the history buffer is cleared, then the number sequence will continue, not start over at zero.

history command

Command type:	local
Default parameter:	show
Min parameters:	0
Max parameters:	2
Return type:	data
YANG file:	yangcli.yang

Command Parameters:

- **choice history-action** (not entered)
  - **clear**
    - type: empty
    - usage: optional
    - default: none

    - This parameter specifies that the history buffer should be cleared. Unless the contents have been saved with the **history save** command, there is no way to recover the cleared buffer contents after this command is executed.

- **load**
    - type: string
    - usage: optional
    - default: $HOME/.yangcli_history
    - This parameter specifies a command history file, and causes the current command line history contents to be loaded from that file.
    - Special processing for this command allows the history file to be omitted in idle mode, even though the load parameter is not type 'empty'.

```
yangcli> history load
  same as:
yangcli> history load=$HOME/.yangcli_history
```

- **save**
    - type: string
    - usage: optional
    - default: $HOME/.yangcli_history
    - This parameter specifies a command history file, and causes the current command line history contents to be saved to that file.
    - Special processing for this command allows the history file to be omitted in idle mode, even though the save parameter is not type 'empty'.

```
▪               yangcli> history save
  same as:
  yangcli> history save=$HOME/.yangcli_history
```

- **show**
    - type: int32 (-1 for all entries; 1..max for N entries)
    - usage: optional
    - default: -1
    - This parameter specifies the maximum number of history entries to show.
        - If no case is selected from this choice, then the command '**history show=-1**' will be used by default.
        - The **help-mode** choice parameter is only used with the **history show** command.
            - If the **--brief** or **--normal** modes are selected the the format will include the command number and the command line.
            - If the **--full** mode is selected, then the command data and time will also be printed.

- **choice help-mode** (not entered)
  This parameter is ignored unless the **history show** command is entered.
  - **brief**
    - type: empty
    - usage: optional
    - default: none
    - This parameter specifies that brief documentation mode should be used.
  - **normal**
    - type: empty
    - usage: optional
    - default: none
    - This parameter specifies that normal documentation mode should be used.
  - **full**
    - type: empty
    - usage: optional
    - default: none
    - This parameter specifies that full documentation mode should be used.

Positive Response:

- the requested history entries will be printed for the **history show** command
- all other commands will return OK

Negative Response:

- An error message will be printed if errors are detected.

Usage:

```
      yangcli> history show=3 full
        [27] 2009-07-04 09:25:34 sget /system --nofill
        [28] 2009-07-04 09:34:17 @myconfig = get-config
source=running
        [29] 2009-07-04 09:43:54 history show=3 full

      yangcli> history save=~/my-temp-history-file
         OK
      yangcli>
```

2.4.25 IF

The **if** command is used to start a conditional command block.

The **expr** parameter is used to specify the XPath expression to test if the if conditional block is true or false. A false block will be skipped and a true block will be executed. The command prompt will contain the string '[F]' while inside a false conditional block in interactive mode. This expression string should be entered with quotes to avoid misinterpreting any whitespace or special characters.

The **docroot** parameter (if present) specifies the XML document that the 'expr' parameter should be evaluated against. This is optional, since only XPath path expressions need to refer to a document.

If the 'expr' expression is true, the conditional block will be executed, and no further conditional blocks within the same if command sequence will be executed.

```
if command
....
[elif command]
....
[elif-command]
...
[else command]
...
end command
```

if command

Command type:	local
Default parameter:	expr
Min parameters:	1
Max parameters:	2
Return type:	status
YANG file:	yangcli.yang

Command Parameters:

- **expr**
  - type: XPath expression string
  - usage: mandatory
  - default: none
  - This parameter specifies the XPath expression to determine if the following commands are within a true or false conditional block.
- **docroot**
  - type: anyxml
  - usage: optional (typically a variable reference is used)
  - default: none
  - This parameter specifies the XML document that should be used if the expr XPath expression references any path nodes.

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if errors are detected locally.

  - invalid XPath expression or invalid docroot reference will cause an error

Usage:

```
yangcli andy@myserver> if "$sysname = 'localhost'"

yangcli andy@myserver>
```

2.4.26 INSERT

The insert command is used to insert or move YANG list or leaf-list data into a NETCONF database. It is a high level command with utilizes the YANG 'insert' extensions to the NETCONF <edit-config> operation.

insert command

Command type:	remote
Default parameter:	target
Min parameters:	2
Max parameters:	7
Return type:	status
YANG file:	yangcli.yang

Command Parameters:

- **choice 'from'** (not entered)
  - type: choice of case 'varref' or case 'from-cli'
  - usage: mandatory
  - default: none
  - This parameter specifies the where **yangcli** should get the data from, for the insert operation. It is either a user variable or from interactive input at the command line.
    - **varref**
      - type: string
      - usage: mandatory
      - default: none
      - This parameter specifies the name of the user variable to use for the target of the insert operation. The parameter must exist (e.g., created with the **fill** command) or an error message will be printed.
    - **case from-cli** (not entered)
      - **target**
        - type: string

        - usage: mandatory
        - default: none
        - This parameter specifies the database target node of the insert operation. This is an **ncx:schema-instance** string, so any instance identifier, or absolute path expression, or something in between, is accepted.
    - **optional**
        - type: empty
        - usage: optional
        - default: controlled by **$$optional** system variable
        - This parameter controls whether optional nodes within the target will be filled in. It can be used to override the **$$optional** system variable, when it is set to 'false'.
    - **value**
        - type: anyxml
        - usage: mandatory
        - default: none
        - This parameter specifies the value that should be used for the contents of the **target** parameter. If it is entered, then the interactive mode prompting for missing parameters will be skipped, if this parameter is complete (or all mandatory nodes present if the **$$optional** system variable is 'false'). For example, if the target is a leaf, then specifying this parameter will always cause the interactive prompt mode to be skipped.
- **edit-target**
    - type: string
    - usage: optional (must be present if the order parameter is set to 'before' or 'after').
    - default: none
    - This parameter specifies the value or key clause that should be used, as the list or leaf-list insertion point. It identifies the existing entry that the new entry will be inserted before or after, depending on the **order** parameter.
        - For a leaf-list, the edit-target contains the value of the target leaf-list node within the configuration being edited. E.g., edit-target='fred'.
        - For a list, the edit-target contains the key values of the target list node within the configuration being edited. E.g., edit-target=[name='fred'][zipcode=90210].
- **order**
    - type: enumeration (first last before after)
    - usage: optional
    - default: last
    - The insert order that should be used. If the value 'before' or 'after' is selected, then the **edit-target** parameter must also be present.
- **operation**
    - type: enumeration (create merge replace)
    - usage: optional

  - default: merge
  - This parameter specifies the **nc:operation** attribute value for the NETCONF <edit-config> operation. The insert operation is secondary to the NETCONF operation attribute.
- **timeout**
  - type: uint32 (0 = no timeout, otherwise the number of seconds to wait)
  - usage: optional
  - default: set to the **$$timeout** system variable, default 30 seconds
  - This parameter specifies the number of seconds to wait for a response from the server before giving up. The session will not be dropped if a remote command times out, but any late response will be dropped. A value of zero means no timeout should be used, and yangcli will wait forever for a response.

System Variables:

- **$$default-operation**
  - The <default-operation> parameter for the <edit-config> operation will be derived from this variable.
- **$$error-option**
  - The <error-option> parameter for the <edit-config> operation will be derived from this variable
- **$$test-option**
  - The <test-option> parameter for the <edit-config> operation will be derived from this variable

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Usage:

```
yangcli andy@myserver> insert varref=myvar order=first

RPC OK Reply 25 for session 10:

yangcli andy@myserver> insert /nacm/rules/data-rule \
              order=after \
              edit-target=”[name='test-rule']”

(user will be prompted to fill in the data-rule contents)

RPC OK Reply 26 for session 10:

yangcli andy@myserver>
```

Reference:

- draft-ietf-netmod-yang-13

2.4.27 KILL-SESSION

The **kill-session** command is used to terminate a NETCONF session (other than the current session). All NETCONF implementations must support this command. It is needed sometimes to unlock a NETCONF database locked by a session that is idle or stuck.

If the **lock** command returns a 'lock-denied' <error-tag>, it will also include an <error-info> field called <session-id>. This is the session number that currently holds the requested lock. The same value should be used for the **session-id** parameter in this command, to terminate the session will the lock.

Note: this is an extreme measure, which should be used with caution.

kill-session command

Command type:	remote
Default parameter:	target
Min parameters:	1
Max parameters:	1
Return type:	status
YANG file:	yuma-netconf.yang

Command Parameters:

- **session-id**
  - type: uint32 (range: 1.. max)
  - usage: mandatory
  - default: none
  - This parameter specifies the session number of the currently active NETCONF session that should be terminated.

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Usage:

```
yangcli andy@myserver> kill-session session-id=11

RPC OK Reply 27 for session 10:

yangcli andy@myserver>
```

Reference:

- RFC 4741, section 7.9

2.4.28 LIST

This **list** command is used to display the commands, objects, and oids (object identifiers) that are available at the time.

The **list commands** command will display the local commands and the remote commands that are available in the current NETCONF session, or which have been loaded with the **mgrload** command.

The **list files** command will display the data files that are in the current data search path. The module parameter has no affect in this mode.

The **list modules** command will display the YANG files that are in the current YANG module search path. The module parameter has no affect in this mode.

The **list objects** command will display the top-level objects that are currently available in the current NETCONF session, or which have been loaded with the **mgrload** command.

The **list oids** command will display the object identifiers of the top-level objects that are currently available in the current NETCONF session, or which have been loaded with the **mgrload** command.

The **list scripts** command will display the script files that are in the current script search path. The module parameter has no affect in this mode.

list command

Command type:	local
Default parameter:	none
Min parameters:	1
Max parameters:	6
Return type:	data
YANG file:	yangcli.yang

Command Parameters:

- **choice help-mode** (not entered)
  This parameter is ignored if the listtype choice is set to ' the **history show** command is entered.
  - **brief**
    - type: empty
    - usage: optional
    - default: none
    - This parameter specifies that brief documentation mode should be used.
  - **normal**
    - type: empty
    - usage: optional
    - default: none
    - This parameter specifies that normal documentation mode should be used.
  - **full**
    - type: empty

        - usage: optional
        - default: none
        - This parameter specifies that full documentation mode should be used.

