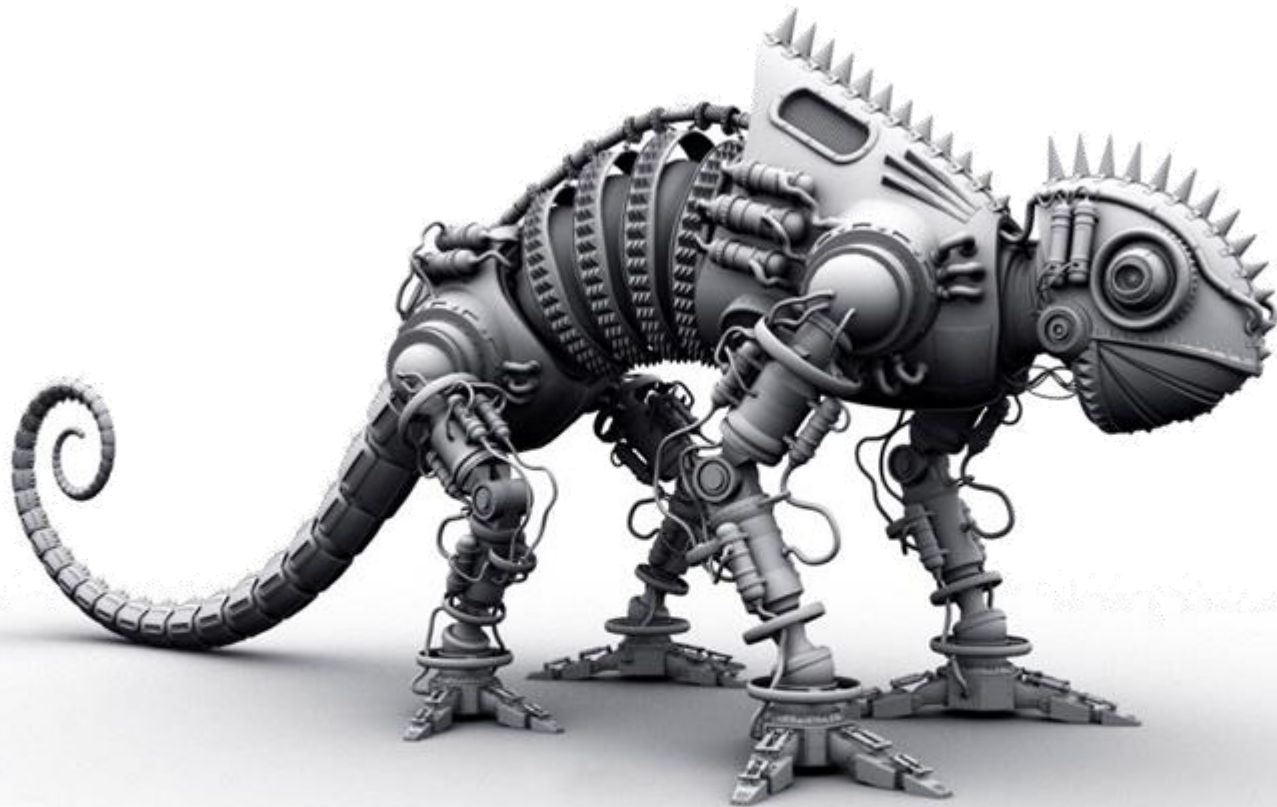




# Xena Automation

Automation – Step by Step





## Intro

## CLI API

Basics

Commands and Status messages

First steps

## Scripting...

Port configuration

Test Logic

Login as example

## Automating Test Suites



## What is Xena Scripting?

Command Line Interface (CLI) to the Xena Server (chassis)

- Covers all functionality.

