

# Automating Xena Test Suites

**FREE**



**Xena2544**

THE EASY WAY TO TEST RFC2544  
ON XENA TEST EQUIPMENT

Free Layer 2-3 Test Application

**FREE**



**Xena2889**

FOR ACCURATELY TESTING  
LAYER 2 SWITCH PERFORMANCE

Free Layer 2-3 Test Application

**FREE**



**Xena3918**

ADVANCED IP MULTICAST  
NETWORK TESTING

Free Layer 2-3 Test Application

**FREE**



**Xena1564**

THE SMART WAY TO TEST Y.1564  
ON XENA TEST EQUIPMENT

Free Layer 2-3 Test Application



Rev. 1



# Preface

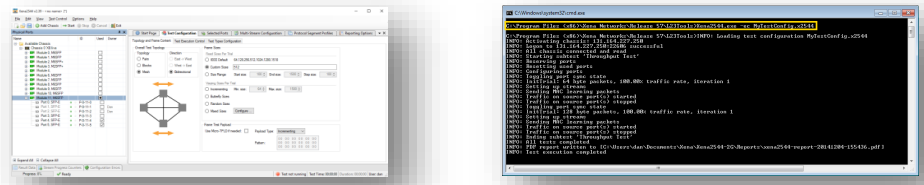
Xena Networks provides several Benchmarking Test Suites.



Each Test Suite has an application Executable.

- Xena1564.exe
- Xena2544.exe
- Xena2889.exe
- Xena3918.exe

Test Suites can be executed from GUI and also Automated.





# Preface

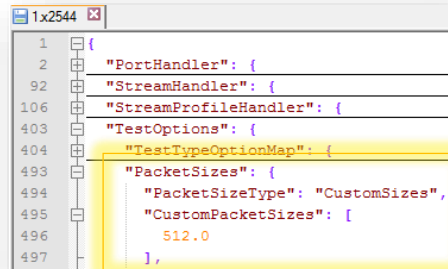
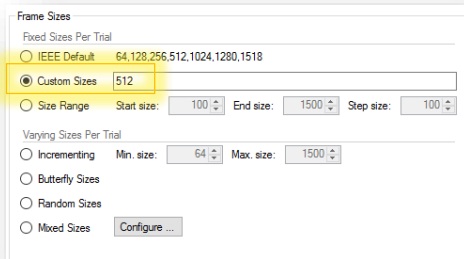
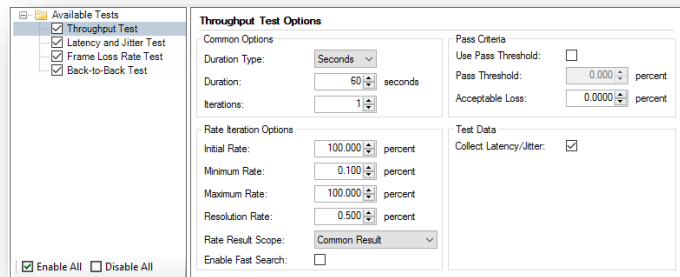
The Application has **embedded test logic** based on:

A standards definitions rules (e.g. *RFC2544* - Throughput).

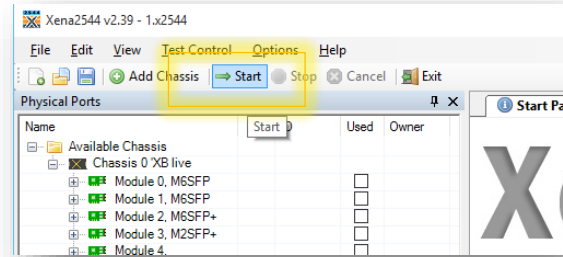
Operations such as Start/Stop/Set Traffic are determined by presets and also changed on the fly based on live results and conditions.

A set of configurable parameters.

Parameters are stored in a JSON formatted file:



A Test can be initiated from within the Test Suite GUI:



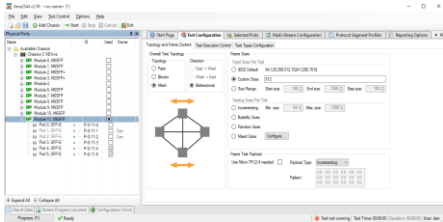
Can also be initiated from a command line (     +  )

Executed either manually or by a script/program.

Supported both on Windows and Linux shell (   ).



