

**TekExpress® PCI Express
Transmitter Compliance and Testing Solution Software
Printable Application Help**



**TekExpress® PCI Express
Transmitter Compliance and Testing Solution Software
Printable Application Help**

Copyright © Tektronix. All rights reserved. Licensed software products are owned by Tektronix or its subsidiaries or suppliers, and are protected by national copyright laws and international treaty provisions. Tektronix products are covered by U.S. and foreign patents, issued and pending. Information in this publication supersedes that in all previously published material. Specifications and price change privileges reserved.

TEKTRONIX and TEK are registered trademarks of Tektronix, Inc.

Contacting Tektronix

Tektronix, Inc.
14150 SW Karl Braun Drive
P.O. Box 500
Beaverton, OR 97077
USA

For product information, sales, service, and technical support:

- In North America, call 1-800-833-9200.
- Worldwide, visit www.tektronix.com to find contacts in your area.

Table of Contents

| | |
|---------------|-----|
| Welcome | vii |
|---------------|-----|

Getting help and support

| | |
|--------------------------------|---|
| Related documentation | 1 |
| Conventions used in help | 2 |
| Technical support | 2 |

Getting started

| | |
|---|----|
| Installing the software | 3 |
| Minimum system requirements | 3 |
| Supported oscilloscopes | 4 |
| Windows 7 user account settings | 5 |
| Install the software | 5 |
| Set application file permissions | 7 |
| Map the my TekExpress folder | 8 |
| Activate the license | 9 |
| View software version and license information | 9 |
| Introduction to the application | 11 |
| Application directories and their contents | 11 |
| File name extensions | 12 |

Operating basics

| | |
|--------------------------------------|----|
| Run the application | 13 |
| Exit the application | 14 |
| Application controls and menus | 14 |
| Application controls | 14 |
| Options menu | 16 |
| Application panels | 22 |
| Application panel overview | 22 |
| Setup panel | 24 |
| Status panel overview | 41 |
| Results panel | 43 |
| Reports panel | 46 |

Setting up and configuring tests

| | |
|----------------------------------|----|
| About setting up tests | 51 |
| Equipment connection setup | 51 |

Running tests

| | |
|---------------------------|----|
| Test setup overview | 53 |
| About running tests | 53 |
| Prerun checklist | 54 |

Saving and recalling test setups

| | |
|--|----|
| About test setups | 55 |
| Save a test setup | 56 |
| Open (load) a saved test setup | 57 |
| Create a new test setup based on an existing one | 57 |

TekExpress programmatic interface

| | |
|---|----|
| About the programmatic interface | 59 |
| To enable remote access | 60 |
| Requirements for developing TekExpress client | 62 |
| Client programmatic interface example | 63 |
| Program remote access code example | 66 |
| Python and C# examples | 67 |
| PCIe application commands | 67 |
| PCIe application commands listing | 67 |
| Connect through an IP address | 75 |
| Lock the server | 76 |
| Disable the popups | 77 |
| Set or get the DUT ID | 79 |
| Select the PCIe device | 80 |
| Select the suite | 81 |
| Set the PCIe test version | 82 |
| Set the data rate parameter | 83 |
| Set the test mode parameter | 84 |
| Set the prerecorded waveform execution mode | 86 |
| Set the 5 Gb/s preemphasis parameter | 88 |
| Set the SSC parameter | 89 |

| | |
|--|-----|
| Set the voltage swing parameter | 91 |
| Set the signal quality preset parameter | 93 |
| Set the preset lanes parameter | 94 |
| Set the lane source parameter | 96 |
| Set the preset parameter | 99 |
| Set the acquisition parameter | 101 |
| Set the analysis mode parameter | 103 |
| Set the sigtest version parameter | 104 |
| Set the on failure action parameter | 107 |
| Set the report update mode parameter | 108 |
| Set the append report parameter | 110 |
| Set the crosstalk parameter | 112 |
| Set the waveform save parameter | 113 |
| Set the link analysis embed-de-embed signal parameter | 115 |
| Set the link analysis embed-de-embed filter file parameter | 117 |
| Set the link analysis other filter file parameter | 119 |
| Set the link analysis equalization dropdown parameter | 121 |
| Set the link analysis CTLE index parameter | 123 |
| Set the link analysis DFE parameter | 124 |
| Set the DUT auto toggle parameter | 126 |
| Set the DUT auto toggle options parameter | 127 |
| Set AFG signal type parameter | 129 |
| Set the AFG signal frequency parameter | 130 |
| Set the AFG signal amplitude parameter | 132 |
| Set the burst count parameter | 133 |
| Set the record length parameter | 135 |
| Set the sample rate parameter | 136 |
| Set the bandwidth parameter | 138 |
| Set the signal validation parameter | 139 |
| Set the slot number parameter | 141 |
| Set SigTest Interface Mode | 142 |
| Set the trigger type parameters | 144 |
| Set Sig Validation Threshold | 145 |
| Set Toggle from Gen3P10 to Gen1 | 147 |
| Set the group test results by parameters | 149 |
| Set the report creation path parameter | 150 |
| Set the report contents to save parameters | 152 |
| Set the report creation type parameter | 155 |
| Set the auto increment report name if duplicate parameter | 156 |