Supports interactive sessions

- Command and Reply
- Replay of command lists

Scripted or programmed sessions

- Pre-programmed actions and check

Reuse of existing GUI setup

- Jump start script writing
- Use as template

# Xena Scripting Advantages

## Versatility

- Can be used with ANY scripting and program language
- Can run in ANY operating system
- No need to install drivers or proprietary programs
- Re-use your existing automation framework

## Fast

- Extremely low communication overhead
- Very simple and efficient protocol
- No requirements to the CLI client

## Simplicity

- Purely text based
- No binary modules

## Fast Learning curve

- Very easy to learn
- Fully documented
- Very easy to debug
- Reuse existing configurations





## Xena Automation is based on:

1. A TCP connection from script/program to Xena tester
- +
2. Input/output via CLI-like API commands over the TCP socket



### TCL Example

Creating a TCP Socket –

```
set s [socket -async $chassis_ip $chassis_port]
```

Communicating via Socket (s) –

```
puts $s "c_logon $chassis_pass"  
gets $s response
```



## Xena Automation is based on:

1. A TCP connection from script/program to Xena tester
- +
2. Input/output via CLI-like API commands over the TCP socket



### Ruby Example:

Creating a TCP Socket –

```
require 'socket'    # Sockets are in standard library
s = TCPSocket.open($hostname, $port)
```

Communicating via Socket (s) –

```
socket.puts(tx_string)
response = socket.gets
```



## Xena Automation is based on:

1. A TCP connection from script/program to Xena tester
- +  
2. Input/output via CLI-like API commands over the TCP socket



### Python Example:

Creating a TCP Socket –

```
from LabUtils.Drivers.SocketDrivers import SimpleSocket  
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

Communicating via Socket (s) –

```
sock.send(cmd + '\n')  
response = sock.recv(1024)
```



## Xena Automation is based on:

1. A TCP connection from script/program to Xena tester
- +
2. Input/output via CLI-like API commands over the TCP socket



### Bash Example:

Creating a TCP Socket –

```
exec 3<> /dev/tcp/${MACHINE}/${PORT}
```

Communicating via Socket (s) –

```
echo -en "C_LOGON ${PASSWORD}\r\n" >&3  
read <&3
```





# WORTH REMEMBERING

TCP Port used for connecting:

22611

Once TCP Socket is open, ASCII text commands are used, must be terminated by:

CR/LF



## Command Structure

`module/port command [index] value value ...`

Example:

`2/5 PS_RATEPPS [3] 500000`

Meaning:

- Module is no. 2
  - Port on module is no. 5
  - Command is: `PS_RATEPPS` =Set Stream Rate as Packets Per Second
  - Stream on port is no. 3 (*SID*)
  - Packets Per Second is set to 500000
- 
- All indices start at zero
- \* Use “?” as value to READ current setting (explained in next slide)



## Command Structure

Most commands can be used both to :  
query the current status/data using “?” or  
modify/set data via the actual value you would like to set:

You would query for the current value this way:

```
0/5 PS_RATEPPS [3] ?
```

And the chassis would respond the same way that you set the value yourself:

```
0/5 PS_RATEPPS [3] 500000
```



## Command Identifiers

- C\_.....CHASSIS PARAMETER
- M\_..... MODULE PARAMETER
- P\_..... PORT PARAMETER
- PS\_.....STREAM PARAMETER
- PM\_..... MATCH TERM PARAMETER
- PL\_..... LENGTH TERM PARAMETER
- PF\_.....FILTER PARMATERS
- PC\_..... CAPTURE PARAMETER
- PT\_..... TRANSMIT STATISTICS PARAMETER
- PR\_..... RECEIVE STATISTICS PARAMETER
- PD\_..... DATASET PARAMETER(HISTOGRAMS)
- PP\_..... 40/100G PARAMETER



## Special scripting commands

Commands for supporting the scripting process itself:

- sync** Produces a reply of **<SYNC>**, helpful when parsing and delimiting returned lines
- sync on** Automatically set “**sync**” after each command.
- sync off** Disables “**sync on**”.
- wait n** Waits for the specified number of seconds, (max 60), then replies **<RESUME>**
- help ?** Gives an overview of the built-in help function
- help "cmd"** Gives a brief overview of the syntax for “cmd”



# Status messages

The set/change commands themselves simply produce a reply from the chassis of:

<OK>

If something is unacceptable to the chassis, it may return one of the following:

- <NOTLOGGEDON> You have not issued a C\_LOGON providing the chassis password.
- <NOTRESERVED> You have not issued a x\_RESERVATION for the resource you want to change.
- <NOTWRITABLE> The parameter is read-only.
- <NOTREADABLE> The parameter is write-only.
- <NOTVALID> The operation is not valid in the current chassis state, e.g. because traffic is on.
- <BADMODULE> The module index value is out of bounds.
- <BADPORT> The port index value is out of bounds.
- <BADINDEX> A parameter sub-index value is wrong.
- <BADSIZE> The size of a data value is not appropriate.
- <BADVALUE> A value is not appropriate.
- <FAILED> An operation failed to produce a result.



## Status messages

If there is a plain syntax error, misspelled parameter, or an inappropriate use of module/port/indices, the chassis will return a line pointing out the column where the error was detected, e.g.:

```
0/5 PS_RATEPPS [] 5q00
                        ----^
```

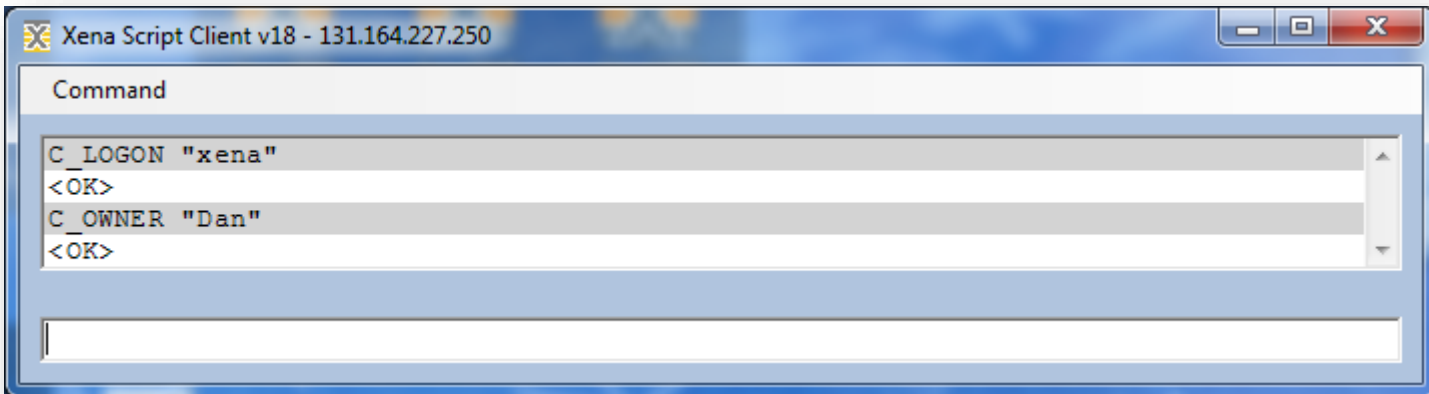
**#Syntax error in column 24**



First open connection via script/program or via Xena Script Client.

Then we authenticate the connection to the chassis and provide a user name for reservation:

```
C_LOGON "xena"  
C_OWNER "example"
```





# CLI API - First Steps



The screenshot shows the Wireshark 1.10.9 interface. The main window displays a packet capture with a filter set to 'tcp.port == 22611'. The packet list shows several TCP packets between 192.168.1.80 and 192.168.1.170. Packet 152 is selected, and the 'Follow TCP Stream' dialog box is open, showing the stream content:

```
Stream Content
c_logon "xena"
<OK>
c_owner "Michael"
<OK>
```

The dialog box also shows the entire conversation (48 bytes) and options for displaying the stream (ASCII, EBCDIC, Hex Dump, C Arrays, Raw). The status bar at the bottom indicates the file path: 'C:\Users\MICHAEL\AppData\Local\T...' and the profile is 'Default'.



An automated test script has two main components:

- 1 - Port Configuration
- 2 - Test Logic

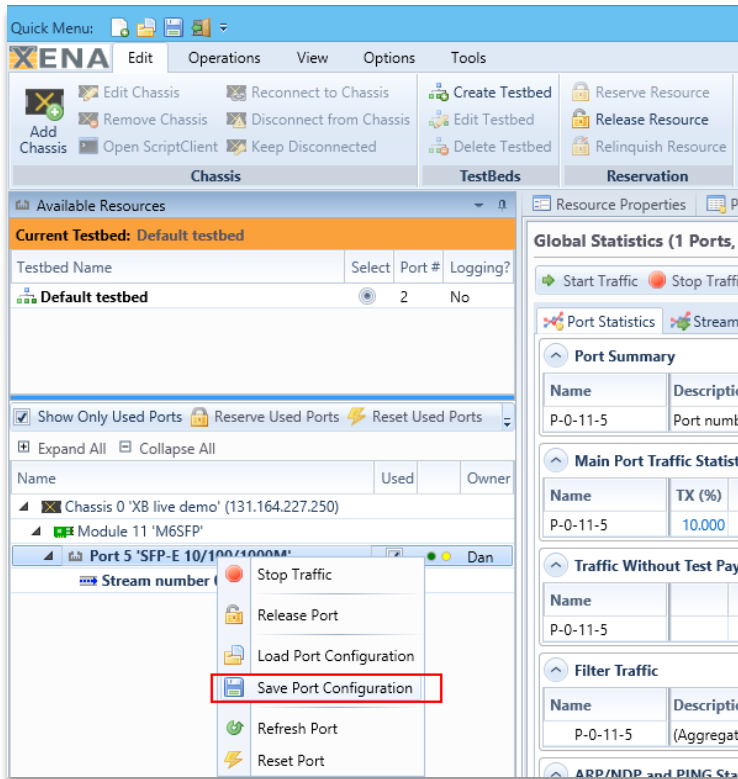


## Port Configuration

Includes *Port, Streams, Capture, Filters, Histograms* Configuration commands

Can be exported via the GUI

Can be imported to a port via a small method



