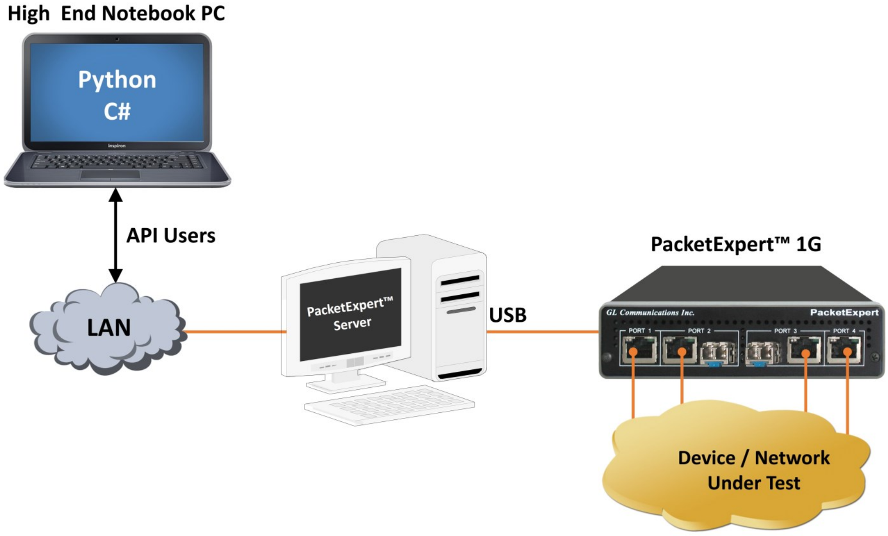
Automated GigE Network Testing using PacketExpert™ 1G - MAPS™ CLI / API Architecture



Overview

PacketExpert™ 1G supports Command line Interface (CLI) for test automation and remote accessibility of various functionalities such as BERT, Loopback, RFC 2544, Record Playback, IPNetSim™, IPLinkSim™, ExpertSAM™, PacketBroker™, and Multi Stream Traffic Generator and Analyzer using Python, C# client APIs and MAPS™ CLI Client/Server architecture.

Required license for CLI Support in PacketExpert™ 1G is -

- CXE100 (CLI Server for PXE100 basic and optional software)

PacketExpert™ can be configured as server-side application using the GL's MAPS™ Client-Server architecture, to provide the capability of remote operation, automation, and multi-site connectivity, using any client-side scripting tools such as the Python and C#. On the client side, the scripting library enables communication with the MAPS™ CLI Server using TCP/IP socket from a client environment.

The MAPS™ CLI server interfaces with the PacketExpert™ hardware through the USB. The MAPS™ CLI Server developed specifically for PacketExpert™ runs (*.gls) scripts that can control the PacketExpert™ hardware. The advantage of such communication enables user to control PacketExpert™ by sending commands and receiving responses in a scripting language such as Python, C# that is already familiar with many users.

For more information, visit PacketExpert™ CLI testing webpage.

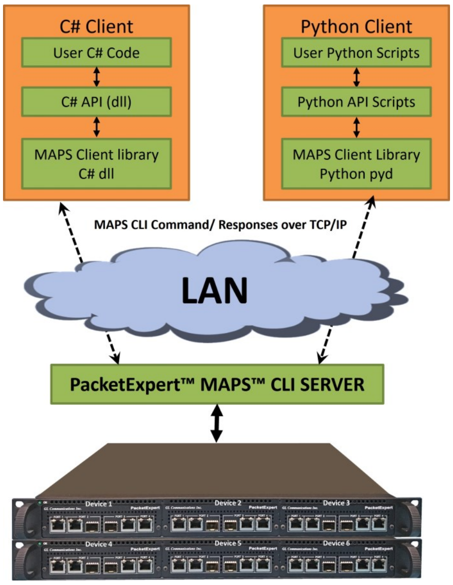
Main Features

- Capability of remote operation, automation and multi-site connectivity using Python/C# client and MAPS™ CLI server.
- Supports Bert, Loopback, RFC 2544, Record Playback, ExpertSAM™, PacketBroker™, and Multi-Stream Traffic Generator and Analyzer functionalities.
- PacketExpert™ CLI offers complete Lab Management, Device Provisioning and Test Automation solutions.
- CLI integration with popular framework such as LabVIEW/TestStand and TestShell for test automation.
- Multiple PacketExpert™ can be controlled remotely from single client application via MAPS™ CLI server.
- Support for a wide range of tests setup, interfaces, protocols, and script languages.
- Python, C# client access through MAPS™ CLI Server.
- High Level APIs allows to access PacketExpert™ functionalities.
- Scripts for MAC, VLAN, MPLS, IP and UDP layers testing
- Remote monitoring capability.
- Requires additional licensing for CLI support across various PacketExpert™ platforms.

Working Principle of MAPS™ CLI Client/Server Architecture

MAPS™ CLI Client/Server platform supports various client libraries in different languages, so that users can make use of these different libraries to communicate with the MAPS™ server, and achieve automation using their language of choice. However, these are relatively low-level libraries, which gives users a very fine grain control.

For PacketExpert™ platform, a set of relatively High-Level APIs have been developed on top of the MAPS™ Client library, which greatly reduces the time to develop sample applications and achieve automation. These APIs are developed in the respective languages and are easy-to-use and intuitive. E.g.: C# APIs are provided by means of API classes for each application. Similarly, Python APIs are provided through API scripts that implement API classes for different applications. Also supplied are sample applications, that users can use to work with APIs. Using these high-level APIs and sample applications, users can develop automated tests in a very short period.