| | |
|---|-----|
| Set the view report after generating parameter | 158 |
| Run with set configurations or stop the run operation | 159 |
| Handle error codes | 160 |
| Save recall or query a saved session | 162 |
| Get or set the timeout value | 164 |
| Wait for the test to complete | 165 |
| After the test is complete | 169 |
| Unlock the server | 175 |
| Disconnect from the server | 176 |

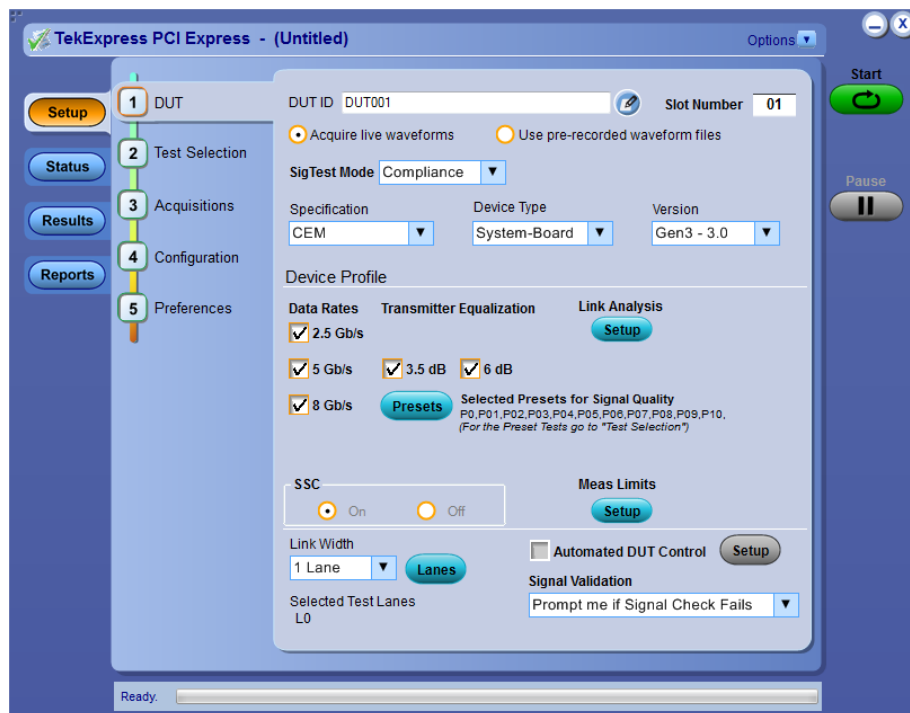
SCPI commands

| | |
|--|-----|
| About SCPI command | 177 |
| Socket configuration for SCPI commands | 177 |
| TEKEXP:*IDN? | 187 |
| TEKEXP:*OPC? | 187 |
| TEKEXP:ACQUIRE_MODE | 188 |
| TEKEXP:ACQUIRE_MODE? | 188 |
| TEKEXP:EXPORT | 189 |
| TEKEXP:INFO? | 189 |
| TEKEXP:INSTRUMENT | 190 |
| TEKEXP:INSTRUMENT? | 191 |
| TEKEXP:LASTERROR? | 191 |
| TEKEXP:LIST? | 192 |
| TEKEXP:MODE | 193 |
| TEKEXP:MODE? | 193 |
| TEKEXP:POPUP | 194 |
| TEKEXP:POPUP? | 194 |
| TEKEXP:REPORT | 195 |
| TEKEXP:REPORT? | 195 |
| TEKEXP:RESULT? | 196 |
| TEKEXP:SELECT | 197 |
| TEKEXP:SELECT? | 198 |
| TEKEXP:SETUP | 198 |
| TEKEXP:STATE | 199 |
| TEKEXP:STATE? | 199 |
| TEKEXP:VALUE | 200 |
| TEKEXP:VALUE? | 201 |
| Command parameters list | 202 |

Reference

| | |
|-----------------------------------|-----|
| De-embed using filter files | 207 |
| Setup files | 208 |

Welcome



Welcome to the TekExpress[®] PCI Express Automated Test Solution Software application (referred to as TekExpress PCIe or PCIe in the rest of the document). TekExpress PCIe provides an automated, simple, and efficient way to test PCI Express interfaces and devices consistent to the requirements of the PCI Express specifications.