- **choice 'listtype'** (not entered)
    - usage: mandatory
    - default: none
    - This parameter specifies the what type of data should be listed.
    - **commands**
        - type: empty
        - usage: mandatory
        - default: none
        - This parameter specifies that the available commands should be listed.
            - If the **help-mode** is set to 'brief', then only the command names will be listed.
            - If the **help-mode** is set to 'normal', then the XML prefix and the command name will be listed.
            - If the **help-mode** is set to 'full', then the module name and the command name will be listed.
    - **files**
        - type: empty
        - usage: mandatory
        - default: none
        - This parameter specifies that all the data files in the current data search path should be listed.
    - **modules**
        - type: empty
        - usage: mandatory
        - default: none
        - This parameter specifies that all the YANG files in the current module search path should be listed.
    - **objects**
        - type: empty
        - usage: mandatory
        - default: none
        - This parameter specifies that the available top-level objects should be listed.
            - If the **help-mode** is set to 'brief', then only the object names will be listed.
            - If the **help-mode** is set to 'normal', then the XML prefix and the object name will be listed.
            - If the **help-mode** is set to 'full', then the module name and the object name will be listed.
    - **oids**

        - type: empty
        - usage: mandatory
        - default: none
        - This parameter specifies that the available top-level object identifiers should be listed.
            - The **help-mode** parameter has no effect
    - **scripts**
        - type: empty
        - usage: mandatory
        - default: none
        - This parameter specifies that all the script files in the current script search path should be listed.
- **module**
    - type: string
    - usage: optional
    - default: none
    - This parameter specifies a module name. If present then only information for the specified YANG module will be displayed.

Positive Response:

- the requested information will be printed

Negative Response:

- An error message will be printed if errors are detected.

Usage:

```
yangcli andy@myserver> list objects full module=test

   test:instance1
   test:instance2
   test:leafref1
   test:leafref2
   test:test1
   test:test2
   test:test3
   test:idtest
   test:musttest
   test:anyxml.1
   test:binary.1
   test:bits.1
   test:boolean.1
   test:empty.1
   test:enumeration.1
   test:identityref.1
   test:instance-identifier.1
   test:instance-identifier.2
   test:int8.1
   test:int16.1
   test:int32.1
   test:int64.1
   test:leafref.1
   test:leafref.2
   test:string.1
   test:uint8.1
   test:uint16.1
   test:uint32.1
   test:uint64.1
   test:dec64.1
   test:dec64.2
   test:dec64.3
   test:union.1
   test:container.1
   test:container.2
   test:list.1
   test:choice.1
   test:xpath.1

yangcli andy@myserver>
```

2.4.29 LOAD

The **load** command is used to load a YANG module into the server.

This command is only supported by the **netconfd** server.

The YANG files must be available in the module search path for the server.

Refer to the **netconfd** configuration section for more details on adding new YANG modules into the server.

After using this command, the **mgrload** command may also be needed to keep the current session synchronized with the server.

Use the **revision** parameter to load a specific revision of a module.

The server will return the revision date of the module that was loaded (or already present).

load command

Command type:	remote
Default parameter:	module
Min parameters:	1
Max parameters:	3
Return type:	data
YANG file:	yuma-system.yang

Command Parameters:

- module
  - type: string (length 1 .. 63)
  - usage: mandatory
  - default: none
  - This parameter specifies the name of the module to load.

- revision
  - type: date string (YYYY-MM-DD)
  - usage: optional
  - default: none
  - This parameter specifies the revision date of the module to load.
- deviation:
  - type: leaf-list of deviation module names
  - usage: optional (0 or more instances)
  - default: none
  - This parameter specifies a deviation module to load prior to loading the requested module.

Positive Response:

- the server returns <mod-revision>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Usage:

```
yangcli andy@myserver> load toaster revision=2009-06-23

RPC Data Reply 27 for session 10:

rpc-reply {
   mod-revision 2009-06-23
}

yangcli andy@myserver>
```

2.4.30 LOCK

The **lock** command is used to request a global lock on a NETCONF database. It is used, along with the **unlock** command, to obtain exclusive write access to the NETCONF server.

The scope of a lock command is the lifetime of the session that requested the lock. This means that if the session that owns the lock is dropped for any reason, all the locks it holds will be released immediately by the server.

The use of database locks is optional in NETCONF, but it must be implemented by every server. It is strongly suggested that locks be used if multiple managers are likely to log into the particular NETCONF server.

One or more locks may be needed to execute a transaction safely:

- If the **:writable-running** capability is supported, then a lock on the <running> database is needed. This database can be locked at any time.
- If the **:startup** capability is supported, then a lock on the <startup> database is needed. This database can be locked at any time.
- If the **:candidate** capability is supported, then a lock on the <candidate> database is needed. A lock on the <running> database is also needed.

- The <candidate> database can only be locked if there are no active edits in it.
- The **discard-changes** command may be needed to clear a <candidate> database that has been left edited by a session that terminated unexpectedly.
- Note: There is no way in NETCONF to tell the difference between an actively edited <candidate> database and an 'abandoned' <candidate> database. The server will almost never clear the <candidate> database. It will only clear any locks held. Use the discard-changes command (for other session's edits) with extreme caution.

lock command

Command type:	remote
Default parameter:	none
Min parameters:	1
Max parameters:	1
Return type:	status
YANG file:	yuma-netconf.yang
Capabilities needed:	none
Capabilities optional:	:candidate :startup :url

Command Parameters:

- **target**
  - type: container with 1 of N choice of leafs
  - usage: mandatory
  - default: none
  - This parameter specifies the name of the target database to be locked.
  - container contents: 1 of N:
    - **candidate**
      - type: empty
      - capabilities needed: :candidate
    - **running**
      - type: empty
      - capabilities needed: none
    - **startup**
      - type: empty
      - capabilities needed: startup
    - **url**
      - type: yang:uri

- capabilities needed: :url, and the scheme used in the URL must be specified in the :url capability 'schemes' parameter.
- To enter this parameter, the interactive mode must be used. The shorthand mode (target=url) cannot be used, since this parameter contains a string.

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.
  - If the <error-tag> is 'lock-denied' then the <error-info> will contain a <session-id> leaf. This identifies the session number of the current lock holder.

Usage:

```
yangcli andy@myserver> lock target=candidate

RPC OK Reply 29 for session 10:

yangcli andy@myserver>
```

2.4.31 LOG-DEBUG

The **log-debug** command prints a message to the program log, if the **$$log-level** system variable is set to 'debug' or a higher enumeration (e.g., 'debug2').

The **msg** parameter is used to provide the message string to print.

log-debug command

Command type:	local
Default parameter:	msg
Min parameters:	1
Max parameters:	1
Return type:	status
YANG file:	yangcli.yang

Command Parameters:

- **msg**
  - type: string
  - usage: mandatory
  - default: none
  - This parameter specifies the string to print to the program log.

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if the **msg** parameter contains errors.

Usage:

```
yangcli andy@myserver>log-debug 'starting strict foo'

Debug: starting script foo

yangcli andy@myserver>
```

2.4.32 LOG-ERROR

The **log-error** command prints a message to the program log, if the **$$log-level** system variable is set to 'error' or a higher enumeration (e.g., 'info').

The **msg** parameter is used to provide the message string to print.

log-error command

Command type:	local
Default parameter:	msg
Min parameters:	1
Max parameters:	1
Return type:	status
YANG file:	yangcli.yang

Command Parameters:

- **msg**
  - type: string
  - usage: mandatory
  - default: none
  - This parameter specifies the string to print to the program log.

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if the **msg** parameter contains errors.

Usage:

```
yangcli andy@myserver>log-error 'sysName not correct'

Error: sysName not correct

yangcli andy@myserver>
```

2.4.33 LOG-INFO

The **log-info** command prints a message to the program log, if the **$$log-level** system variable is set to 'info' or a higher enumeration (e.g., 'debug').

The **msg** parameter is used to provide the message string to print.

log-info command

Command type:	local
Default parameter:	msg
Min parameters:	1
Max parameters:	1
Return type:	status
YANG file:	yangcli.yang

Command Parameters:

- **msg**
  - type: string
  - usage: mandatory
  - default: none
  - This parameter specifies the string to print to the program log.

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if the **msg** parameter contains errors.

Usage:

```
yangcli andy@myserver>log-info 'sysName is correct'

Info: sysName is correct

yangcli andy@myserver>
```

2.4.34 LOG-WARN

The **log-warn** command prints a message to the program log, if the **$$log-level** system variable is set to 'warn' or a higher enumeration (e.g., 'debug').

The **msg** parameter is used to provide the message string to print.

log-warn command

Command type:	local
Default parameter:	msg
Min parameters:	1
Max parameters:	1
Return type:	status
YANG file:	yangcli.yang

Command Parameters:

- **msg**
  - type: string
  - usage: mandatory
  - default: none
  - This parameter specifies the string to print to the program log.

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if the **msg** parameter contains errors.

Usage:

```
yangcli andy@myserver>log-warn 'sysName object not found'

Warning: sysName object not found

yangcli andy@myserver>
```

2.4.35 MERGE

The **merge** command is a high-level <edit-config> operation. It is used to merge some new nodes into the default target database.

A target node is specified (in 1 of 2 ways), and then the data structure is filled in. Only mandatory nodes will be filled in unless the **$$optional** system variable is set to 'true'.

Refer to the **fill** command for more details on interactive mode data structure completion.

merge command

Command type:	remote
Default parameter:	target
Min parameters:	1
Max parameters:	5
Return type:	status
YANG file:	yangcli.yang
Capabilities needed:	:candidate or :writable-running

Command Parameters:

- **choice 'from'** (not entered)
  - type: choice of case 'varref' or case 'from-cli'
  - usage: mandatory
  - default: none
  - This parameter specifies the where **yangcli** should get the data from, for the merge operation. It is either a user variable or from interactive input at the command line.
    - **varref**
      - type: string
      - usage: mandatory
      - default: none
      - This parameter specifies the name of the user variable to use for the target of the merge operation. The parameter must exist (e.g., created with the **fill** command) or an error message will be printed.
    - **case from-cli** (not entered)
      - **target**
        - type: string
        - usage: optional (target or urltarget must be entered)
        - default: none
        - This parameter specifies the database target node of the merge operation. This is an **ncx:schema-instance** string, so any instance identifier, or absolute path expression, or something in between, is accepted.
      - **urltarget**
        - type: string
        - usage: optional
        - default: none
        - This parameter specifies the database target node of the merge operation. This is an **UrlPath** string.
      - **optional**

        - type: empty
        - usage: optional
        - default: controlled by **$$optional** system variable
        - This parameter controls whether optional nodes within the target will be filled in. It can be used to override the **$$optional** system variable, when it is set to 'false'.
    - **value**
        - type: anyxml
        - usage: mandatory
        - default: none
        - This parameter specifies the value that should be used for the contents of the **target** parameter. If it is entered, then the interactive mode prompting for missing parameters will be skipped, if this parameter is complete (or all mandatory nodes present if the **$$optional** system variable is 'false'). For example, if the target is a leaf, then specifying this parameter will always cause the interactive prompt mode to be skipped.
- **timeout**
    - type: uint32 (0 = no timeout, otherwise the number of seconds to wait)
    - usage: optional
    - default: set to the **$$timeout** system variable, default 30 seconds
    - This parameter specifies the number of seconds to wait for a response from the server before giving up. The session will not be dropped if a remote command times out, but any late response will be dropped. A value of zero means no timeout should be used, and **yangcli** will wait forever for a response.