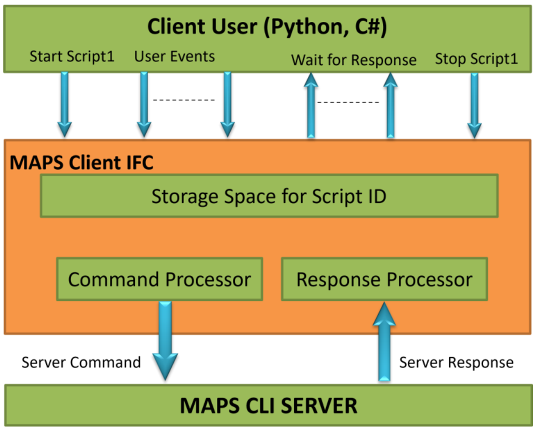


PacketExpert™ MAPS™ CLI Working Principle

CLI Functional Modules

CLI application consists of 3 functional modules.

- Client Users (Python/C#) - Acts as User Interface which executes Python/C# Scripts instructing the CLI/API server to run the particular script to perform the specific test like BERT/RFC 2544, etc.
- MAPS™ Client Interface (MAPS Client IFC) - acts as an interface between MAPS™ CLI Server and its clients Python/C#. The MAPS™ Python/C# Client application includes a dll file, a packaged library that enables communication with the MAPS™ Server from the client environment.
- MAPS™ CLI Server - is an executable which inherits all features of MAPS™ GUI. MAPS™ CLI/API Server is a scripting based frame work which controls the PacketExpert™ hardware using proprietary MAPS™ scripts.



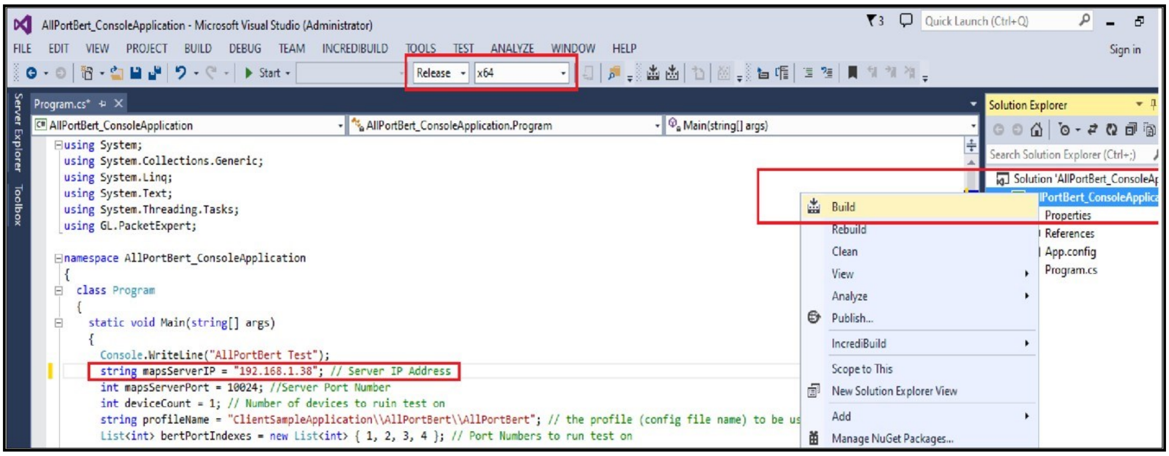
MAPS™ CLI Functional Modules

C# Client and Scripting

The C# interface developed for PacketExpert™ allows users to control all features of PacketExpert™ through C# APIs. The C# interface is implemented based on a client-server model. The C# client connects to the MAPS™ CLI server using TCP/IP sockets. MAPS™ CLI Server interfaces with PacketExpert™ low level API controlling the hardware. There will be different MAPS™ scripts to implement different applications like BERT, RFC 2544 etc.,

The MAPS C# Interface (MAPS Client IFC) application includes a MAPSCSAPI.dll file, a packaged library that enables communication with the MAPS™ CLI Server from a C# environment.

C# Client invokes APIs which executes the command, that instructs the MAPS™ CLI Server to run the particular script which performs the particular PacketExpert™ tests like BERT, RFC 2544 etc.