Tek Express PCIe key features and benefits

New features from current release

- Support U.2 (SFF-8639) specification, provided the machine has SigTest v3.2.0 with U.2 templates
- Trigger type support for Gen3 (Auto/Width/Edge)
- Support new SigTest v3.2.0 template with pattern check
- TekExpress setup files in-line with PCI-SIG Compliance Workshop
- Faster test execution time with improved Autoset

Features of previous release

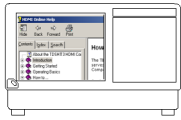
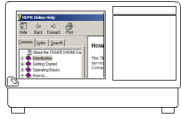
- Automates compliance measurements for PCI Express 3.0 CEM Specification Rev 0.9. for the following configurations:
 - PCIE_1_0a- PCIEX_TX_ADD_CON_250UI
 - PCIE_1_0a- PCIEX_TX_SYS_CON_250UI
 - PCIE_CEM_CARD_1_1
 - PCIE_CEM_SYS_1_1
 - PCIE_2_0_CARD
 - PCIE_2_0_SYS
 - PCIE_3_0_CARD
 - PCIE_3_0_SYS
- Fully automated General, Jitter, Composite Eye, Transition Eye, and Non Transition Eye measurements
- Provides both an automation solution (for compliance) and DPOJET (for debug)
- The PCI-SIG[®] PCI Express Compliance Test Library is integrated into the TekExpress framework
- Reduces the time required to conduct testing
- Minimizes user intervention when conducting time-consuming testing
- Enables loading filter files to support system and add-in card measurements
- Performs fully-automated testing for system and add-in card measurements
- Provides individual or group test selection by using a tree-structure menu
- Built-in reporting features:
 - Provides a Pass/Fail summary table
 - Provides margin details on each test
 - Provides a consolidated report for all tests
- Complete programmatic interface enables automation scripts to call PCIe functions

Getting help and support

Related documentation

The following manuals are available as part of the TekExpress PCIe Compliance and Debug Solution documentation set.

Table 1: Product documentation

| Item | Purpose | Location |
|-----------------------------|--|---|
| Application Help | In-depth operation and UI help |  |
| PDF of the Application Help | Printable version of the compiled Application help |  |

See also [Technical support](#)

Conventions used in help

Application Help uses the following conventions:

- The term “DUT” is an abbreviation for Device Under Test.
- The term “select” is a generic term that applies to the two methods of choosing an option: using a mouse or using the touch screen.

Technical support

Tektronix values your feedback on our products. To help us serve you better, please send us your suggestions, ideas, or comments on your application or oscilloscope. Contact Tektronix through mail, telephone, or the Web site. See [Contacting Tektronix](#) for more information.

When you contact Tektronix Technical Support, please include the following information (be as specific as possible):

General information

- All instrument model numbers
- Hardware options, if any
- Probes used
- Your name, company, mailing address, phone number, FAX number
- Please indicate if you would like to be contacted by Tektronix about your suggestion or comments.

Application specific information

- Software version number
- Description of the problem such that technical support can duplicate the problem
- If possible, save the setup files for all the instruments used and the application
- If possible, save the TekExpress setup files, log.xml, *.TekX (session files and folders), and status messages text file
- If possible, save the waveform on which you are performing the measurement as a .wfm file

Getting started

Installing the software

Minimum system requirements

The following table shows the minimum system requirements needed for an oscilloscope to run TekExpress PCI Express.

Table 2: System requirements

| Component | Requirement |
|--|--|
| Oscilloscope | See Supported oscilloscopes |
| Processor | Same as the oscilloscope |
| Operating system | Microsoft Windows 7 (64-bit only) Required Windows 7 user account settings |
| Memory | Same as the oscilloscope |
| Hard disk | Same as the oscilloscope |
| Display | Same as the oscilloscope ¹ |
| Firmware | Tekscope 10.3.0 or later for MSO/DSA/DPO70000C,D,DX |
| Software | <ul style="list-style-type: none">■ DPOJET, Jitter and Eye Diagram Analysis Tool ²■ Microsoft .NET 4.0 Framework■ Microsoft Internet Explorer 8.0 SP1 or later■ PyVisa version 1.0.0■ IronPython version 2.7.3■ Microsoft Photo Editor 3.0 or equivalent software for viewing image files■ Adobe Reader 7.0 or equivalent software for viewing portable document format (PDF) files |
| Arbitrary Function Generator (AFG) ³ (for automatic test pattern toggling) | <ul style="list-style-type: none">■ Tektronix AFG3252, AFG3252C |

¹ If TekExpress is running on an instrument having a video resolution lower than 800x600 (for example, a sampling oscilloscope), it is recommended that you connect a secondary monitor, which must be enabled before launching the application.