System Variables:

- **$$default-operation**
    - The <default-operation> parameter for the <edit-config> operation will be derived from this variable.
- **$$error-option**
    - The <error-option> parameter for the <edit-config> operation will be derived from this variable
- **$$test-option**
    - The <test-option> parameter for the <edit-config> operation will be derived from this variable

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Usage:

```
yangcli andy@myserver> merge \
       target=/nacm/rules/moduleRule[moduleName='netconf']\
       [allowed-rights='read write']/allowed-group \
       value=ncx:guest


(no user prompting; <edit-config> request sent right away)

RPC OK Reply 31 for session 10:

yangcli andy@myserver>
```

Reference:

- RFC 4741, section 7.2

2.4.36 MGRLOAD

The **mgrload** command is used to load a YANG module into **yangcli**.

The YANG files must be available in the module search path for **yangcli**.

mgrload command

Command type:	local
Default parameter:	module
Min parameters:	1
Max parameters:	3
Return type:	status
YANG file:	yangcli.yang

Command Parameters:

- module
  - type: string (length 1 .. 63)
  - usage: mandatory
  - default: none
  - This parameter specifies the name of the module to load.
- revision
  - type: date string (YYYY-MM-DD)
  - usage: optional
  - default: none
  - This parameter specifies the revision date of the module to load.
- deviation:
  - type: leaf-list of deviation module names
  - usage: optional (0 or more instances)

- default: none
- This parameter specifies a deviation module to load prior to loading the requested module.

Positive Response:

- OK

Negative Response:

- An error message will be printed if errors are detected.

Usage:

```
yangcli> mgrload toaster

Load module toaster OK

yangcli>
```

2.4.37 NO-OP

The **no-op** command is used to test server message processing response times, by providing a baseline response time to do nothing except return <ok/>.

It can also be used as an application-level keep-alive to prevent proxy idle timeout or server idle timeout problems from occurring.

This command is only supported by the **netconfd** server. The server will simply respond 'OK', if the request is well-formed.

no-op command

Command type:	remote
Default parameter:	none
Min parameters:	0
Max parameters:	0
Return type:	status
YANG file:	yuma-system.yang

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Usage:

```
yangcli andy@myserver> no-op

RPC OK Reply 31 for session 10:

yangcli andy@myserver>
```

2.4.38 PWD

The **pwd** command is used to print the current working directory.

pwd command

Command type:	local
Default parameter:	none
Min parameters:	0
Max parameters:	0
Return type:	status
YANG file:	yangcli.yang

Positive Response:

- the current working directory is printed to the log or STDOUT

Negative Response:

- An error message will be printed if errors are detected.

Usage:

```
yangcli> pwd

Current working directory is /home/andy

yangcli>
```

2.4.39 QUIT

The **quit** command is used to terminate the **yangcli** program.

If a NETCONF session is in progress, it will be dropped without sending a **close-session** command first. This should be taken into account if the server reports dropped TCP connections as an error.

quit command

Command type:	local
Default parameter:	none
Min parameters:	0
Max parameters:	0
Return type:	none
YANG file:	yangcli.yang

Positive Response:

- The program terminates; no response will be printed.

Negative Response:

- An error message will be printed if errors are detected.

Usage:

```
yangcli> quit

andy@myworkstation>
```

2.4.40 RECALL

The **recall** command is used to recall a previously entered command line from the command line history buffer.

A command is recalled by its command ID number. Use the **history show** command to see the contents of the command line history buffer.

recall command

Command type:	local
Default parameter:	index
Min parameters:	1
Max parameters:	1
Return type:	data
YANG file:	yangcli.yang

Positive Response:

- The specified command line is recalled into the current command line.

Negative Response:

- An error message will be printed if errors are detected.

Usage:

```
yangcli> recall 7

yangcli>sget /system
```

2.4.41 RELEASE-LOCKS

The **release-locks** command is used to release all the locks that were previously granted with the get-locks command.

If the get-locks command was not used, then this command will fail with an error message that no locks are active to unlock.

release-locks command

Command type:	remote
Default parameter:	none
Min parameters:	0
Max parameters:	0
Return type:	status
YANG file:	yangcli.yang

Positive Response:

- The previously granted locks are released.

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Usage: (showing 2 locks being released)

```
yangcli ady@myserver> release-locks

RPC OK Reply 8 for session 23:
RPC OK Reply 9 for session 23:

yangcli andy@myserver>
```

2.4.42 REPLACE

The **replace** command is a high-level <edit-config> operation. It is used to replace an existing subtree with some new nodes, in the default target database.

Only the subtree indicated by the **target** or **varref** parameter will be replaced.

A target node is specified (in 1 of 2 ways), and then the data structure is filled in. Only mandatory nodes will be filled in unless the **$$optional** system variable is set to 'true'.

Refer to the **fill** command for more details on interactive mode data structure completion.

replace command

Command type:	remote
Default parameter:	target
Min parameters:	1
Max parameters:	5
Return type:	status

YANG file:	yangcli.yang
Capabilities needed:	:candidate or :writable-running

Command Parameters:

- **choice 'from'** (not entered)
  - type: choice of case 'varref' or case 'from-cli'
  - usage: mandatory
  - default: none
  - This parameter specifies the where **yangcli** should get the data from, for the replace operation. It is either a user variable or from interactive input at the command line.
    - **varref**
      - type: string
      - usage: mandatory
      - default: none
      - This parameter specifies the name of the user variable to use for the target of the replace operation. The parameter must exist (e.g., created with the **fill** command) or an error message will be printed.
    - **case from-cli** (not entered)
      - **target**
        - type: string
        - usage: optional (target or urltarget must be present)
        - default: none
        - This parameter specifies the database target node of the replace operation. This is an **ncx:schema-instance** string, so any instance identifier, or absolute path expression, or something in between, is accepted.
      - **urltarget**
        - type: string
        - usage: optional
        - default: none
        - This parameter specifies the database target node of the replace operation. This is an **UrlPath** string.
      - **optional**
        - type: empty
        - usage: optional
        - default: controlled by **$$optional** system variable
        - This parameter controls whether optional nodes within the target will be filled in. It can be used to override the **$$optional** system variable, when it is set to 'false'.
      - **value**
        - type: anyxml

- usage: mandatory
- default: none
- This parameter specifies the value that should be used for the contents of the **target** parameter. If it is entered, then the interactive mode prompting for missing parameters will be skipped, if this parameter is complete (or all mandatory nodes present if the **$$optional** system variable is 'false'). For example, if the target is a leaf, then specifying this parameter will always cause the interactive prompt mode to be skipped.

- **timeout**
  - type: uint32 (0 = no timeout, otherwise the number of seconds to wait)
  - usage: optional
  - default: set to the **$$timeout** system variable, default 30 seconds
  - This parameter specifies the number of seconds to wait for a response from the server before giving up. The session will not be dropped if a remote command times out, but any late response will be dropped. A value of zero means no timeout should be used, and **yangcli** will wait forever for a response.

System Variables:

- **$$default-operation**
  - The <default-operation> parameter for the <edit-config> operation will be derived from this variable.
- **$$error-option**
  - The <error-option> parameter for the <edit-config> operation will be derived from this variable
- **$$test-option**
  - The <test-option> parameter for the <edit-config> operation will be derived from this variable

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Usage:

```
        yangcli andy@myserver> replace \
                target=/nacm/rules/moduleRule[moduleName='yuma-
system']\
                [allowed-rights='exec']

        (user prompted to fill in specified <moduleRule> element)


        RPC OK Reply 31 for session 10:

        yangcli andy@myserver>
```

Reference:

- RFC 4741, section 7.2

2.4.43 RESTART

The **restart** command is used to restart the NETCONF server.

This command is only supported by the **netconfd** server.

The default access control configuration for **netconfd** will not allow any user except the designated 'superuser' account to invoke this command. Refer to the **netconfd** user manual for details on configuring access control.

restart command

Command type:	remote
Default parameter:	none
Min parameters:	0
Max parameters:	0
Return type:	status
YANG file:	yuma-system.yang

Positive Response:

- the server will drop all sessions and restart

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Usage:

```
yangcli andy@myserver> restart

ses: session 10 shut by remote peer

yangcli>
```

2.4.44 RUN

The run command is used to run a **yangcli** script.

Refer to the section on scripts for details on how to write a script.

The script name is the only mandatory parameter. There are 9 generic parameters (named P1 to P9) that can be used to pass string parameters to scripts. Within a script, these are referenced with local variables ($1 to $9).

The run command can be used within a script.

Scripts do not return any status or data at this time.

The run command can appear inside a script, starting a new run level. An error will occur if a loop occurs or too many nest levels are used. Up to 64 run levels are supported in **yangcli**.

run command

Command type:	local
Default parameter:	script
Min parameters:	1
Max parameters:	10
Return type:	status
YANG file:	yangcli.yang

Command Parameters:

- **P1, P2, P3, P4, P5, P6, P7, P8, P9**
  - type: string
  - usage: optional
  - default: none
  - These parameters provide a generic string parameter passing mechanism for each script invocation, similar to unix shell scripts. Within the script, the parameters **$1**, **$2, $3**, **$4**, **$5**, **$6**, **$7**, **$8** and **$9** will be non-NULL only if the corresponding **'Pn'** parameter was set in the script invocation.
- **script**
  - type: string
  - usage: mandatory
  - default: none
  - This parameter specifis the name of the script to be run. If a path specification is included, then it will be used. Otherwise the current script search path will be used to find the script.

System Variables:

- **$$runpath**
  - Controls where **yangcli** will look for files specified in the **script** parameter.

Positive Response:

- the specified script will be executed

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error, if any remote commands are contained in the script.