# Preface



**GUI**

- Xena1564.exe
- Xena2544.exe
- Xena2889.exe
- Xena3918.exe

**Command line**



Microsoft .NET Based

```

C:\Windows\system32\cmd.exe
C:\Program Files (x86)\Xena Networks\Release 57\L23Tools>Xena2544.exe -ec MyTestConfig.x2544
C:\Program Files (x86)\Xena Networks\Release 57\L23Tools>INFO: Loading test configuration MyTestConfig.x2544
INFO: Activating chassis: 131.164.227.250
INFO: Logon to 131.164.227.250:22606 successful
INFO: All chassis connected and read
INFO: Starting subtest 'Throughput Test'
INFO: Reserving ports
INFO: Resetting used ports
INFO: Configuring ports
INFO: Toggling port sync state
INFO: InitTrial: 64 byte packets, 100.00% traffic rate, iteration 1
INFO: Setting up streams
INFO: Sending MAC learning packets
INFO: Traffic on source port(s) started
INFO: Traffic on source port(s) stopped
INFO: Toggling port sync state
INFO: InitTrial: 128 byte packets, 100.00% traffic rate, iteration 1
INFO: Setting up streams
INFO: Sending MAC learning packets
INFO: Traffic on source port(s) started
INFO: Traffic on source port(s) stopped
INFO: Ending subtest 'Throughput Test'
INFO: All tests completed
INFO: PDF report written to [C:\Users\dan\Documents\Xena\Xena2544-26\Reports\xena2544-report-20141204-155436.pdf]
INFO: Test execution completed
  
```

```

monotest01.x2544.exe
monotest01.x2544.exe
[mikkel@localhost xenabin]$ mono ./Xena2544.exe -c monotest01.x2544 -e
INFO: Loading test configuration monotest01.x2544
INFO: Activating chassis: 192.168.1.170
INFO: Logon to 192.168.1.170:22606 successful
INFO: All chassis connected and read
INFO: Starting subtest 'Throughput Test'
INFO: Reserving ports
INFO: Resetting used ports
INFO: Configuring ports
INFO: Toggling port sync state
INFO: InitTrial: 512 byte packets, iteration 1
INFO: Setting up streams
INFO: Sending MAC learning packets
INFO: Rate iteration: 100.000% traffic rate
INFO: Traffic on source port(s) started
INFO: Traffic on source port(s) stopped
INFO: Ending subtest 'Throughput Test'
INFO: All tests completed
INFO: PDF report written to [/home/mikkel/Xena/Xena2544-26/Reports/xena2544-report-20150904-141323.pdf]
INFO: Test execution completed
[mikkel@localhost xenabin]$
  
```

THE GLOBAL PRICE/PERFORMANCE LEADER IN GIGABIT ETHERNET TEST & MEASUREMENT



## Pre-Test – Configuration Stage:

# Import JSON Library

```
import json
```

# Read configuration file to variable

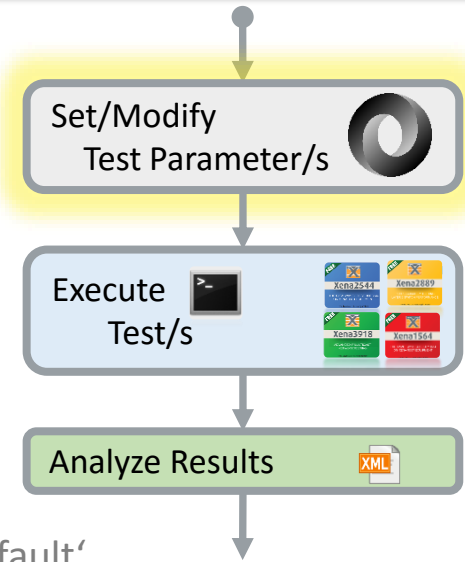
```
with open('Configuration.x2544', 'r', encoding='utf-8') as data_file:  
    x2544_Configuration = json.loads(data_file.read())
```

# Modify Value of variable (e.g. TestOptions->PacketSizes->PacketSizeType)

```
x2544_Configuration['TestOptions']['PacketSizes']['PacketSizeType'] = 'IEEEDefault'
```

# Write modified to the file that is expected to be used.

```
with open('2bUsed.x2544', 'w', encoding='utf-8') as f:  
    json.dump(x2544_Configuration, f, indent = 2, sort_keys = True, ensure_ascii=True)
```