² For software version, refer to Readme TekExpress PCI Express.txt file at C:\Program Files (x86)\Tektronix\TekExpress\TekExpress PCI Express

³ The listed AFG/AWG instruments support both differential inputs (requires 2 channels) and 100 MHz burst mode.

| Component | Requirement |
|---|--|
| Arbitrary Waveform Generator (AWG) (for automatic test pattern toggling) | <ul style="list-style-type: none"> ■ Tektronix AWG5002B/C, AWG5012B/C, AWG5014B/C ■ Tektronix AWG7082B/C, AWG7122B/C ■ Tektronix AWG70001A, AWG70002A |
| Other devices | <ul style="list-style-type: none"> ■ SMP-SMA cables ■ TCA-SMA connectors ■ Matched pair cables ■ Tektronix P7313 SMA Differential Probe |

Supported oscilloscopes

The TekExpress PCIe application runs on the following Tektronix oscilloscopes:

- MSO70604⁴, DPO/MSO70604C (Gen1 testing only)
- MSO70804⁴, DPO/MSO70804C (Gen1 and Gen2 testing only)
- MSO71254⁴, DPO/MSO71254C (Gen1, Gen2, and Gen3 testing)
- MSO71604⁴, DPO/MSO71604C (Gen1, Gen2, and Gen3 testing)
- MSO72004⁴, DPO/MSO72004C (Gen1, Gen2, and Gen3 testing)
- DPO/DSA72504D (Gen 1, Gen2, and Gen 3 testing)
- DPO/DSA73304D (Gen 1, Gen 2, and Gen 3 testing)
- DPO/MSO72304DX (Gen1, Gen2, and Gen3 testing)
- DPO/MSO72504DX (Gen1, Gen2 and Gen3 testing)
- DPO/MSO73304DX (Gen1, Gen2 and Gen3 testing)
- DPO72304SX (Gen1, Gen2 and Gen3 testing)
- DPO73304SX (Gen1, Gen2 and Gen3 testing)
- DPO75002SX [Standalone⁵ or 2 stack] (Gen1, Gen2 and Gen3 testing)
- DPO75902SX [Standalone⁵ or 2 stack] (Gen1, Gen2 and Gen3 testing)
- DPO77002SX [Standalone⁵ or 2 stack] (Gen1, Gen2 and Gen3 testing)

See also.

[Minimum system requirements](#)

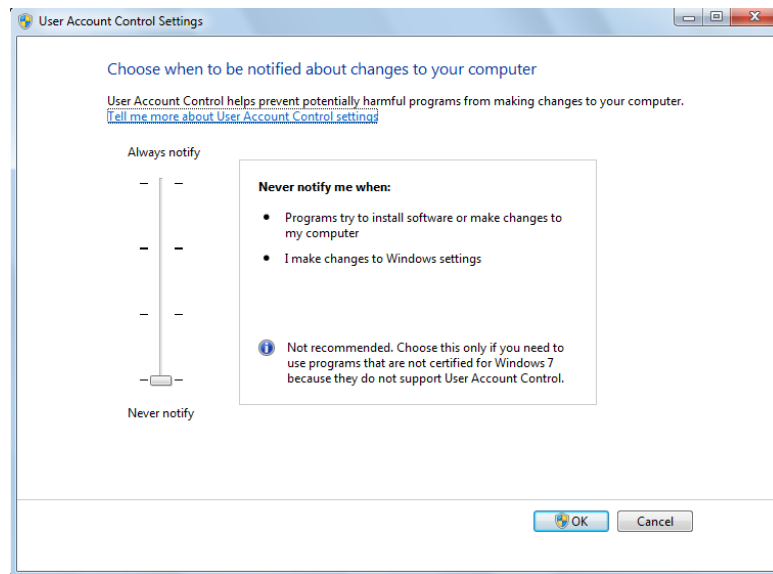
⁴ Requires Microsoft Windows 7 (64-bit) operating system. Contact your local Tektronix Customer Service representative for upgrade information.