Usage:

```
yangcli> run connect P1=yangrocks
```

```
runstack: Starting level 1 script /home/andy/scripts/connect

(commands and responses are printed as if they were typed)

runstack: Ending level 1 script /home/any/scripts/connect
yangcli andy@myserver>
```

2.4.45 SAVE

The **save** command is used to save NETCONF database edits to non-volatile storage, on the server. It is a pseudo-command, mapped to other remote commands depending on which database target is supported by the server (<candidate> or <running>).

The **save** command usually maps to either the **commit** or the **copy-config** command. Refer to the section on NETCONF sessions for more details.

save command

Command type:	remote
Default parameter:	none
Min parameters:	0
Max parameters:	0
Return type:	none
YANG file:	yangcli.yang

Positive Response:

- The server returns one or two <ok/> responses.

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Usage:

```
yangcli andy@myserver> save

RPC OK Reply 34 on session 10

yangcli andy@myserver>
```

2.4.46 SET-LOG-LEVEL

The **set-log-level** command is used to configure the server logging verbosity level. It is only supported by the **netconfd** server.

This operation is defined as **nacm:secure**, so only the system super-user can invoke this command by default.

set-log-level command

Command type:	remote
Default parameter:	none
Min parameters:	1
Max parameters:	1
Return type:	status
YANG file:	yuma-system.yang
Capabilities needed:	none

Command Parameters:

- **log-level**
  - type: enumeration (off, error, warn, info, debug, debug2, debug3)
  - This mandatory parameter specifies the desired server logging verbosity level.

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.]

```
yangcli andy@myserver> set-log-level log-level=debug2

RPC OK Reply 29 for session 10:

yangcli andy@myserver>
```

2.4.47 SET-MY-SESSION

The **set-my-session** command is used to configure the session customization parameters on the server. It is only supported by the **netconfd** server.

The session line size and default behavior for the with-defaults parameter can be controlled at this time.

Since all parameters are optional, they need to be entered in the same command line as the command name. Otherwise the **$$optional** system variable needs to be set to 'true'. If not, then no parameters will be sent to the server, and no session parameters will be changed.

set-my-session command

Command type:	remote

Default parameter:	none
Min parameters:	0
Max parameters:	3
Return type:	status
YANG file:	yuma-mysession.yang
Capabilities needed:	none

Command Parameters:

- **indent**
  - type: uint32 (range: 0 .. 9)
  - This parameter specifies the desired number of spaces to indent for each new XML nest level, for the session. If missing, then the indent amount is not changed.
- **linesize**
  - type: uint32 (range: 40 .. 1024)
  - This parameter specifies the desired line length for the session. If missing, then the current line size is not changed.
- **with-defaults**
  - type: enumeration (report-all trim explicit)
  - This parameter specifies the desired default with-defaults filtering behavior for the session. If missing, the current with-defaults behavior is not changed.

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.]

```
yangcli andy@myserver> set-my-session \
       --linesize=64 --with-defaults=trim

RPC OK Reply 25 for session 10:

yangcli andy@myserver>
```

2.4.48 SGET

The **sget** command is used to invoke a NETCONF <get> operation with a subtree filter.

A target node is specified (in 1 of 2 ways), and then the data structure is filled in. Only mandatory nodes will be filled in unless the **$$optional** system variable is set to 'true'. This step can be skipped completely with the **nofill** parameter.

Refer to the **fill** command for more details on interactive mode data structure completion.

sget command

Command type:	remote
Default parameter:	target
Min parameters:	1
Max parameters:	5
Return type:	data
YANG file:	yangcli.yang
Capabilities needed:	none
Capabilities optional:	:with-defaults

Command Parameters:

- **choice 'from'** (not entered)
  - type: choice of case 'varref' or case 'from-cli'
  - usage: mandatory
  - default: none
  - This parameter specifies the where **yangcli** should get the data from, for the subtree filter target of the <get> operation. It is either a user variable or from interactive input at the command line.
    - **varref**
      - type: string
      - usage: mandatory
      - default: none
      - This parameter specifies the name of the user variable to use for the subtree filter target of the <get> operation. The parameter must exist (e.g., created with the **fill** command) or an error message will be printed.
    - **case from-cli** (not entered)
      - **target**
        - type: string
        - usage: optional (target or urltarget must be present)
        - default: none
        - This parameter specifies the database target node of the <get> operation. This is an **ncx:schema-instance** string, so any instance identifier, or absolute path expression, or something in between, is accepted.
          - The escape command ('?s') can be used in parameter mode to skip a parameter completely.
          - Entering the empty string for a parameter will cause a subtree selection node to be created instead of a content match node
      - **urltarget**
        - type: string
        - usage: optional

        - default: none
        - This parameter specifies the database target node of the get operation. This is an **UrlPath** string.
    - **optional**
        - type: empty
        - usage: optional
        - default: controlled by **$$optional** system variable
        - This parameter controls whether optional nodes within the target will be filled in. It can be used to override the **$$optional** system variable, when it is set to 'false'.
    - **value**
        - type: anyxml
        - usage: mandatory
        - default: none
        - This parameter specifies the value that should be used for the contents of the **target** parameter. If it is entered, then the interactive mode prompting for missing parameters will be skipped, if this parameter is complete (or all mandatory nodes present if the **$$optional** system variable is 'false'). For example, if the target is a leaf, then specifying this parameter will always cause the interactive prompt mode to be skipped.
- **nofill**
    - type: empty
    - usage: optional
    - default: none
    - This parameter specifies the that no interactive prompting to fill in the contents of the specified subtree filter target will be done.
        - This causes the entire selected subtree to be requested in the filter.
        - yangcli will attempt to figure out if there can be multiple instances of the requested subtree. If not, then **nofill** will be the default for that target parameter.
- **with-defaults**
    - type: enumeration (none report-all report-all-tagged trim explicit)
    - usage: optional
    - default: none
    - capabilities needed: with-defaults
    - This parameter controls how nodes containing only default values are returned in the <rpc-reply>.

System Variables:

- **$$timeout**
    - The response timeout interval will be derived from this system variable.
- **$$with-defaults**
    - The <with-defaults> parameter for the <get> operation will be derived from this system variable.

Positive Response:

- the server returns <data>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Output:

- **data**
  - type: anyxml
  - This element will contain the requested data from the <running> database or non-config data structures.

Usage:

```
yangcli andy@myserver> sget sytem

Filling container /system:
RPC Data Reply 32 for session 10:

rpc-reply {
   data {
      system {
         sysName myserver.localdomain
         sysCurrentDateTime 2009-07-06T02:24:19Z
         sysBootDateTime 2009-07-05T19:22:28Z
      }
   }
}

yangcli andy@myserver>
```

Reference:

- RFC 4741, section 7.7

2.4.49 SGET-CONFIG

The **sget-config** command is used to invoke a NETCONF <get-config> operation with a subtree filter.

A target node is specified (in 1 of 2 ways), and then the data structure is filled in. Only mandatory nodes will be filled in unless the **$$optional** system variable is set to 'true'. This step can be skipped completely with the **nofill** parameter.

Refer to the **fill** command for more details on interactive mode data structure completion.

sget-config command

Command type:	remote
Default parameter:	target
Min parameters:	2

Max parameters:	6
Return type:	data
YANG file:	yangcli.yang
Capabilities needed:	none
Capabilities optional:	:candidate :startup :url :with-defaults

Command Parameters:

- **choice 'from'** (not entered)
  - type: choice of case 'varref' or case 'from-cli'
  - usage: mandatory
  - default: none
  - This parameter specifies the where **yangcli** should get the data from, for the subtree filter target of the <get> operation. It is either a user variable or from interactive input at the command line.
    - **varref**
      - type: string
      - usage: mandatory
      - default: none
      - This parameter specifies the name of the user variable to use for the subtree filter target of the <get> operation. The parameter must exist (e.g., created with the **fill** command) or an error message will be printed.
    - **case from-cli** (not entered)
      - **target**
        - type: string
        - usage: optional (target or urltarget must be present)
        - default: none
        - This parameter specifies the database target node of the <get> operation. This is an **ncx:schema-instance** string, so any instance identifier, or absolute path expression, or something in between, is accepted.
          - The escape command ('?s') can be used in parameter mode to skip a parameter completely.
          - Entering the empty string for a parameter will cause a subtree selection node to be created instead of a content match node
      - **urltarget**
        - type: string
        - usage: optional
        - default: none
        - This parameter specifies the database target node of the get-config operation. This is an **UrlPath** string.

        - **optional**
            - type: empty
            - usage: optional
            - default: controlled by **$$optional** system variable
            - This parameter controls whether optional nodes within the target will be filled in. It can be used to override the **$$optional** system variable, when it is set to 'false'.
        - **value**
            - type: anyxml
            - usage: mandatory
            - default: none
            - This parameter specifies the value that should be used for the contents of the **target** parameter. If it is entered, then the interactive mode prompting for missing parameters will be skipped, if this parameter is complete (or all mandatory nodes present if the **$$optional** system variable is 'false'). For example, if the target is a leaf, then specifying this parameter will always cause the interactive prompt mode to be skipped.

- **nofill**
    - type: empty
    - usage: optional
    - default: none
    - This parameter specifies the that no interactive prompting to fill in the contents of the specified subtree filter target will be done.
        - This causes the entire selected subtree to be requested in the filter.
        - yangcli will attempt to figure out if there can be multiple instances of the requested subtree. If not, then **nofill** will be the default for that target parameter.
- **source**
    - type: container with 1 of N choice of leafs
    - usage: mandatory
    - default: none
    - This parameter specifies the name of the source database for the retrieval operation.
    - container contents: 1 of N:
        - **candidate**
            - type: empty
            - capabilities needed: :candidate
        - **running**
            - type: empty
            - capabilities needed: none
        - **startup**
            - type: empty
            - capabilities needed: startup

        - **url**
            - type: yang:uri
            - capabilities needed: :url, and the scheme used in the URL must be specified in the :url capability 'schemes' parameter
            - To enter this parameter, the interactive mode must be used. The shorthand mode (source=url) cannot be used, since this parameter contains a string.
- **with-defaults**
    - type: enumeration (none report-all report-all-tagged trim explicit)
    - usage: optional
    - default: none
    - capabilities needed: with-defaults
    - This parameter controls how nodes containing only default values are returned in the <rpc-reply>.

System Variables:

- **$$timeout**
    - The response timeout interval will be derived from this system variable.
- **$$with-defaults**
    - The <with-defaults> parameter for the <get-config> operation will be derived from this system variable.