```

File Edit Search View Encoding Language Sett
1xpc
6 ;Global: S+C+T+R+
7 P_RESET
8 P_SPEEDSELECTION F10G
9 P_COMMENT "Port number 1"
10 P_SPEEDREDUCTION -1
11 P_INTERFRAMEGAP 20
12 P_MACADDRESS 0x04F4BC0C2911

```



```

proc LoadPortConfig { s port file_name console} {
  set file_data [read $fp]
  set data [split $file_data "\n"]

  foreach line $data {
    if {$line==""} {break}
    if ![string match ".*" $line] {
      puts $s "$port $line"
      gets $s response
    }
  }
  close $fp
}

```

# Scripting... – Port Configuration (reverse)



The screenshot shows the XenaManager-2G v1.12 interface with several windows and menus. The main window displays a tree view of testbeds. A red box labeled '1' highlights the selected testbed: 'Chassis 0 'XB live demo' (131.164.227.250)'. The 'Chassis' menu is open, and a red box labeled '2' highlights the 'Open ScriptClient' option. A secondary window titled 'Chassis 0 'XB live demo' (131.164.227.250)' is open, with a red box labeled '3' highlighting the 'Load Commands' button. A third window titled 'Load Commands From ...' is open, showing a file explorer view of the Desktop. A red box labeled '4' highlights the 'Open' button in the file dialog.



## Test Logic

If/else/while/etc... logic provided by the scripting or program language

Real-time interaction commands with Xena referred to as the **SCRIPT CORE COMMANDS**

Real-time interaction commands with the DUT

### Connect

Connect / Login

### Reservation

Reserve / Release / +IsReserved? / + IsReservedByMe?

### Port

PortLinkUp / PortLinkDown / +HasLink?

### Configuration

Clear / Load

### Traffic

Start / Stop

### Capture

Start / Stop / Save

### Results

Clear / Get (Port / Stream / Filter) / Save (Port / Stream / *Filter*) to .CSV



```
# ----- Login -----
proc Login {s chassis_pass chassis_user console} {

    set pf_flag 1

    puts $s "c_logon $chassis_pass"
    gets $s response
    if {$response == ""} { gets $s response }

    if {"$response" != "<OK>"} {set pf_flag 0}
    if {$console == 1} { puts "Logging | $response" }