## Live Test – Execute(/Monitor) Stage:

```
# Import subprocess Library
```

```
import subprocess
```

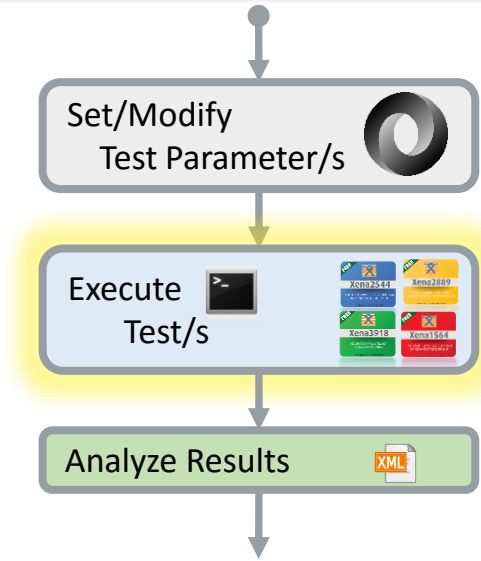
```
# Call the exe using "-ec" and the "2bUsed" Configuration file
```

```
subprocess.call(["C:\Program Files (x86)\Xena Networks\Release  
62.3\L23Tools\Xena2544.exe", "-ec", "2bUsed.x2544"])
```

```
# Linux example
```

```
args = ["mono", "./tools/pkt_gen/xena/Xena2544.exe", "-ec",  
        "./tools/pkt_gen/xena/profiles/2bUsed.x2544"]
```

```
subprocess.call([args, stdout=sys.stdout])
```