Positive Response:

- the server returns <data>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Output:

- **data**
    - type: anyxml
    - This element will contain the requested data from the database indicated by the **source** parameter.

Usage:

```
yangcli andy@myserver> sget-config /nacm/rules/data-rule \
       source=candidate \
       nofill

RPC Data Reply 35 for session 10:

rpc-reply {
   data {
     nacm {
         rules {
            data-rule nacm-tree {
               name nacm-tree
               path /nacm:nacm
```

```
                     allowed-rights {
                        read
                        write
                     }
                     allowed-group nacm:admin
                     comment 'access to nacm config'
                  }
               }
            }
         }
      }

      yangcli andy@myserver>
```

Reference:

- RFC 4741, section 7.1

2.4.50 show

The **show** command is used to show program status and YANG details that may be needed during management of a NETCONF server.

There are several variants of the **show** command:

- The **show cli** command prints the contents of the yangcli CLI and config file parameters.
- The **show global** command prints the contents of one global user variable.
- The **show globals** command prints a summary or the contents of all the global user variables.
- The **show local** command prints the contents of one local user variable.
- The **show locals** command prints a summary or the contents of all the local user variables.
- The **show module** command prints meta information or help text for one YANG module.
- The **show modules** command prints meta information for all the currently available YANG modules. IF a session is active, this will be the list of modules the server reported, plus any modules loaded with the **mgrload** command.
- The **show objects** command prints the available objects or help text for the available objects.
- The **show session** command prints the session startup screen information for the current session.
- The **show system** command prints the contents of the yangcli system variables.
- The **show var** command prints the contents of the specified variable.
- The **show vars** command prints a summary or the contents of all the program variables.
- The **show version** command prints the **yangcli** version number

show command

Command type:	local
Default parameter:	none
Min parameters:	1
Max parameters:	2

Return type:	data
YANG file:	yangcli.yang

Command Parameters:

- **choice help-mode** (not entered) (default choice is 'normal')
  - **brief**
    - type: empty
    - usage: optional
    - default: none
    - This parameter specifies that brief documentation mode should be used.
  - **normal**
    - type: empty
    - usage: optional
    - default: none
    - This parameter specifies that normal documentation mode should be used.
  - **full**
    - type: empty
    - usage: optional
    - default: none
    - This parameter specifies that full documentation mode should be used.
- **choice showtype** (not entered)
  - mandatory 1 of N choice for the sub-command for the **show** command
    - **cli**
      - type: empty
      - usage: mandatory
      - default: none
      - This parameter selects the yangcli CLI and configuration variables.
        - The **help-mode** parameter has no affect on this command.
    - **global**
      - type: string
      - usage: mandatory
      - default: none
      - This parameter specifies the name of the global variable to show information for.
        - If **help-mode** is 'brief', then the name, variable object, and type of the global variable object will be printed.
        - If **help-mode** is 'normal' or 'full', then the contents of the specified global variable will be printed.
    - **globals**

    - type: empty
    - usage: mandatory
    - default: none
    - This parameter specifies that information for all global variables should be printed.
        - If **help-mode** is 'brief', then the name, variable object, and type of each global variable object will be printed.
        - If **help-mode** is 'normal' or 'full', then the contents of each global variable will be printed.
- **local**
    - type: string
    - usage: mandatory
    - default: none
    - This parameter specifies the name of the local variable to show information for.
        - If **help-mode** is 'brief', then the name, variable object, and type of the local variable object will be printed.
        - If **help-mode** is 'normal' or 'full', then the contents of the specified local variable will be printed.
- **locals**
    - type: empty
    - usage: mandatory
    - default: none
    - This parameter specifies that information for all local variables should be printed.
        - If **help-mode** is 'brief', then the name, variable object, and type of each local variable object will be printed.
        - If **help-mode** is 'normal' or 'full', then the contents of each local variable will be printed.
- **module**
    - type: string
    - usage: mandatory
    - default: none
    - This parameter specifies the name of the module to show information for.
        - If **help-mode** is 'brief' or 'normal', then the name, version, namespace, and source of the module will be printed.
        - If **help-mode** is 'full', then the module level help text for the specified module will be printed.
- **modules**
    - type: empty
    - usage: mandatory
    - default: none
    - This parameter specifies that information for all available modules should be printed:

      - If **help-mode** is 'brief', then the name of each module will be printed.
      - If **help-mode** is 'normal', then the XML prefix, name, and revision date of each module will be printed.
      - If **help-mode** is 'full', then the name, revision date, prefix, namespace, and source of each module will be printed.
- **objects**
  - type: empty
  - usage: mandatory
  - default: none
  - This parameter specifies that information for all available database objects should be printed:
    - If **help-mode** is 'brief', then the module name and name of each object will be printed.
    - If **help-mode** is 'normal', then the YANG object type, object name, and YANG base type will be printed.

      If **help-mode** is 'full', then brief help text will be printed for every object.
- **system**
  - type: empty
  - usage: mandatory
  - default: none
  - This parameter specifies that information for all system variables should be printed.
    - The **help-mode** parameter has no affect on this command.
- **var**
  - type: string
  - usage: mandatory
  - default: none
  - This parameter specifies the name of any variable to show information for.
    - If **help-mode** is 'brief', then the name, variable object, and type of the variable object will be printed.
    - If **help-mode** is 'normal' or 'full', then the contents of the specified variable will be printed.
- **vars**
  - type: empty
  - usage: mandatory
  - default: none
  - This parameter specifies that information for all variables should be printed. This includes all CLI, read-only system, read-write system, global user, and local user variables.
    - If **help-mode** is 'brief', then the name, variable object, and type of each variable object will be printed.

    - If **help-mode** is 'normal' or 'full', then the contents of each variable will be printed.

- **version**
  - type: empty
  - usage: mandatory
  - default: none
  - This parameter specifies that the yangcli program version string should be printed.
    - The **help-mode** parameter has no affect in this mode.

Positive Response:

- the server prints the requested information

Negative Response:

- An error message will be printed if errors are detected.

Usage:

```
yangcli andy@myserver> show modules

  nc:netconf/2009-04-10
  inet:ietf-inet-types/2009-05-13
  ns:ietf-netconf-monitoring/2009-03-03
  wd:ietf-with-defaults/2009-04-10
  yang:ietf-yang-types/2009-05-13
  nacm:nacm/2009-05-13
  manageEvent:nc-notifications/2008-07-14
  ncx:ncx/2009-06-12
  ncxapp:ncx-app-common/2009-04-10
  nt:ncxtypes/2008-07-20
  nd:netconfd/2009-05-28
  ncEvent:notifications/2008-07-14
  sys:system/2009-06-04
  t:test/2009-06-10

yangcli andy@myserver>
```

2.4.51 SHUTDOWN

The **shutdown** command is used to shut down the NETCONF server.

This command is only supported by the **netconfd** server.

The default access control configuration for **netconfd** will not allow any user except the designated 'superuser' account to invoke this command. Refer to the **netconfd** user manual for details on configuring access control.

shutdown command

Command type:	remote
Default parameter:	none
Min parameters:	0

Max parameters:	0
Return type:	status
YANG file:	yuma-system.yang

Positive Response:

- the server will drop all sessions and terminate.

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Usage:

```
yangcli andy@myserver> shutdown

ses: session 10 shut by remote peer

yangcli>
```

2.4.52 START-TIMER

The **start-timer** command is used to start a timer to do simple performance measurements.
It is used along with the stop-timer command.

start-timer command

Command type:	local
Default parameter:	id
Min parameters:	1
Max parameters:	2
Return type:	status
YANG file:	yangcli.yang
Capabilities needed:	none
Capabilities optional:	none

Command Parameters:

- **id**
  - type: number
  - usage: mandatory
  - default: 0
  - This parameter specifies the numeric ID of the timer.
    It is a number in the range 0 to 15.

- **restart-ok**
  - type: boolean
  - usage: optional
  - default: true
  - Indicates whether the timer will be used if it is already running. If 'true', the timer will be restarted if it is already running. If 'false', an error will occur if the timer is already running.
  -

Positive Response:

- the client prints OK

Negative Response:

- An error message will be printed if errors are detected locally.

Usage:

```
yangcli andy@myserver> start-timer 1

OK

yangcli andy@myserver>
```

2.4.53 STOP-TIMER

The **stop-timer** command is used to stop a timer to do simple performance measurements.

It is used along with the start-timer command.

stop-timer command

Command type:	local
Default parameter:	id
Min parameters:	1
Max parameters:	2
Return type:	data
YANG file:	yangcli.yang
Capabilities needed:	none
Capabilities optional:	none

Command Parameters:

- **id**
  - type: number
  - usage: mandatory
  - default: 0

   - This parameter specifies the numeric ID of the timer.
     It is a number in the range 0 to 15.
- **echo**
   - type: boolean
   - usage: optional
   - default: true
   - Indicates whether the delta result should be printed to the log.

Positive Response:

- data: the elapsed time is returned and can be used in assignment statements

Negative Response:

- An error message will be printed if errors are detected locally.

Usage:

```
yangcli andy@myserver> $time = stop-timer 1

Elapsed time: 2.003345 seconds

yangcli andy@myserver>
```

2.4.54 UNLOCK

The **unlock** command is used to request a global lock on a NETCONF database. It is used, along with the **lock** command, to obtain exclusive write access to the NETCONF server.

unlock command

Command type:	remote
Default parameter:	none
Min parameters:	1
Max parameters:	1
Return type:	status
YANG file:	yuma-netconf.yang
Capabilities needed:	none
Capabilities optional:	:candidate :startup :url

Command Parameters:

- **target**
   - type: container with 1 of N choice of leafs

- usage: mandatory
- default: none
- This parameter specifies the name of the target database to be locked.
- container contents: 1 of N:
  - **candidate**
    - type: empty
    - capabilities needed: :candidate
  - **running**
    - type: empty
    - capabilities needed: none
  - **startup**
    - type: empty
    - capabilities needed: startup
  - **url**
    - type: yang:uri
    - capabilities needed: :url, and the scheme used in the URL must be specified in the :url capability 'schemes' parameter.
    - To enter this parameter, the interactive mode must be used. The shorthand mode (target=url) cannot be used, since this parameter contains a string.

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Usage:

```
yangcli andy@myserver> unlock target=candidate

RPC OK Reply 38 for session 10:

yangcli andy@myserver>
```

2.4.55 VALIDATE

The **validate** command is used to check if a complete NETCONF database is valid or not.