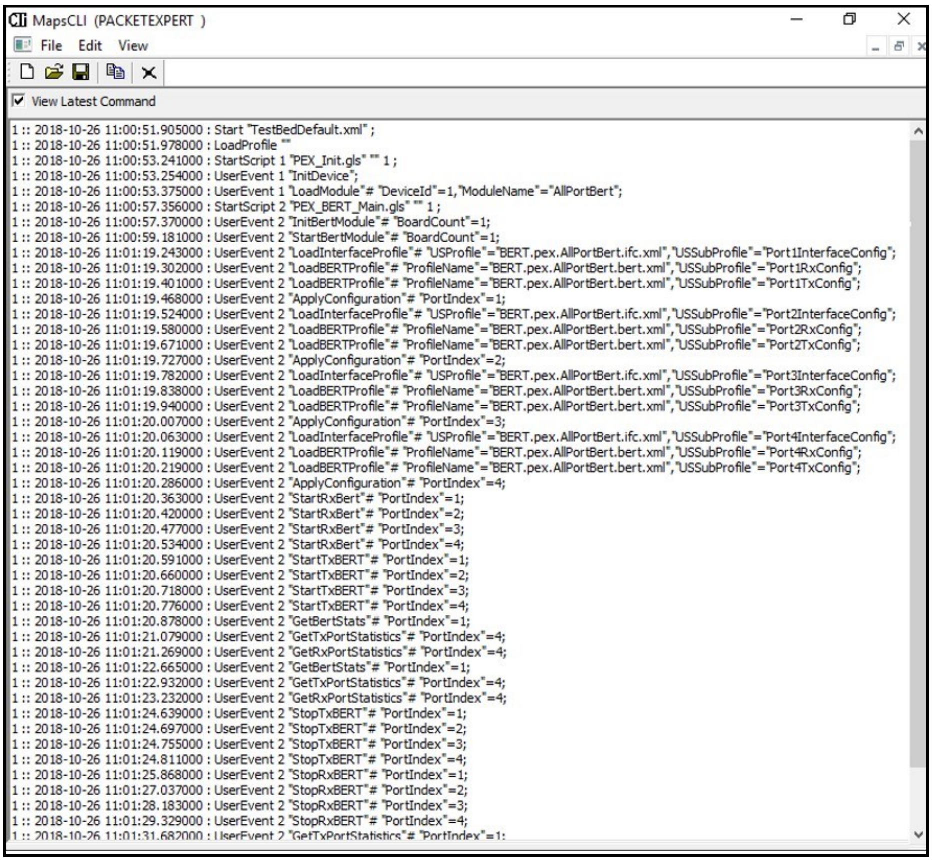

AllPortBert_ConsoleApplication - Microsoft Visual Studio (Administrator)
Quick Launch (Ctrl+Q)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM INCREDIBUILD TOOLS TEST ANALYZE WINDOW HELP
Sign in
Start
Release
x64
Server Explorer
Toolbox
Program.cs*
AllPortBert_ConsoleApplication
AllPortBert_ConsoleApplication.Program
Main(string[] args)
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AllPortBert_ConsoleAp
lPortBert_ConsoleApplic
Properties
References
App.config
Program.cs
Build
Rebuild
Clean
View
Analyze
Publish...
IncrediBuild
Scope to This
New Solution Explorer View
Add
Manage NuGet Packages...

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using GL.PacketExpert;