⁵ Standalone can be used only for CEM specification Add-In-Card device type or U.2(SFF8639) specification Module device type

Windows 7 user account settings

Windows 7 instruments need to have the User Account Control Settings set to **Never Notify**. To set User Account Control Settings:

1. Go to **Control Panel > User Accounts > Change User Account Control settings**.
2. Set it to **Never Notify** as shown in the image.

**See also.**

[Supported oscilloscopes](#)

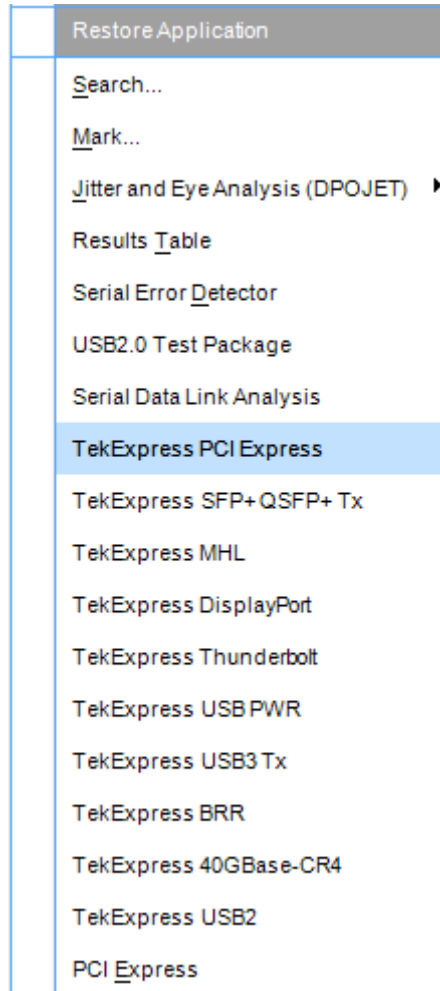
Install the software

Use the following steps to install PCI Express software on any compatible instrument running Microsoft Windows 7 (64-bit). See [Minimum System Requirements](#) for details.

1. Close all applications (including the TekScope application).
2. Go to the www.tek.com Web site and search for TekExpress PCI Express to locate the installation file. Download the file TekExpress_PCIE_Deployment_Package.exe.
3. Copy or download the PCIe installer file to the oscilloscope.
4. Double-click the installer .exe file to extract the installation files and launch the InstallShield Wizard. Follow the on-screen instructions. The software installs in the following location:

C:\Program Files (x86)\Tektronix\TekExpress\TekExpress PCI Express

5. The installer updates the TekScope Analyze menu to include the installed options.



See also.

[*Minimum system requirements*](#)

[*Supported oscilloscopes*](#)

Set application file permissions

Before you run tests for the first time, do the following:

1. Understand where your test files are stored on the instrument.

After you install and launch TekExpress PCIe, it creates the following folders on the oscilloscope:

- \My Documents\My TekExpress\PCI Express
- \My Documents\My TekExpress\PCI Express\Untitled Session

Every time you launch TekExpress PCIe, an Untitled Session folder is created in the PCIe folder. The Untitled Session folder is automatically deleted when you exit the PCIe application. To preserve your test session files, save the test setup before exiting the TekExpress application.



CAUTION.

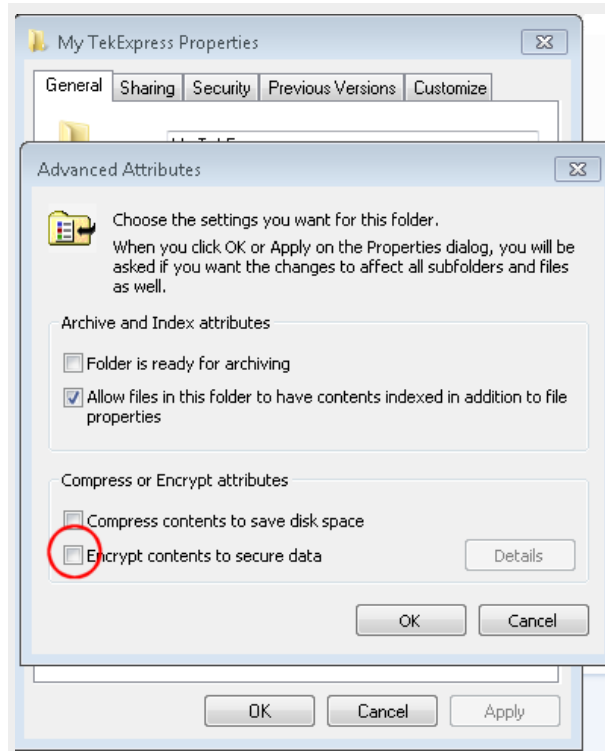
Do not modify any of the session files or folders because this may result in loss of data or corrupted session files. Each session has multiple files associated with it. When you save a session, a .TekX file, and a folder named for the session that contains associated files, is created on the oscilloscope X: drive.