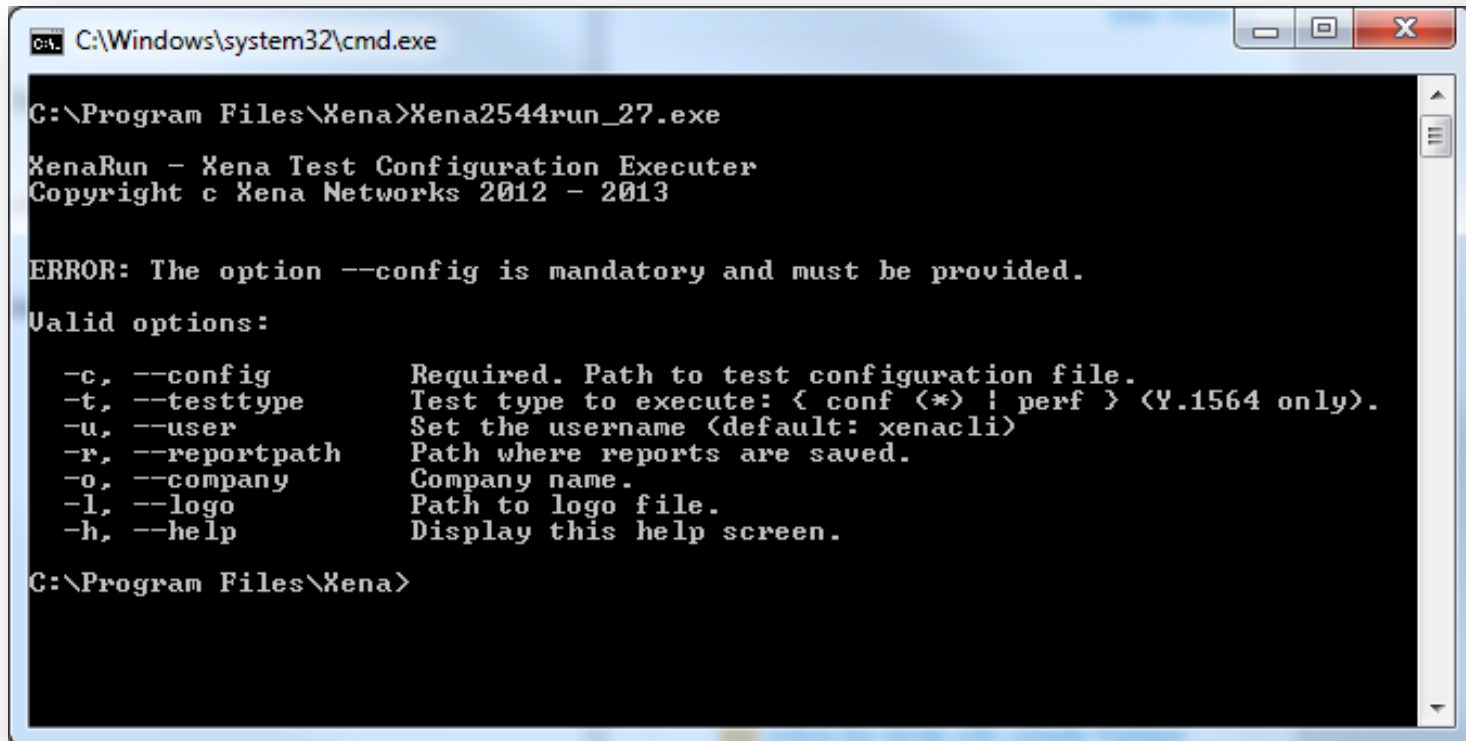
    puts $s "c_owner $chassis_user"
    gets $s response
    if {$response == ""} { gets $s response }

    if {$console == 1} { puts "Owner | $response" }
    if {"$response" != "<OK>"} {set pf_flag 0}

    return $pf_flag
}
#
# -----
```

Use simple one line shell command to automate a full RFC test based on prebuilt test configuration

TCL uses "EXEC" in order to execute a process



```
C:\Windows\system32\cmd.exe

C:\Program Files\Xena>Xena2544run_27.exe

XenaRun - Xena Test Configuration Executer
Copyright c Xena Networks 2012 - 2013

ERROR: The option --config is mandatory and must be provided.

Valid options:

-c, --config          Required. Path to test configuration file.
-t, --testtype       Test type to execute: { conf (*) ! perf } (Y.1564 only).
-u, --user           Set the username (default: xenacli)
-r, --reportpath     Path where reports are saved.
-o, --company        Company name.
-l, --logo           Path to logo file.
-h, --help           Display this help screen.

C:\Program Files\Xena>
```



## RESOURCES

Wiki: [wiki.xenanetworks.com](http://wiki.xenanetworks.com)

Website: [www.xenanetworks.com/resources/](http://www.xenanetworks.com/resources/)

Email: [support@xenanetworks.com](mailto:support@xenanetworks.com)



**XenaWiki**  
Detailed user manuals and  
technical documentation