namespace AllPortBert_ConsoleApplication
{
  class Program
  {
    static void Main(string[] args)
    {
      Console.WriteLine("AllPortBert Test");
      string mapsServerIP = "192.168.1.38"; // Server IP Address
      int mapsServerPort = 10024; //Server Port Number
      int deviceCount = 1; // Number of devices to ruin test on
      string profileName = "ClientSampleApplication\\AllPortBert\\AllPortBert"; // the profile (config file name) to be us
      List<int> bertPortIndexes = new List<int> { 1, 2, 3, 4 }; // Port Numbers to run test on
```

C# Sample API Client

MAPS™ CLI Server (C#)



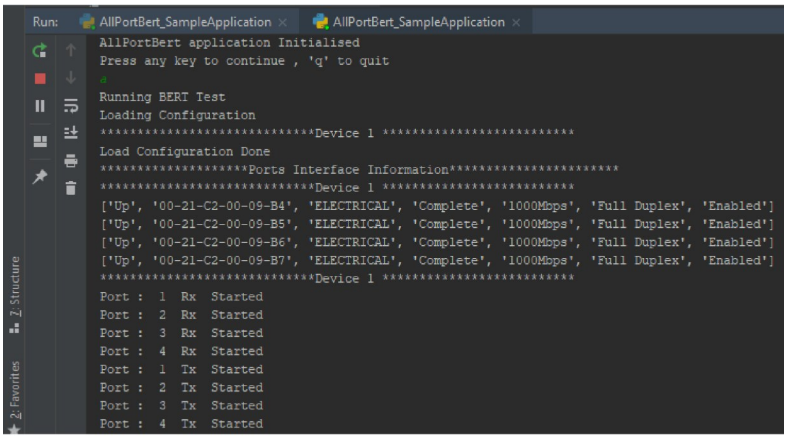
```
1 :: 2018-10-26 11:00:51.905000 : Start "TestBedDefault.xml" ;
1 :: 2018-10-26 11:00:51.978000 : LoadProfile ""
1 :: 2018-10-26 11:00:53.241000 : StartScript 1 "PEX_Init.gls" "" 1 ;
1 :: 2018-10-26 11:00:53.254000 : UserEvent 1 "InitDevice";
1 :: 2018-10-26 11:00:53.375000 : UserEvent 1 "LoadModule"# "DeviceId"=1,"ModuleName"="AllPortBert";
1 :: 2018-10-26 11:00:57.356000 : StartScript 2 "PEX_BERT_Main.gls" "" 1 ;
1 :: 2018-10-26 11:00:57.370000 : UserEvent 2 "InitBertModule"# "BoardCount"=1;
1 :: 2018-10-26 11:00:59.181000 : UserEvent 2 "StartBertModule"# "BoardCount"=1;
1 :: 2018-10-26 11:01:19.243000 : UserEvent 2 "LoadInterfaceProfile"# "USProfile"="BERT.pex.AllPortBert.ifc.xml","USSubProfile"="Port1InterfaceConfig";
1 :: 2018-10-26 11:01:19.302000 : UserEvent 2 "LoadBERTProfile"# "ProfileName"="BERT.pex.AllPortBert.bert.xml","USSubProfile"="Port1RxConfig";
1 :: 2018-10-26 11:01:19.401000 : UserEvent 2 "LoadBERTProfile"# "ProfileName"="BERT.pex.AllPortBert.bert.xml","USSubProfile"="Port1TxConfig";
1 :: 2018-10-26 11:01:19.468000 : UserEvent 2 "ApplyConfiguration"# "PortIndex"=1;
1 :: 2018-10-26 11:01:19.524000 : UserEvent 2 "LoadInterfaceProfile"# "USProfile"="BERT.pex.AllPortBert.ifc.xml","USSubProfile"="Port2InterfaceConfig";
1 :: 2018-10-26 11:01:19.580000 : UserEvent 2 "LoadBERTProfile"# "ProfileName"="BERT.pex.AllPortBert.bert.xml","USSubProfile"="Port2RxConfig";
1 :: 2018-10-26 11:01:19.671000 : UserEvent 2 "LoadBERTProfile"# "ProfileName"="BERT.pex.AllPortBert.bert.xml","USSubProfile"="Port2TxConfig";
1 :: 2018-10-26 11:01:19.727000 : UserEvent 2 "ApplyConfiguration"# "PortIndex"=2;
1 :: 2018-10-26 11:01:19.782000 : UserEvent 2 "LoadInterfaceProfile"# "USProfile"="BERT.pex.AllPortBert.ifc.xml","USSubProfile"="Port3InterfaceConfig";
1 :: 2018-10-26 11:01:19.838000 : UserEvent 2 "LoadBERTProfile"# "ProfileName"="BERT.pex.AllPortBert.bert.xml","USSubProfile"="Port3RxConfig";
1 :: 2018-10-26 11:01:19.940000 : UserEvent 2 "LoadBERTProfile"# "ProfileName"="BERT.pex.AllPortBert.bert.xml","USSubProfile"="Port3TxConfig";
1 :: 2018-10-26 11:01:20.007000 : UserEvent 2 "ApplyConfiguration"# "PortIndex"=3;
1 :: 2018-10-26 11:01:20.063000 : UserEvent 2 "LoadInterfaceProfile"# "USProfile"="BERT.pex.AllPortBert.ifc.xml","USSubProfile"="Port4InterfaceConfig";
1 :: 2018-10-26 11:01:20.119000 : UserEvent 2 "LoadBERTProfile"# "ProfileName"="BERT.pex.AllPortBert.bert.xml","USSubProfile"="Port4RxConfig";
1 :: 2018-10-26 11:01:20.219000 : UserEvent 2 "LoadBERTProfile"# "ProfileName"="BERT.pex.AllPortBert.bert.xml","USSubProfile"="Port4TxConfig";
1 :: 2018-10-26 11:01:20.286000 : UserEvent 2 "ApplyConfiguration"# "PortIndex"=4;
1 :: 2018-10-26 11:01:20.363000 : UserEvent 2 "StartRxBert"# "PortIndex"=1;
1 :: 2018-10-26 11:01:20.420000 : UserEvent 2 "StartRxBert"# "PortIndex"=2;
1 :: 2018-10-26 11:01:20.477000 : UserEvent 2 "StartRxBert"# "PortIndex"=3;
1 :: 2018-10-26 11:01:20.534000 : UserEvent 2 "StartRxBert"# "PortIndex"=4;
1 :: 2018-10-26 11:01:20.591000 : UserEvent 2 "StartTxBERT"# "PortIndex"=1;
1 :: 2018-10-26 11:01:20.660000 : UserEvent 2 "StartTxBERT"# "PortIndex"=2;
1 :: 2018-10-26 11:01:20.718000 : UserEvent 2 "StartTxBERT"# "PortIndex"=3;
1 :: 2018-10-26 11:01:20.776000 : UserEvent 2 "StartTxBERT"# "PortIndex"=4;
1 :: 2018-10-26 11:01:20.878000 : UserEvent 2 "GetBertStats"# "PortIndex"=1;
1 :: 2018-10-26 11:01:21.079000 : UserEvent 2 "GetTxPortStatistics"# "PortIndex"=4;
1 :: 2018-10-26 11:01:21.269000 : UserEvent 2 "GetRxPortStatistics"# "PortIndex"=4;
1 :: 2018-10-26 11:01:22.665000 : UserEvent 2 "GetBertStats"# "PortIndex"=1;
1 :: 2018-10-26 11:01:22.932000 : UserEvent 2 "GetTxPortStatistics"# "PortIndex"=4;
1 :: 2018-10-26 11:01:23.232000 : UserEvent 2 "GetRxPortStatistics"# "PortIndex"=4;
1 :: 2018-10-26 11:01:24.639000 : UserEvent 2 "StopTxBERT"# "PortIndex"=1;
1 :: 2018-10-26 11:01:24.697000 : UserEvent 2 "StopTxBERT"# "PortIndex"=2;
1 :: 2018-10-26 11:01:24.755000 : UserEvent 2 "StopTxBERT"# "PortIndex"=3;
1 :: 2018-10-26 11:01:24.811000 : UserEvent 2 "StopTxBERT"# "PortIndex"=4;
1 :: 2018-10-26 11:01:25.868000 : UserEvent 2 "StopRxBERT"# "PortIndex"=1;
1 :: 2018-10-26 11:01:27.037000 : UserEvent 2 "StopRxBERT"# "PortIndex"=2;
1 :: 2018-10-26 11:01:28.183000 : UserEvent 2 "StopRxBERT"# "PortIndex"=3;
1 :: 2018-10-26 11:01:29.329000 : UserEvent 2 "StopRxBERT"# "PortIndex"=4;
1 :: 2018-10-26 11:01:31.682000 : UserEvent 2 "GetTxPortStatistics"# "PortIndex"=1;
```

MAPS™ CLI Server (C#)

Python Client and Scripting

The Python interface developed for PacketExpert™ allows users to control all features of PacketExpert™ through Python APIs. The Python interface is implemented based on a client-server model. The server is the MAPS™ CLI server, which interfaces with the PacketExpert™ hardware through the USB. The client consists of a Python API dll and Python API scripts, which allows user to control the MAPS™ CLI server, issue commands and get back results.

The MAPS™ Python Interface (MAPS Client IFC) application includes a PythonMapsCliIfc.pyd file or PythonMapsCliIfc.so , a packaged library that enables communication with the MAPS™ CLI Server from a Python environment. MAPS Client IFC provides added benefits of a fully capable flow control engine with built commands.