2. *Map the shared My TekExpress folder* as **X:** (X drive) on the instruments used in test setups running Microsoft Windows Operating System.

The My TekExpress folder has the share name format <domain><user ID>My TekExpress. Or, if the instrument is not connected to a domain, the share name format is <instrument name><user ID>My TekExpress. This shared folder is used to save the waveform files and is used during other file transfer operations.

NOTE. *If the X: drive is mapped to any other shared folder, the application will display a warning message asking you to disconnect the X: drive manually.*

3. Make sure that the My TekExpress folder (Drive X:) has read and write access:
 - a. Right-click the folder and select **Properties**.
 - b. Select the **General** tab and then click **Advanced**.
 - c. In the Advanced Attributes dialog box, make sure that the option **Encrypt contents to secure data** is NOT selected (not checked). Example.



4. See the [prerun checklist](#) before you run a test.

See also.

[Configuration test parameters](#)

[View test-related files](#)

[Application directories and usage](#)

[File name extensions](#)

Map the my TekExpress folder

Follow these steps to map the My TekExpress folder on the instrument:

1. Open Windows Explorer.
2. From the Windows Explorer menu, click **Computer**.
3. In the menu bar, select **Map network drive**.
4. Select the Drive letter as **X:** (if there is any previous connection on X:, disconnect it first through **Tools > Disconnect Network drive** menu of Windows Explorer. Windows 7 users: if you do not see the Tools menu, press the **Alt** key).
5. In the Folder field, enter the remote My TekExpress folder path (for example, \\192.158.97.65\ My TekExpress).

To determine the IP address of the instrument where the My TekExpress folder exists, do the following:

1. On the instrument where the My TekExpress folder exists, click **Start** and select **Run**.
2. Type **cmd** and then press **Enter**.
3. At the command prompt, type **ipconfig** and then press **Enter**.

See also.

[Before you click start](#)

[Install the software](#)

Activate the license

Activate the license using the Option Installation wizard on the oscilloscope. Instructions for using the Options Installation window to activate licenses for installed applications is provided in the oscilloscope online Help:

1. From the oscilloscope menu bar, click **Utilities > Option Installation**.

The TekScope Option Installation wizard opens.

2. Push the **F1** key on the oscilloscope keyboard to open the Option Installation help topic. Follow the directions in the topic to activate the license.

See also.

[View version and license information](#)

View software version and license information

Use the following instructions to view version information for the application and for the application modules such as the Programmatic Interface and the Programmatic Interface Client.

To view version information for PCIe:

1. In the PCIe application, click the **Options** button and select **About TekExpress**.

The About Tektronix TekExpress PCI Express dialog box appears, showing the version details.



To view license and option key information:

1. From the TekScope menu, select **Help > About TekScope**.
2. Scroll through the Options section list to locate PCI Express.
3. To view the Option key, look below the **Options** list.

See also.

[Activate the license](#)

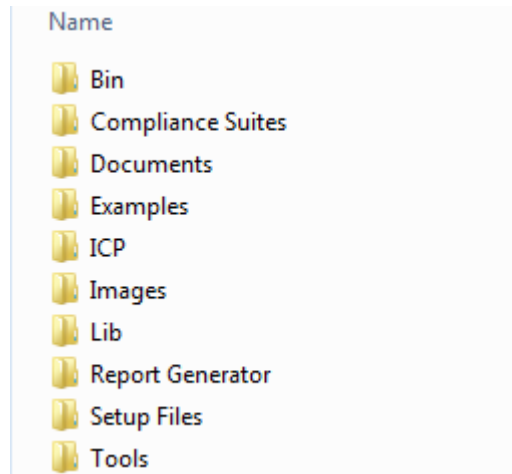
[Options menu](#)

Introduction to the application

Application directories and their contents

TekExpress PCIe application. The TekExpress PCIe application files are installed at "C:\Program Files (x86)\Tektronix\TekExpress\TekExpress PCI Express".

It contains the following folders:



The following table lists the application directory names and their purpose.