The **:validate** capability must be supported by the server to use this command.

validate command

Command type:	remote

Default parameter:	none
Min parameters:	1
Max parameters:	1
Return type:	status
YANG file:	yuma-netconf.yang
Capabilities needed:	:validate
Capabilities optional:	:candidate :startup :url

Command Parameters:

- **target**
  - type: container with 1 of N choice of leafs
  - usage: mandatory
  - default: none
  - This parameter specifies the name of the target database to be validated.
  - container contents: 1 of N:
    - **candidate**
      - type: empty
      - capabilities needed: :candidate
    - **running**
      - type: empty
      - capabilities needed: none
    - **startup**
      - type: empty
      - capabilities needed: startup
    - **url**
      - type: yang:uri
      - capabilities needed: :url, and the scheme used in the URL must be specified in the :url capability 'schemes' parameter.
      - To enter this parameter, the interactive mode must be used. The shorthand mode (target=url) cannot be used, since this parameter contains a string.
    - **config**
      - type: anyxml
      - capabilities
      - This parameter represents an entire database, using the in-line config form, similar to the **edit-config** command, except this config parameter contains no nc:operation attributes and must be a complete database, not a subset.

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Usage:

```
yangcli andy@myserver> validate source=candidate

RPC OK Reply 36 for session 10:

yangcli andy@myserver>
```

2.4.56 WHILE

The **while** command is used to start a conditional loop command block.

The **expr** parameter is used to specify the XPath expression to test if the while conditional loop block is true or false. A false block will be skipped and a true block will be executed. The command prompt will contain the string '[F]' while inside a false conditional block in interactive mode. This expression string should be entered with quotes to avoid misinterpreting any whitespace or special characters.

The **docroot** parameter (if present) specifies the XML document that the 'expr' parameter should be evaluated against. This is optional, since only XPath path expressions need to refer to a document.

If the 'expr' expression is true, the conditional block will be executed, and no further conditional blocks within the same if command sequence will be executed.

The maxloops parameter specifies the maximum number of iterations that should be allowed, even if the expression continues to evaluate to 'true'. This allows constant expressions to be used to construct a simple 'for' loop.

```
while command
...
end command
```

while command

Command type:	local
Default parameter:	expr
Min parameters:	1
Max parameters:	3
Return type:	status
YANG file:	yangcli.yang

Command Parameters:

- **expr**

  - type: XPath expression string
  - usage: mandatory
  - default: none
  - This parameter specifies the XPath expression to determine if the following commands are within a true or false conditional block.
- **docroot**
  - type: anyxml
  - usage: optional (typically a variable reference is used)
  - default: none
  - This parameter specifies the XML document that should be used if the expr XPath expression references any path nodes.
- **maxloops**
  - type: uint32
  - usage: optional
  - default: 65535
  - Set a maximum number of loops that can be iterated for this while command. The value zero means that no maximum will be enforced. Use this mode with caution.

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if errors are detected locally.
  - invalid XPath expression or invalid docroot reference will cause an error

Usage:

```
yangcli andy@myserver> while 1 --maxloops=10

yangcli andy@myserver> while '$x < 10'

yangcli andy@myserver>
```

2.4.57 XGET

The **xget** command is used to invoke a NETCONF <get> operation with an XPath filter.

xget command

Command type:	remote
Default parameter:	select

Min parameters:	1
Max parameters:	3
Return type:	data
YANG file:	yangcli.yang
Capabilities needed:	none
Capabilities optional:	:with-defaults

Command Parameters:

- **choice 'from'** (not entered)
  - type: choice of case 'varref' or case 'select'
  - usage: mandatory
  - default: none
  - This parameter specifies the where **yangcli** should get the data from, for the XPath filter target of the <get> operation. It is an XPath expression, either contained in a user variable or directly from the **select** parameter.
    - **varref**
      - type: string
      - usage: mandatory
      - default: none
      - This parameter specifies the name of the user variable the contains the XPath expression for the XPath filter in the <get> operation. The parameter must exist (e.g., created with an assignment statement) or an error message will be printed.
    - **select**
      - type: string
      - usage: mandatory
      - default: none
      - This parameter specifies the XPath expression to use for the XPath filter in the <get> operation.
- **timeout**
  - type: uint32 (0 = no timeout, otherwise the number of seconds to wait)
  - usage: optional
  - default: set to the **$$timeout** system variable, default 30 seconds
  - This parameter specifies the number of seconds to wait for a response from the server before giving up. The session will not be dropped if a remote command times out, but any late response will be dropped. A value of zero means no timeout should be used, and **yangcli** will wait forever for a response.
- **with-defaults**
  - type: enumeration (none report-all report-all-tagged trim explicit)
  - usage: optional
  - default: none

- capabilities needed: with-defaults
- This parameter controls how nodes containing only default values are returned in the <rpc-reply>.

System Variables:

- **$$timeout**
  - The response timeout interval will be derived from this system variable.
- **$$with-defaults**
  - The <with-defaults> parameter for the <get> operation will be derived from this system variable.

Positive Response:

- the server returns <data>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Output:

- **data**
  - type: anyxml
  - This element will contain the requested data from the <running> database or non-config data structures.

Usage:

```
yangcli andy@myserver> xget //name

RPC Data Reply 42 for session 10:

rpc-reply {
   data {
      nacm {
         rules {
            data-rule nacm-tree {
               name nacm-tree
            }
         }
      }
      xpath.1 {
         name barney
      }
      netconf-state {
         datastores {
            datastore {
               name {
                  candidate
               }
            }
         }
      }
      netconf {
         streams {
            stream NETCONF {
               name NETCONF
```

```
                    }
                }
            }
        }
        }
    }

    yangcli andy@myserver>
```

Reference:

- RFC 4741, section 7.7

2.4.58 XGET-CONFIG

The **xget-config** command is used to invoke a NETCONF <get-config> operation with an XPath filter.

xget-config command

Command type:	remote
Default parameter:	select
Min parameters:	2
Max parameters:	4
Return type:	data
YANG file:	yangcli.yang
Capabilities needed:	none
Capabilities optional:	:candidate :startup :url :with-defaults

Command Parameters:

- **choice 'from'** (not entered)
  - type: choice of case 'varref' or case 'select'
  - usage: mandatory
  - default: none
  - This parameter specifies the where **yangcli** should get the data from, for the XPath filter target of the <get-config> operation. It is an XPath expression, either contained in a user variable or directly from the **select** parameter.
    - **varref**
      - type: string
      - usage: mandatory
      - default: none
      - This parameter specifies the name of the user variable the contains the XPath expression for the XPath filter in the <get-config> operation. The parameter must

        exist (e.g., created with an assignment statement) or an error message will be printed.

    - **select**
        - type: string
        - usage: mandatory
        - default: none
        - This parameter specifies the XPath expression to use for the XPath filter in the <get-config> operation.

- **source**
  - type: container with 1 of N choice of leafs
  - usage: mandatory
  - default: none
  - This parameter specifies the name of the source database for the retrieval operation.
  - container contents: 1 of N:
    - **candidate**
      - type: empty
      - capabilities needed: :candidate
    - **running**
      - type: empty
      - capabilities needed: none
    - **startup**
      - type: empty
      - capabilities needed: startup
    - **url**
      - type: yang:uri
      - capabilities needed: :url, and the scheme used in the URL must be specified in the :url capability 'schemes' parameter
      - To enter this parameter, the interactive mode must be used. The shorthand mode (source=url) cannot be used, since this parameter contains a string.
- **timeout**
  - type: uint32 (0 = no timeout, otherwise the number of seconds to wait)
  - usage: optional
  - default: set to the **$$timeout** system variable, default 30 seconds
  - This parameter specifies the number of seconds to wait for a response from the server before giving up. The session will not be dropped if a remote command times out, but any late response will be dropped. A value of zero means no timeout should be used, and **yangcli** will wait forever for a response.
- **with-defaults**
  - type: enumeration (none report-all report-all-tagged trim explicit)
  - usage: optional

  - default: none
  - capabilities needed: with-defaults
  - This parameter controls how nodes containing only default values are returned in the <rpc-reply>.

System Variables:

- **$$timeout**
  - The response timeout interval will be derived from this system variable.
- **$$with-defaults**
  - The <with-defaults> parameter for the <get-config> operation will be derived from this system variable.

Positive Response:

- the server returns <data>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Output:

- **data**
  - type: anyxml
  - This element will contain the requested data from the **source** database.

Usage:

```
yangcli andy@myserver> xget /nacm/groups \
       source=running

RPC Data Reply 41 for session 10:

rpc-reply {
   data {
      nacm {
         groups {
            group nacm:admin {
               groupIdentity nacm:admin
               userName andy
               userName fred
               userName barney
            }
            group nacm:guest {
               groupIdentity nacm:guest
               userName guest
            }
         }
      }
   }
}

yangcli andy@svnserver>
```

3 CLI Reference

The Yuma programs uses command line interface (CLI) parameters to control program behavior.

Many parameters are supported by more than one program, such as **--log.**

The following sections document all the Yuma CLI parameters, in alphabetical order.

3.1 --alt-names

The –**alt-names** parameter controls whether alternative names are searched when processing a UrlPath search.

--alt-names parameter

Syntax	boolean (true or false)
Default:	TRUE
Min Allowed:	0
Max Allowed:	1
Supported by:	yangcli
Example:	`yangcli --alt-names=false`

3.2 --autocomp

The **--autocomp** parameter controls automatic completion of command and parameter names.

If true, then the program will attempt to find a partial match by comparing only the number of characters entered. For example '--log-l=debug' is the same as '--log-level=debug'.

If 'false', then command and parameter names must match exactly.

--autocomp parameter

Syntax	boolean (true or false)
Default:	TRUE
Min Allowed:	0
Max Allowed:	1
Supported by:	yangcli
Example:	`yangcli --autocomp=false`

3.3 --autohistory

The **--autohistory** parameter controls automatic loading and saving of the command line history buffer in the **yangcli** program..

If true, then when the program starts, the default command line history buffer will be loaded, and the previous set of commands will be available with the **history** and **recall** commands.

If 'false', then the command line history buffer will not be loaded and saved automatically.

The default location for this file is ~/.yuma/.yangcli_history.

--autohistory parameter

Syntax	boolean (true or false)
Default:	TRUE
Min Allowed:	0
Max Allowed:	1
Supported by:	yangcli
Example:	`yangcli --autohistory=false`

3.4 --autoload

The **--autoload** parameter controls automatic loading of YANG modules, as needed.

If true, then the program will attempt to find a needed YANG module.

If 'false', then YANG modules must be loaded manually.