```
AllPortBert application Initialised
Press any key to continue , 'q' to quit
a
Running BERT Test
Loading Configuration
*****************************Device 1 **************************
Load Configuration Done
********************Ports Interface Information**********************
*****************************Device 1 **************************
['Up', '00-21-C2-00-09-B4', 'ELECTRICAL', 'Complete', '1000Mbps', 'Full Duplex', 'Enabled']
['Up', '00-21-C2-00-09-B5', 'ELECTRICAL', 'Complete', '1000Mbps', 'Full Duplex', 'Enabled']
['Up', '00-21-C2-00-09-B6', 'ELECTRICAL', 'Complete', '1000Mbps', 'Full Duplex', 'Enabled']
['Up', '00-21-C2-00-09-B7', 'ELECTRICAL', 'Complete', '1000Mbps', 'Full Duplex', 'Enabled']
*****************************Device 1 **************************
Port :  1  Rx  Started
Port :  2  Rx  Started
Port :  3  Rx  Started
Port :  4  Rx  Started
Port :  1  Tx  Started
Port :  2  Tx  Started
Port :  3  Tx  Started
Port :  4  Tx  Started
```

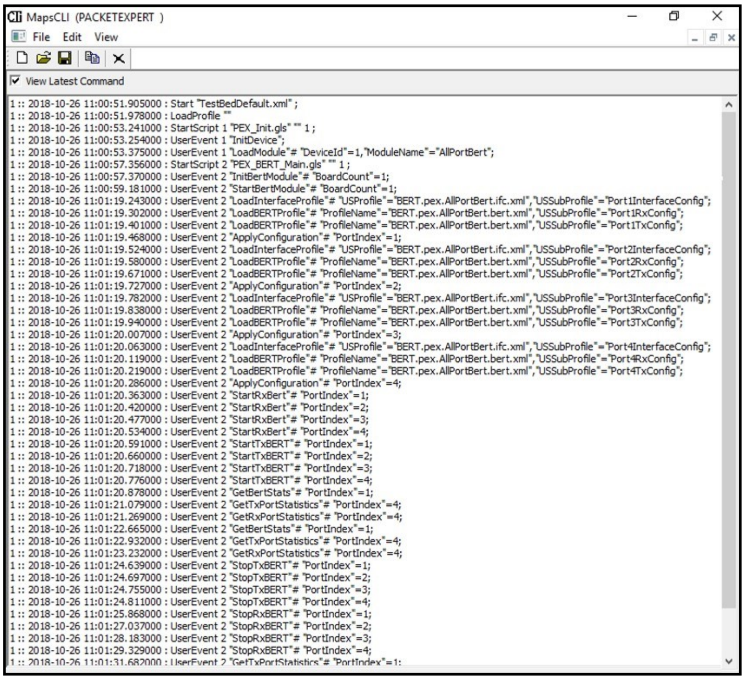
Figure: Executing the Client script in "PyCharm" console

MAPS™ CLI Server

The client connects to the MAPS™ CLI server via TCP/IP sockets.

PacketExpert™ will run the MAPS™ CLI Server, which can interface with PacketExpert™ low level API to control hardware. There are different MAPS™ scripts to implement various applications like BERT, RFC 2544 etc.

MAPS™ CLI Server consists of GL's proprietary scripts (.gls files) that actually implement various PacketExpert™ functionalities like BERT, RFC 2544 etc. XML files (called as "Profiles") containing the configuration information for the test. PacketExpert™ internal low level APIs are used within MAPS™ scripts to control the PacketExpert™ hardware.