Table 3: Application directories and usage

| Directory names | Usage |
|-------------------|---|
| ACP | Contains instrument and PCIe application-specific interface libraries |
| Bin | Contains miscellaneous PCIe application libraries |
| Compliance Suites | Contains compliance-specific files |
| Data Manager | Contains result management-specific libraries of the PCIe application |
| Data Storage | Contains libraries needed for storing data |
| Documents | Contains the technical documentation for the PCIe application |
| Examples | Contains various support files and example Python and C# test files |
| ICP | Contains instrument and PCIe application-specific interface libraries |
| Lib | Contains utility files specific to the PCIe application |
| Report Generator | Contains style sheets for report generation |

| Directory names | Usage |
|-----------------------------|--|
| Setup files | Contains PCI-SIG work shop compliant TekExpress setup files for System-Bard, Add-In-Card of CEM specification and Host, Module of U.2(SFF8639) specification |
| Tools | Contains instrument and PCIe application-specific files |

TekExpress TekSig. The TekExpress TekSig application files are installed at C:\Program Files (x86)\Tektronix\TekSigPackage.

See also.

[View test-related files](#)

[File name extensions](#)

File name extensions

The TekExpress PCIe application uses the following file name extensions:

| File name extension | Description |
|---------------------|--|
| .TekX | Application session files (the extensions may not be displayed) |
| .py | Python test file. See the TekExpress PCI Express\Examples folder for a sample file |
| .xml | Test-specific configuration information (encrypted) file Application log file |
| .wfm | Test waveform file |
| .mht | Test result reports (default). Test reports can also be saved in HTML format |
| .flt | Filter files |
| .chm, pdf | Help manuals |

See also.

[Select report options](#)

[View test-related files](#)

[Application directories and their contents](#)

[Before you click start](#)

Operating basics

Run the application

To launch the PCIe application, do either of the following:

- Select **Analyze > TekExpress PCI Express** from the TekScope menu.
- Double-click any saved PCIe session file (<file name>.TekX).

When you first run the application after installation, the application checks for a file called Resources.xml located at C:\Users\<>username>\My TekExpress\PCI Express. The Resources.xml file gets mapped to the X: drive when the application launches. Session files are then stored inside the X:\PCI Express folder.

The Resources.xml file contains information about available network-connected instruments. If this file is not found, the application runs an instrument discovery program, before launching PCIe, to locate available instruments.



To keep the application window on top, select **Keep On Top** from the PCIe *Options menu*. If the application goes behind the oscilloscope application, click **Analyze > TekExpress PCI Express** to move the application to be in front.


See also [Activate the license](#)

Exit the application

Use the following method to exit the application:


NOTE. Using other methods to exit the application results in abnormal termination of the application.








1.  Click  on the application title bar.
2. Do one of the following:
 - If you have an unsaved session or test setup, you are asked to save it before exiting. To save it, click **Yes**. Otherwise click **No**. The application closes.
 - A message box appears asking if you really want to exit TekExpress. To exit, click **Yes**.

NOTE. To minimise the application, click  on the application title bar.

Application controls and menus

Application controls **Table 4: Application controls descriptions**

| Item | Description |
|---|---|
| <p><i>Options menu</i></p>  | <p>Menu to display global application controls</p> |
| <p><i>Panel buttons</i></p>  | <p>Controls that open panels for configuring test settings and options.</p> |

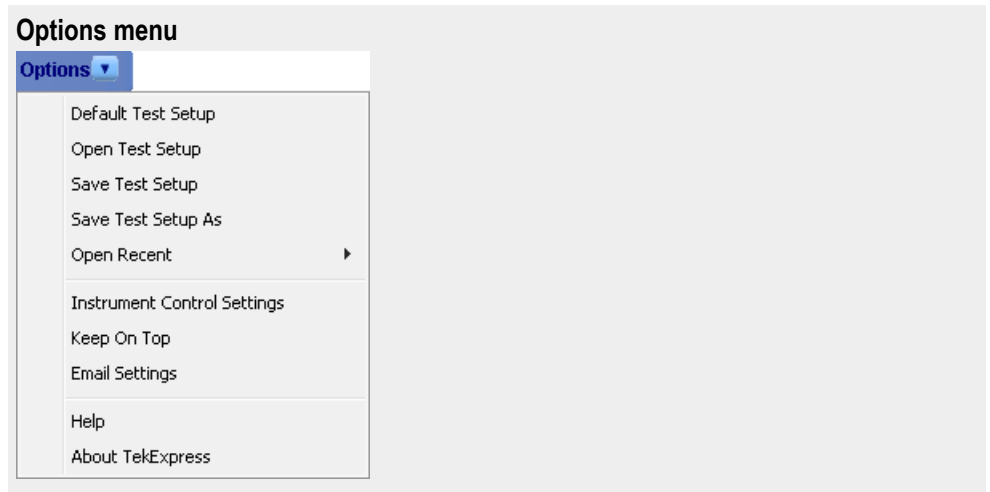
| Item | Description |
|--|--|
| <p>Start/Stop button</p> <p>Start</p>  <p>Stop</p>  | <p>Use the Start button to start the test run of the measurements in the selected order. If prior acquired measurements have not been cleared, the new measurements are added to the existing set.</p> <p>The button toggles to the Stop mode while tests are running. Use the Stop button to abort the test.</p> |
| <p>Pause \ Continue button</p> <p>Pause</p>  <p>Continue</p>  | <p>Use the Pause button to temporarily interrupt the current acquisition. When a test is paused, the button name changes to "Continue."</p> |
| <p>Clear button</p> <p>Clear</p>  | <p>Use the Clear button to clear all existing measurement results. Adding or deleting a measurement, or changing a configuration parameter of an existing measurement, also clears measurements. This is to prevent the accumulation of measurement statistics or sets of statistics that are not coherent. This button is available only on the Results pane.</p> |
| <p>Minimize button</p>  | <p>Minimizes the application.</p> |
| <p>Close button</p>  | <p>Exits the application.</p> |
| <p>Application window move</p> | <p>Place the cursor over the application window and drag it to the desired location.</p> |