--autoload parameter

Syntax	boolean (true or false)
Default:	TRUE
Min Allowed:	0
Max Allowed:	1
Supported by:	yangcli
Example:	`yangcli --autoload=false`

3.5 --bad-data

The **--bad-data** parameter controls how invalid parameter input is handled by the program.

- **ignore**: Use invalid parameter values. Use with caution.
- **warn**: Issue a warning message, but use the invalid data anyway.
- **check**: Prompt the user interactively, and check if the invalid data should be used, or a new value re-entered.
- **error**: Do not use invalid parameter values.

--bad-data parameter

Syntax	enumeration: **ignore** **warn** **check** **error**
Default:	check
Min Allowed:	0
Max Allowed:	1
Supported by:	yangcli
Example:	`yangcli ----bad-data=warn`

3.6 --batch-mode

The **--batch-mode** parameter specifies that the interactive CLI will not be used.

Instead, if a script is provided with the 'run-script' parameter, or a command provided with the 'run-command' parameter, then it will be executed automatically before the program exits. This mode can be used to invoke NETCONF commands from Unix shell scripts.

--batch-mode parameter

Syntax	empty
Default:	none
Min Allowed:	0
Max Allowed:	1
Supported by:	yangcli
Example:	`yangcli --batch-mode \` `--run-script=~/run-tests`

3.7 --config

The **--config** parameter specifies the name of a Yuma configuration file that contains more parameters to process, in addition to the CLI parameters.

Refer to the 'Configuration Files' section for details on the format of this file.

--config parameter

Syntax	string: complete file specification of the text file to parse for more parameters.
Default:	/etc/yuma/<program-name>.conf

Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd yangcli yangdiff yangdump

Example:	`yangcli testmod \`
	`  --config=~/testconf.conf`

3.8 --datapath

The **--datapath** parameter specifies the directory search path to use while searching for data files. It consists of a colon (':') separated list of path specifications, commonly found in Unix, such as the **$PATH** environment variable.

This parameter overrides the **$YUMA_DATAPATH** environment variable, if it is present.

--datapath parameter

Syntax	string: list of directory specifications
Default:	**$YUMA_DATAPATH** environment variable
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd yangcli yangdiff yangdump

Example:	`yangcli \`
	`  --datapath=”~/work2:~/data”`

3.9 --default-module

The **--default-module** parameter specifies the module to search for identifiers, after the **netconf** and **ncx** modules have been checked. This allows a specific module to match identifier names first, before all modules are searched at once. This can avoid a collision if the same identifier value is used in more than one module.

--default-module parameter

Syntax	string: module name without any path or file extension included.
Default:	none
Min Allowed:	0

Max Allowed:	1
Supported by:	yangcli
Example:	`yangcli --default-module=testmod`

3.10 --deviation

The **--deviation** parameter is a leaf-list of modules that should be loaded automatically when the program starts, as a deviation module. In this mode, only the deviation statements are parsed and then made available later when the module that contains the objects being deviated is parsed.

The deviations must be known to the parser before the target module is parsed.

This parameter is used to identify any modules that have deviation statements for the set of modules being parsed (e.g., **--module** and **--subtree** parameters).

A module can be listed with both the **--module** and **--deviation** parameters, but that is not needed unless the module contains external deviations. If the module only contains deviations for objects in the same module, then the **--deviation** parameter does not need to be used.

The program will attempt to load each module in deviation parsing mode, in the order the parameters are entered.

For the **netconfd** program, If any modules have fatal errors then the program will terminate.

For the **yangdump** and **yangcli** programs, each module will be processed as requested.

--deviation parameter

Syntax	module name or filespec
Default:	none
Min Allowed:	0
Max Allowed:	unlimited
Supported by:	netconfd yangcli yangdump

Example:	`yangcli \`
	`  --module=test1 \`
	`  --deviation=test1_deviations`

3.11 --display-mode

The **--display-mode** parameter controls how data is displayed in the program.

The qualified names (identifiers, idenitityref, XPath) within a text configuration can be generated several ways. This is needed because sibling nodes in different XML namespaces can have the same name.

--display-mode values

enum value	description	example
plain	no prefix, just local-name	sysBootError
prefix	XML prefix and local-name	sys:sysBootError
module	module name and local name	system:sysBootError
xml	XML format	<sysBootError xmlns=”foo”>
xml-nons	XML format without namespace (xmlns) attributes	<sysBootError>

When saving configuration files in non-volatile storage, then it is suggested that only 'module' or 'xml' modes be used, since these are the only deterministic modes.

The set of XML prefixes in use at any one time is not persistent, and cannot be relied upon, unless the namespace declarations are saved (xml mode) or the module names are saved (module mode).

display-mode parameter

Syntax	enumeration: **plain** **prefix** **module** **xml** **--xml-nons**
Default:	plain
Min Allowed:	0
Max Allowed:	1
Supported by:	yangcli
Example:	`yangcli --display-mode=module`

3.12 --feature-disable

The **--feature-disable** parameter directs all programs to disable a specific feature.

This parameter is a formatted string containing a module name, followed by a colon ':', followed by a feature name, e.g.,

```
test:feature1
```

It is an error if a **--feature-enable** and **--feature-disable** parameter specify the same feature.

Parameters for unknown features will be ignored.

--feature-disable parameter

Syntax	formatted string (module:feature

Default:	none
Min Allowed:	0
Max Allowed:	unlimited
Supported by:	yangcli yangdiff yangdump netconfd
Example:	`yangcli --feature-disable=test:feature1`

3.13 --feature-enable

The **--feature-enable** parameter directs all programs to enable a specific feature.

This parameter is a formatted string containing a module name, followed by a colon ':', followed by a feature name, e.g.,

```
test:feature1
```

It is an error if a **--feature-disable** and **--feature-enable** parameter specify the same feature.

Parameters for unknown features will be ignored.

--feature-enable parameter

Syntax	formatted string (module:feature
Default:	none
Min Allowed:	0
Max Allowed:	unlimited
Supported by:	yangcli yangdiff yangdump netconfd
Example:	`yangcli--feature-enable=test:feature1`

3.14 --feature-enable-default

The **--feature-enable-default** parameter controls how **yangdump** will generate C code for YANG features by default.

If 'true', then by default, features will be enabled.

If 'false', then by default, features will be disabled.

If a **--feature-enable** or **--feature-disable** parameter is present for a specific feature, then this parameter will be ignored for that feature.

--feature-enable-default parameter

Syntax	boolean (true or false)
Default:	TRUE
Min Allowed:	0
Max Allowed:	1
Supported by:	yangcli yangdiff yangdump netconfd

Example:	`yangcli \`
	`   --feature-enable-default=false`

3.15 --fixorder

The **--fixorder** parameter controls whether PDU parameters will be automatically sent in NETCONF PDUs in the correct order.

If 'true', the schema-defined, canonical order will be used.

If 'false', the specified order that parameters are entered will be used for the PDU order as well.

--fixorder parameter

Syntax	boolean (true or false)
Default:	TRUE
Min Allowed:	0
Max Allowed:	1
Supported by:	yangcli
Example:	`yangcli --fixorder=false`

3.16 --force-target

The **--force-target** parameter controls whether the candidate or running configuration datastore will be used as the default edit target, when both are supported by the server.

--force-target parameter

Syntax	enum (candidate or running)
Default:	candidate
Min Allowed:	0

Max Allowed:	1
Supported by:	yangcli
Example:	`yangcli --force-target=running`

3.17 --help

The **--help** parameter causes program help text to be printed, and then the program will exit instead of running as normal.

This parameter can be combined with the **--help-mode** parameter to control the verbosity of the help text. Use **--brief** for less, and **--full** for more than the normal verbosity.

This parameter can be combined with the **--version** parameter in all programs. It can also be combined with the **--show-errors** parameter in **yangdump**.

The program configuration parameters will be displayed in alphabetical order, not in schema order.

--help parameter

Syntax	empty
Default:	none
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd yangcli yangdiff yangdump
Example:	`yangcli --help`

3.18 --help-mode

The **--help-mode** parameter is used to control the amount of detail printed when help text is requested in some command. It is always used with another command, and makes no sense by itself. It is ignored unless used with the **--help** parameter.

--help-mode parameter

Syntax	choice of 3 empty leafs **--brief** **--normal** **--full**
Default:	normal
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd

	yangcli yangdiff yangdump
Example:	`yangcli --help --help-mode=full`

- **default choice:** normal
- **choice help-mode**
  - **brief**
    - type: empty
    - This parameter specifies that brief documentation mode should be used.
  - **normal**
    - type: empty
    - This parameter specifies that normal documentation mode should be used.
  - **full**
    - type: empty
    - This parameter specifies that full documentation mode should be used.

3.19 --indent

The **--indent** parameter specifies the number of spaces that will be used to add to the indentation level, each time a child node is printed during program operation.

--indent parameter

Syntax	uint32 (0 .. 9)
Default:	3
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd yangcli yangdiff yangdump
Example:	`yangcli --indent=4`

3.20 --log

The **--log** parameter specifies the file name to be used for logging program messages, instead of STDOUT. It can be used with the optional **--log-append** and **--log-level** parameters to control how the log file is written.

--log parameter

Syntax	string: log file specification
Default:	none
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd yangcli yangdiff yangdump
Example:	`yangcli --log=~/test.log&`

3.21 --log-append

The **--log-append** parameter specifies that the existing log file (if any) should be appended , instead of deleted. It is ignored unless the **--log** parameter is present.

--log-append parameter

Syntax	empty
Default:	none
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd yangcli yangdiff yangdump

Example:	`yangcli --log-append \`
	`  --log=~/test.log&`

3.22 --log-level

The **--log-level** parameter controls the verbosity level of messages printed to the log file or STDOUT, if no log file is specified.

The log levels are incremental, meaning that each higher level includes everything from the previous level, plus additional messages.

There are 7 settings that can be used:

- **none**: All logging is suppressed. Use with extreme caution!
- **error**: Error messages are printed, indicating problems that require attention.
- **warn**: Warning messages are printed, indicating problems that may require attention.
- **info**: Informational messages are printed, that indicate program status changes.
- **debug**: Debugging messages are printed that indicate program activity.

- **debug2**: Protocol debugging and trace messages are enabled.
- **debug3**: Very verbose debugging messages are enabled. This has an impact on resources and performance, and should be used with caution.
- **debug4**: Maximum debugging messages are enabled.

log-level parameter

Syntax	enumeration: **off** **error** **warn** **info** **debug** **debug2** **debug3** **debug4**
Default:	--info (--debug for DEBUG builds)
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd yangcli yangdiff yangdump

Example:	`yangcli --log-level=debug \`
	`  --log=~/test.log&`

3.23 --match-names

The **--match-names** parameter specifies how names are matched when performing UrlPath searches.