```
1 :: 2018-10-26 11:00:51.905000 : Start "TestBedDefault.xml" ;
1 :: 2018-10-26 11:00:51.978000 : LoadProfile ""
1 :: 2018-10-26 11:00:53.241000 : StartScript 1 "PEX_Init.gls" "" 1 ;
1 :: 2018-10-26 11:00:53.254000 : UserEvent 1 "InitDevice";
1 :: 2018-10-26 11:00:53.375000 : UserEvent 1 "LoadModule"# "DeviceId"=1,"ModuleName"="AllPortBert";
1 :: 2018-10-26 11:00:57.356000 : StartScript 2 "PEX_BERT_Main.gls" "" 1 ;
1 :: 2018-10-26 11:00:57.370000 : UserEvent 2 "InitBertModule"# "BoardCount"=1;
1 :: 2018-10-26 11:00:59.181000 : UserEvent 2 "StartBertModule"# "BoardCount"=1;
1 :: 2018-10-26 11:01:19.243000 : UserEvent 2 "LoadInterfaceProfile"# "USProfile"="BERT.pex.AllPortBert.ifc.xml","USSubProfile"="Port1InterfaceConfig";
1 :: 2018-10-26 11:01:19.302000 : UserEvent 2 "LoadBERTProfile"# "ProfileName"="BERT.pex.AllPortBert.bert.xml","USSubProfile"="Port1RxConfig";
1 :: 2018-10-26 11:01:19.401000 : UserEvent 2 "LoadBERTProfile"# "ProfileName"="BERT.pex.AllPortBert.bert.xml","USSubProfile"="Port1TxConfig";
1 :: 2018-10-26 11:01:19.468000 : UserEvent 2 "ApplyConfiguration"# "PortIndex"=1;
1 :: 2018-10-26 11:01:19.524000 : UserEvent 2 "LoadInterfaceProfile"# "USProfile"="BERT.pex.AllPortBert.ifc.xml","USSubProfile"="Port2InterfaceConfig";
1 :: 2018-10-26 11:01:19.580000 : UserEvent 2 "LoadBERTProfile"# "ProfileName"="BERT.pex.AllPortBert.bert.xml","USSubProfile"="Port2RxConfig";
1 :: 2018-10-26 11:01:19.671000 : UserEvent 2 "LoadBERTProfile"# "ProfileName"="BERT.pex.AllPortBert.bert.xml","USSubProfile"="Port2TxConfig";
1 :: 2018-10-26 11:01:19.727000 : UserEvent 2 "ApplyConfiguration"# "PortIndex"=2;
1 :: 2018-10-26 11:01:19.782000 : UserEvent 2 "LoadInterfaceProfile"# "USProfile"="BERT.pex.AllPortBert.ifc.xml","USSubProfile"="Port3InterfaceConfig";
1 :: 2018-10-26 11:01:19.838000 : UserEvent 2 "LoadBERTProfile"# "ProfileName"="BERT.pex.AllPortBert.bert.xml","USSubProfile"="Port3RxConfig";
1 :: 2018-10-26 11:01:19.940000 : UserEvent 2 "LoadBERTProfile"# "ProfileName"="BERT.pex.AllPortBert.bert.xml","USSubProfile"="Port3TxConfig";
1 :: 2018-10-26 11:01:20.007000 : UserEvent 2 "ApplyConfiguration"# "PortIndex"=3;
1 :: 2018-10-26 11:01:20.063000 : UserEvent 2 "LoadInterfaceProfile"# "USProfile"="BERT.pex.AllPortBert.ifc.xml","USSubProfile"="Port4InterfaceConfig";
1 :: 2018-10-26 11:01:20.119000 : UserEvent 2 "LoadBERTProfile"# "ProfileName"="BERT.pex.AllPortBert.bert.xml","USSubProfile"="Port4RxConfig";
1 :: 2018-10-26 11:01:20.219000 : UserEvent 2 "LoadBERTProfile"# "ProfileName"="BERT.pex.AllPortBert.bert.xml","USSubProfile"="Port4TxConfig";
1 :: 2018-10-26 11:01:20.286000 : UserEvent 2 "ApplyConfiguration"# "PortIndex"=4;
1 :: 2018-10-26 11:01:20.363000 : UserEvent 2 "StartRxBert"# "PortIndex"=1;
1 :: 2018-10-26 11:01:20.420000 : UserEvent 2 "StartRxBert"# "PortIndex"=2;
1 :: 2018-10-26 11:01:20.477000 : UserEvent 2 "StartRxBert"# "PortIndex"=3;