# Test Stages



## Post Test – Analysis Stage:

# Import ElementTree Library for easy XML parsing

```
import xml.etree.ElementTree as ET
```

# Parse the XML report and get root handle

```
root = ET.parse(r'xena2544-report.xml').getroot()
```

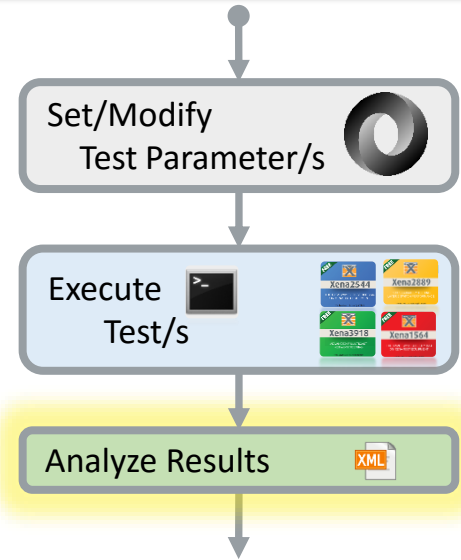
# Address specific results counters by name/location

```
TotalTxFrames = root[0][1][0].get('TotalTxFrames')
```

```

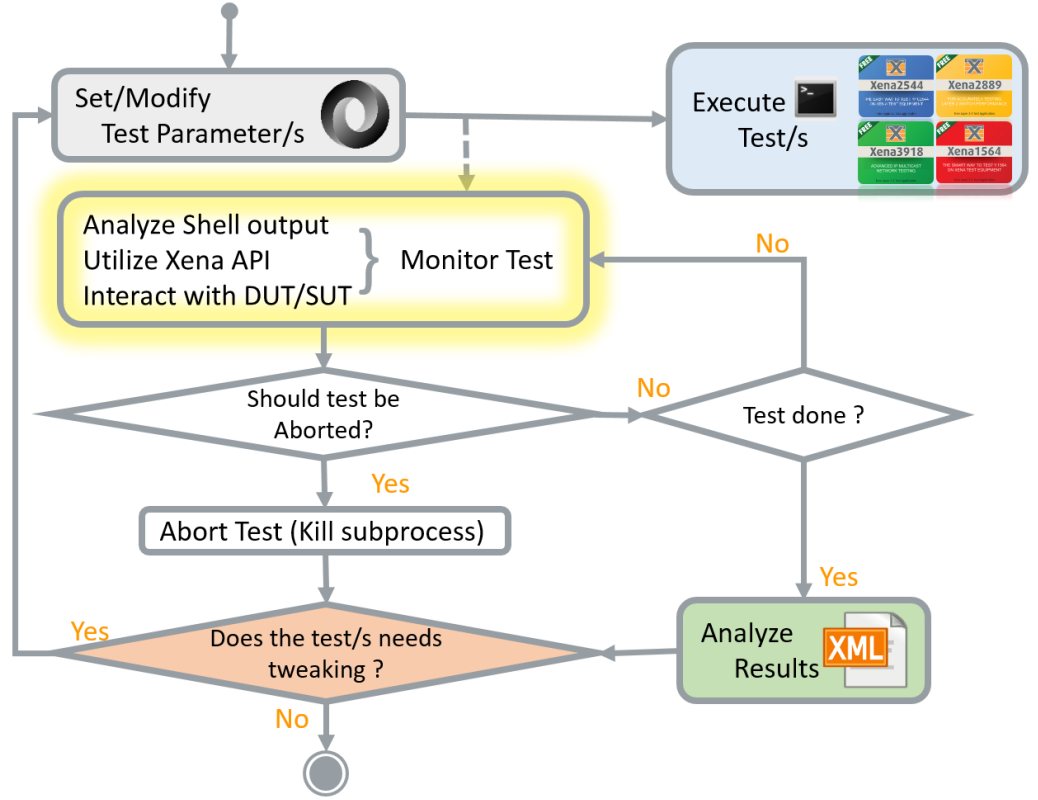
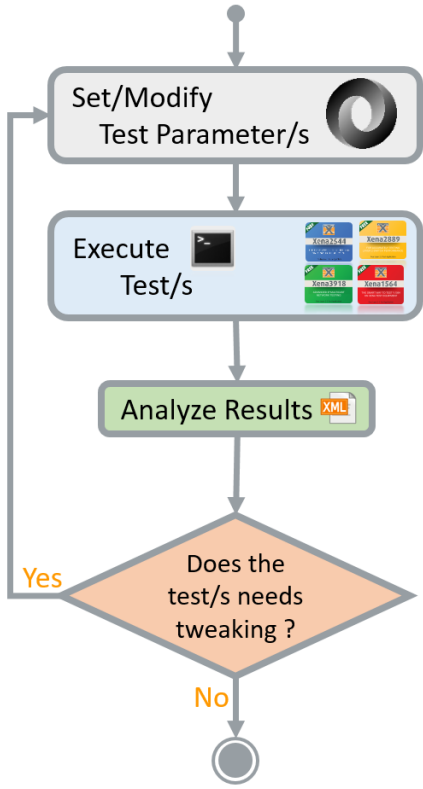
1 <?xml version="1.0" encoding="utf-8"?>
2 <!--Xena2544 Test Report-->
3 <Xena2544>
4 0 <TestResults>
5   <Summary>
6     <Identification TestCompany="Xena Networks ApS" Customer="Xena Networks" CustomerAccessID="" CustomerServiceID="" />
7     <Metrics TestDateTime="20151006-122916" TestDuration="21" GeneratedBy="Xena2544 v2.28" />
8     <Comment><![CDATA[]]></Comment>
9   </Summary>
10 1 <TestResult TestType="Throughput">
11 0 <Result FrameSize="100" TestState="PASS" TotalTxRatePcnt="1" TotalTxFrames="20832" TotalTxRateBpsL1="19998720" TotalTxRa
12 <Port ID="P-0-11-0" PortTxFrames="10416" PortTxBpsL1="9999360" PortTxBpsL2="8332800" PortTxPps="10416" PortRxFrames="1"
13 <Port ID="P-0-11-1" PortTxFrames="10416" PortTxBpsL1="9999360" PortTxBpsL2="8332800" PortTxPps="10416" PortRxFrames="1"
14 </Result>
15 </TestResult>
16 </TestResults>
17 <TestConfiguration>
18 </Xena2544>

```





# Blocking Vs Non Blocking



Analyze Shell output

Utilize Xena API

Interact with DUT/SUT



Monitor Test

In order to execute a Test Suite and monitor test one must first spawn a subprocess

Use **Popen** to create a subprocess:

```
ps = subprocess.Popen(args, stdout=sys.stdout)
```

Use **Communicate** to retrieve Shell output:

```
ps.communicate()
```

Use **Terminate** or **Kill** to Abort or Stop test:

```
ps.kill()
```

Analyze Shell output  
Utilize Xena API  
Interact with DUT/SUT } Monitor Test

In order to also monitor the Xena test ports counters one must Utilize Xena RAW API.

Opening a TCP socket to the chassis IP (port 22611) and retrieving counters via the API commands.

In order to monitor test ports no Reservation is required.

Overriding Test Ports reservation will abort the test.

More information on how to use Xena RAW API @ **Xena Automation Introduction** pdf.