The following values are supported::

- **exact**

  The name must exactly match the node name for all characters in both name strings.

- **exact-nocase**

  The name must match the node name for all characters in both name strings.

  Strings are not case-sensitive.

- **one**

  The name must exactly match the first N characters of just one node name, which must be the only partial name match found.

- **one-nocase**

  The name must exactly match the first N characters of just one node name, which must be the only partial name match found. Strings are not case-sensitive.

- **first**

  The name must exactly match the first N characters of any node name. The first one found will be used.

- **first-nocase**

  The name must exactly match the first N characters of any node name. The first one found will be used. Strings are not case-sensitive.

--match-names parameter

Syntax	enum: exact exact-nocase one one-nocase first first-nocase
Default:	one-nocase
Min Allowed:	0
Max Allowed:	1
Supported by:	yangcli
Example:	`yangcli --match-names=exact`

3.24 --modpath

The **--modpath** parameter specifies the YANG module search path to use while searching for YANG files. It consists of a colon (':') separated list of path specifications, commonly found in Unix, such as the **$PATH** environment variable.

This parameter overrides the **$YUMA_MODPATH** environment variable, if it is present.

--modpath parameter

Syntax	string: list of directory specifications
Default:	**$YUMA_MODPATH** environment variable
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd yangcli yangdiff yangdump

Example:	`yangcli \`
	`  --modpath=”~/testmodules:~/modules:`
	`~/trunk/netconf/modules”`

3.25 --module

The **--module** parameter is a leaf-list of modules that should be loaded automatically when the program starts.

The program will attempt to load each module in the order the parameters were entered.

For the **netconfd** program, If any modules have fatal errors then the program will terminate.

For the **yangdump** program, each module will be processed as requested.

--module parameter

Syntax	module name or filespec
Default:	none
Min Allowed:	0
Max Allowed:	unlimited
Supported by:	netconfd yangcli yangdump

Example:	`yangcli \`
	`  --module=test1 \`
	`  --module=test2`

3.26 --ncport

The **--ncport** parameter specifies the TCP port number that should be used for NETCONF sessions.

This parameter specifies the TCP port number to use when a NETCONF session is created during the startup of the program. The parameter can only be entered once.

--ncport parameter

Syntax	uint32
Default:	830
Min Allowed:	0
Max Allowed:	1
Supported by:	yangcli

Example:	`yangcli --server=myserver`
	`  --ncport=22 \`
	`  --user=fred`
	`  --password=mypassword`

3.27 --password

The **--password** parameter specifies the password string that should be used when a NETCONF session is connected during the startup of the program.

--password parameter

Syntax	string
Default:	none
Min Allowed:	0
Max Allowed:	1
Supported by:	yangcli

Example:	`yangcli --server=myserver \`
	`  --password=yangrocks \`
	`  --user=andy`

3.28 --private-key

The **--private-key** parameter specifies the file path specification for the file containing the client-side private key.

If both 'public-key' and 'private-key' files are present, the client will attempt to connect to the server using these keys. If this fails, or not done, then password authentication will be attempted.

--private-key parameter

Syntax	string
Default:	$HOME/.ssh/id_rsa
Min Allowed:	0
Max Allowed:	1
Supported by:	yangcli
Example:	`yangcli –private-key=~/.ssh/mykey`

3.29 --protocols

The **--protocols** parameter specifies which protocol versions the program or session will attempt to use. Empty set is not allowed.

--protocols parameter

Syntax	bits
Default:	“netconf1.0 netconf1.1”
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd yangcli
Example:	`yangcli –protocols=netconf1.0`

3.30 --public-key

The **--public-key** parameter specifies the file path specification for the file containing the client-side public key.

If both 'public-key' and 'private-key' files are present, the client will attempt to connect to the server using these keys. If this fails, or not done, then password authentication will be attempted.

--public-key parameter

Syntax	string
Default:	$HOME/.ssh/id_rsa.pub
Min Allowed:	0
Max Allowed:	1
Supported by:	yangcli
Example:	`yangcli –public-key=~/.ssh/mykey.pub`

3.31 --runpath

The **--runpath** parameter specifies the directory search path to use while searching for script files. It consists of a colon (':') separated list of path specifications, commonly found in Unix, such as the **$PATH** environment variable.

This parameter overrides the **$YUMA_RUNPATH** environment variable, if it is present.

--runpath parameter

Syntax	string: list of directory specifications
Default:	**$YUMA_RUNPATH** environment variable
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd yangcli yangdiff yangdump

Example:	`yangcli \`
	`  --runpath=”~/testscripts”`

3.32 --run-command

The **--run-command** parameter specifies a command will be invoked upon startup.

In the **yangcli** program, if the auto-connect parameters are provided, then a session will be established before running the command.

--run-command parameter

Syntax	string
Default:	none
Min Allowed:	0
Max Allowed:	1
Supported by:	yangcli

Example:	`yangcli \`
	`  --run-command="history load=~/test3-history"`

3.33 --run-script

The **--run-script** parameter specifies a script will be invoked upon startup.

In the **yangcli** program, if the auto-connect parameters are provided, then a session will be established before running the script.

--run-script parameter

Syntax	string (file specification)
Default:	none
Min Allowed:	0
Max Allowed:	1
Supported by:	yangcli

Example:	`yangcli \`
	`  --run-script="test3-start"`

3.34 --server

The **--server** parameter specifies the IP address or DNS name of the NETCONF server that should be connected to automatically, when the program starts.

--server parameter

Syntax	string:IP address or DNS name
Default:	none
Min Allowed:	0

Max Allowed:	1
Supported by:	yangcli
Example:	`yangcli --server=myserver`

3.35 --subdirs

The **--subdirs** parameter controls whether sub-directories should be searched or not, if they are found during a module search.

If false, the file search paths for modules, scripts, and data files will not include sub-directories if they exist in the specified path.

--subdirs parameter

Syntax	boolean
Default:	TRUE
Min Allowed:	0
Max Allowed:	1
Supported by:	yangdump
Example:	`yangcli  --subdirs=false`

3.36 --time-rpcs

The **--time-rpcs** parameter is used to measure the round-trip time of each <rpc> request and <rpc-reply> at the session level. Echo the elapsed time value to screen if in interactive mode, as well as the log if the log is a file instead of stdout. The $$time-rpcs system variable is derived from this parameter.";

--time-rpcs parameter

Syntax	boolean
Default:	FALSE
Min Allowed:	0
Max Allowed:	1
Supported by:	yangcli
Example:	`yangcli --time-rpcs=true`

3.37 --timeout

The **--timeout** parameter specifies the number of seconds that should pass before giving up on a response during a NETCONF session. The value zero means wait forever (no timeout).

I

--timeout parameter

Syntax	uint32 (0 for no timeout; otherwise number of seconds to wait)
Default:	30 seconds
Min Allowed:	0
Max Allowed:	1
Supported by:	yangcli
Example:	`yangcli --timeout=0`

3.38 --use-xmlheader

The **--use-xmlheader** parameter specifies how file result variables will be written for XML files.

Controls whether the XML preamble header will be written or not.

--use-xmlheader parameter

Syntax	boolean
Default:	TRUE
Min Allowed:	0
Max Allowed:	1
Supported by:	yangcli
Example:	`yangcli --use-xmlheader=false`

3.39 --user

The **--user** parameter specifies the user name string on the NETCONF server that should be used when establishing a NETCONF session.

--user parameter

Syntax	string:user name
Default:	none
Min Allowed:	0
Max Allowed:	1
Supported by:	yangcli

Example:	`yangcli --user=admin \`
	`  --server=myserver`

3.40 --version

The **--version** parameter causes the program version string to be printed, and then the program will exit instead of running as normal.

All Yuma version strings use the same format:

DEBUG: <major>.<minor>.<svn-build-version>

or

NON-DEBUG: <major>.<minor>-<release>

An example version number that may be printed:

```
        yangcli 1.15-4
```

This parameter can be combined with the **--help** parameter.

--version parameter

Syntax	empty
Default:	none
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd yangcli yangdiff yangdump
Example:	`yangcli --version`

3.41 --warn-idlen

The **--warn-idlen** parameter controls whether identifier length warnings will be generated.

The value zero disables all identifier length checking. If non-zero, then a warning will be generated if an identifier is defined which has a length is greater than this amount.

--warn-idlen parameter

Syntax	uint32: 0 to disable, or 8 .. 1023
Default:	64
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd yangcli yangdiff

	yangdump
Example:	`yangcli --warn-idlen=50`

3.42 --warn-linelen

The **--warn-linelen** parameter controls whether line length warnings will be generated.

The value zero disables all line length checking. If non-zero, then a warning will be generated if a YANG file line is entered which has a length is greater than this amount.

Tab characters are counted as 8 spaces.

--warn-linelen parameter

Syntax	uint32: 0 to disable, or 40 .. 4095
Default:	72
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd yangcli yangdiff yangdump
Example:	`yangcli --warn-linelen=79`

3.43 --warn-off

The **--warn-off** parameter suppresses a specific warning number.

The error and warning numbers, and the default messages, can be viewed with the yangdump program by using the **--show-errors** configuration parameter.

The specific warning message will be disabled for all modules. No message will be printed and the warning will not count towards the total for that module.

--warn-off parameter

Syntax	uint32: 400 .. 899
Default:	none
Min Allowed:	0
Max Allowed:	499
Supported by:	netconfd yangcli yangdiff yangdump
Example:	`yangcli --warn-off=435`

	`# revision order not descending`

3.44 --yuma-home

The **--yuma-home** parameter specifies the project directory root to use when searching for files.

If present, this directory location will override the '**$YUMA_HOME** environment variable, if it is set. If this parameter is set to a zero-length string, then the $**YUMA_HOME** environment variable will be ignored.

The following directories are searched when either the **$YUMA_HOME** environment variable or this parameter is set:

- **$YUMA_HOME/modules**
  - This sub-tree is searched for YANG files.
- **$YUMA_HOME/data**
  - This directory is searched for data files.
- **$YUMA_HOME/scripts**
  - This directory is searched for **yangcli** script files.

yuma-home parameter

Syntax	string: directory specification
Default:	**$YUMA_HOME** environment variable
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd yangcli yangdiff yangdump

Example:	`yangcli \`
	`  --yuma-home=~/sw/netconf \`
	`  --log=~/test.log&`