1 :: 2018-10-26 11:01:20.534000 : UserEvent 2 "StartRxBert"# "PortIndex"=4;
1 :: 2018-10-26 11:01:20.591000 : UserEvent 2 "StartTxBERT"# "PortIndex"=1;
1 :: 2018-10-26 11:01:20.660000 : UserEvent 2 "StartTxBERT"# "PortIndex"=2;
1 :: 2018-10-26 11:01:20.718000 : UserEvent 2 "StartTxBERT"# "PortIndex"=3;
1 :: 2018-10-26 11:01:20.776000 : UserEvent 2 "StartTxBERT"# "PortIndex"=4;
1 :: 2018-10-26 11:01:20.878000 : UserEvent 2 "GetBertStats"# "PortIndex"=1;
1 :: 2018-10-26 11:01:21.079000 : UserEvent 2 "GetTxPortStatistics"# "PortIndex"=4;
1 :: 2018-10-26 11:01:21.269000 : UserEvent 2 "GetRxPortStatistics"# "PortIndex"=4;
1 :: 2018-10-26 11:01:22.665000 : UserEvent 2 "GetBertStats"# "PortIndex"=1;
1 :: 2018-10-26 11:01:22.932000 : UserEvent 2 "GetTxPortStatistics"# "PortIndex"=4;
1 :: 2018-10-26 11:01:23.232000 : UserEvent 2 "GetRxPortStatistics"# "PortIndex"=4;
1 :: 2018-10-26 11:01:24.639000 : UserEvent 2 "StopTxBERT"# "PortIndex"=1;
1 :: 2018-10-26 11:01:24.697000 : UserEvent 2 "StopTxBERT"# "PortIndex"=2;
1 :: 2018-10-26 11:01:24.755000 : UserEvent 2 "StopTxBERT"# "PortIndex"=3;
1 :: 2018-10-26 11:01:24.811000 : UserEvent 2 "StopTxBERT"# "PortIndex"=4;
1 :: 2018-10-26 11:01:25.868000 : UserEvent 2 "StopRxBERT"# "PortIndex"=1;
1 :: 2018-10-26 11:01:27.037000 : UserEvent 2 "StopRxBERT"# "PortIndex"=2;
1 :: 2018-10-26 11:01:28.183000 : UserEvent 2 "StopRxBERT"# "PortIndex"=3;
1 :: 2018-10-26 11:01:29.329000 : UserEvent 2 "StopRxBERT"# "PortIndex"=4;
1 :: 2018-10-26 11:01:31.682000 : UserEvent 2 "GetTxPortStatistics"# "PortIndex"=1;
```

MAPS CLI Server (Python)

LabVIEW / Test Stand Integration

Using PacketExpert™ APIs, it is very easy to integrate PacketExpert™ into LabVIEW. Since LabVIEW supports C# language, the PacketExpert™ C# API dll can be directly imported into LabVIEW and used in the Graphical environment that LabVIEW provides to control PacketExpert™ devices and automate testing.

With LabVIEW, it is easy to create flexible test scripts that control multiple hardware, customize test system with graphical programming, included analysis, drag and drop interface. This makes the system compatible with GL's PacketExpert™ software. Eg: Importing the C# Client API dll into LabVIEW instantly provides ability to run any PacketExpert™ test application like – BERT, RFC2544, Loopback and others.

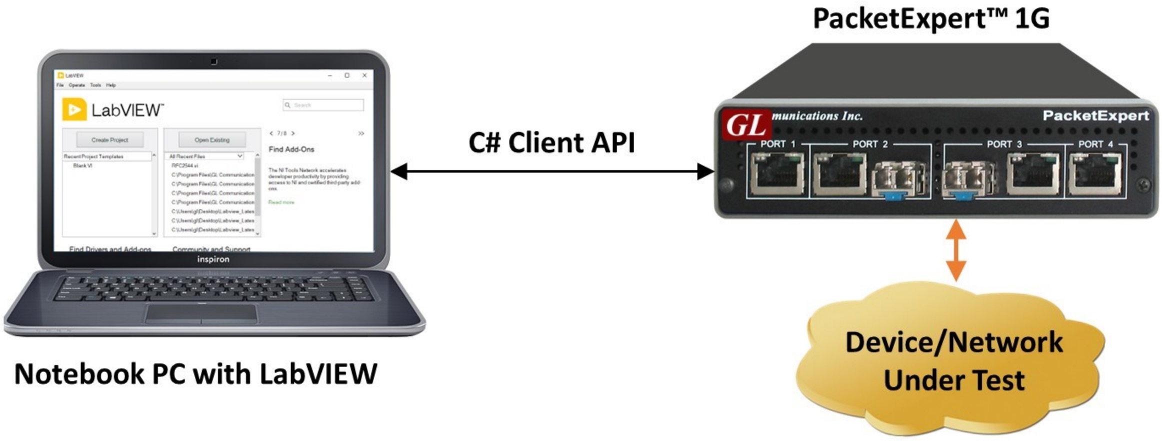
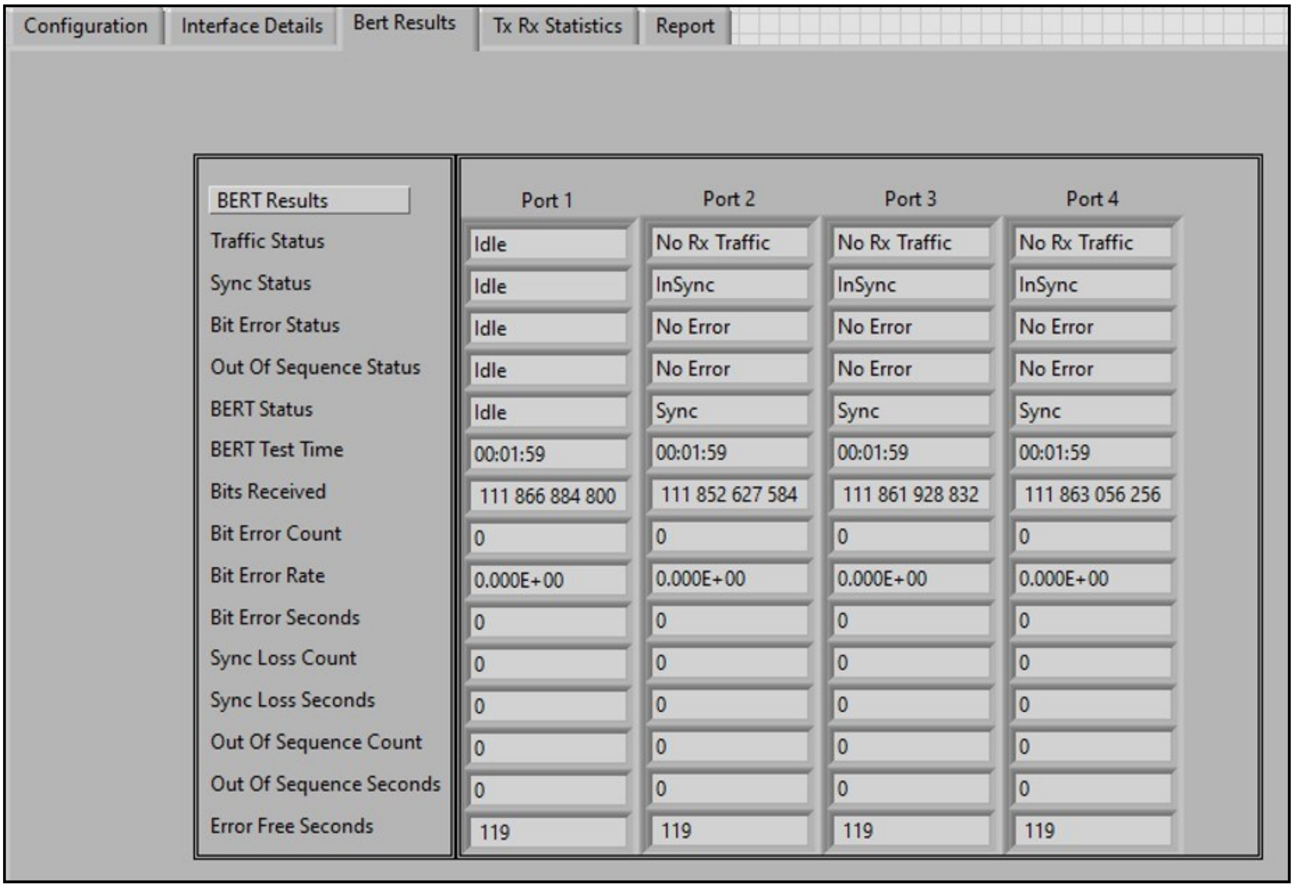


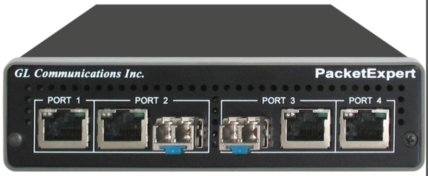
Figure: LabVIEW with PacketExpert™



BERT Results	Port 1	Port 2	Port 3	Port 4
Traffic Status	Idle	No Rx Traffic	No Rx Traffic	No Rx Traffic
Sync Status	Idle	InSync	InSync	InSync
Bit Error Status	Idle	No Error	No Error	No Error
Out Of Sequence Status	Idle	No Error	No Error	No Error
BERT Status	Idle	Sync	Sync	Sync
BERT Test Time	00:01:59	00:01:59	00:01:59	00:01:59
Bits Received	111 866 884 800	111 852 627 584	111 861 928 832	111 863 056 256
Bit Error Count	0	0	0	0
Bit Error Rate	0.000E+00	0.000E+00	0.000E+00	0.000E+00
Bit Error Seconds	0	0	0	0
Sync Loss Count	0	0	0	0
Sync Loss Seconds	0	0	0	0
Out Of Sequence Count	0	0	0	0
Out Of Sequence Seconds	0	0	0	0
Error Free Seconds	119	119	119	119

Figure: LabVIEW Test Results

Hardware Specifications

 Portable 1G Hardware Unit	 1U mTOP™ PacketExpert™ 1G Rack Unit (3 PXE100s) Stacked 1U PacketExpert™ 1G Rack Unit (6 PXE100s)	 PacketExpert™ 1G mTOP™ Probe
Physical Specification: Length: 8.45 in. (214.63 mm) Width: 5.55 in. (140.97 mm) Height: 1.60 in (40.64 mm) Weight: 1.66 lbs. (0.75 kg)	**Dimension:** Length: 16 Inches Width: 19 Inches Height: Stacked 1U Rack (HD-PacketExpert-24) or 1U mTOP™ (HD-PacketExpert-12)	**Physical Specification:** Length: 10.4 in. (264.16 mm) Width: 8.4 in. (213.36 mm) Height: 3.0 in. (76.2 mm)
Bus Interface: USB 2.0 or USB 3.0 **Power Supply**: +12 Volts (Medical Grade), 3 Amps	**SBC Specifications:** • Intel Core i3 or optional i7 NUC Equivalent, • Windows® 11 64-bit Pro Operating System • USB 3.0 and USB 2.0 Ports, ATX Power Supply • USB Type C Ports, Ethernet 2.5GigE port • 256 GB Hard drive, 8G Memory (Min) • Two HDMI ports	**SBC Specifications:** • Intel Core i3 or optional i7 NUC Equivalent, • Windows® 11 64-bit Pro Operating System • USB 3.0 and USB 2.0 Ports, 12V/3A Power Supply • USB Type C Ports, Ethernet 2.5GigE port • 256 GB Hard drive, 8G Memory (Min) • Two HDMI ports
Interface: 2 x 10 / 100 / 1000 Base-T Electrical only. 2 x 1000 Base-X Optical OR 10/100/1000 Base-T Electrical. Single Mode or Multi Mode Fiber SFP support with LC connector.	**Interface:** 12 Total Ethernet Ports (HD-PacketExpert-12) • mTOP™ System (embedded SBC, 3x PXE100) • PacketExpert™ 1G (PXE100) interfaces - ⇒ 6x 1000 Base-X Optical OR 10/100/1000 Base -T Electrical ⇒ 6x (10/100/1000) Base-T Electrical 24 Total Ethernet Ports (HD-PacketExpert-24) • mTOP™ System (embedded SBC, 6x PXE100) • PacketExpert™ 1G (PXE100) interfaces - ⇒ 12x 1000 Base-X Optical OR 10/100/1000 Base-T Electrical ⇒ 12x (10/100/1000) Base-T Electrical	**Interface:** 4x Total Ethernet ports 2x 10/100/1000 Base-T Electrical only 2x 1000 Base-X Optical OR 10/100/1000 Base-T Electrical Single Mode or Multi Mode Fiber SFP support with LC connector

Temperature: Operating Temperature: +5 to +40C
Non-Operating Temperature: -30° to +60° C

Humidity: Operating Humidity: 0% to 80% RH
Non-Operating Humidity: 0% to 95% RH

Altitude: Operating Altitude: up to 10,000 feet
Non-Operating Altitude: up to 50,000 feet

Protocols:
RFC 2544 compliance

Pelican Carry Case

Buyer’s Guide

Item No	Product Description
PXE100	PacketExpert™ 1G
CXE100	CLI Server for PXE100
PXE112	PacketExpert™ 1G – SA (12-Port)
PXE124	PacketExpert™ 1G – SA (24-Port)
MT001	mTOP 1U Rack Mount Enclosure w/SBC
PXN100	PacketExpert™ 10GX
CXN100	CLI Server for PXN100

Note: PCs which include GL hardware/software require Intel or AMD processors for compliance.

For more information, visit PacketExpert™ webpage.