Options menu **Options menu overview.** The Options menu is located in the upper right corner of the application.

The Options menu has the following selections:

| Menu | Function |
|------------------------------------|--|
| Default Test Setup | Opens an untitled test setup with defaults selected |
| Open Test Setup | Opens a saved test setup |
| Save Test Setup | Saves the current test setup selections ¹ |
| Save Test Setup As | Creates a new test setup based on an existing one ¹ |
| Open Recent | Displays a menu of recently opened test setups to select from |
| <i>Instrument Control Settings</i> | Shows the list of instruments connected to the test setup and allows you to locate and refresh connections to those instruments |
| Keep On Top | Keeps the TekExpress PCIe utility on top of other open windows on the desktop |
| <i>Email Settings</i> | Use to configure email options for test run and results notifications |
| Help | Displays the TekExpress PCIe Online help |
| About TekExpress | <ul style="list-style-type: none"> ■ Displays application details such as software name, version number, and copyright ■ Provides access to <i>license information</i> for your PCIe installation ■ Provides a link to the Tektronix Web site |

¹ In pre-recorded mode, waveform recall will not be successful if the session name is lengthy, i.e. more than 10 characters.

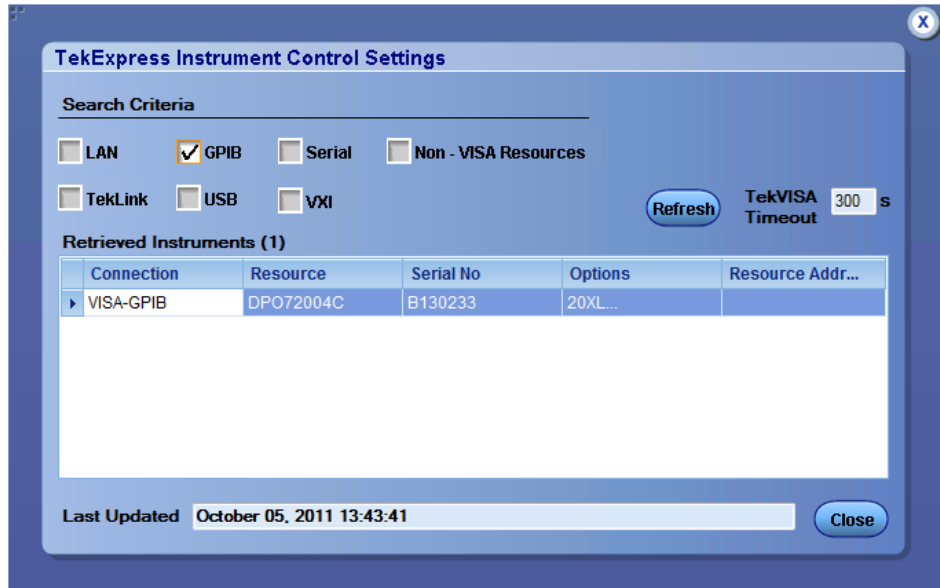


See also.

[Application controls](#)

Instrument control settings dialog box. Use the TekExpress Instrument Control Settings dialog box to search for and list the connected resources (instruments) found on specified connections (LAN, GPIB, USB, and so on) and each instruments connection information. You access this dialog box from the Options menu.

Access this dialog box from the **Options** menu.



Use the Instrument Control Settings feature to *search for connected instruments* and view instrument connection details. Connected instruments displayed here can be selected for use under Global Settings in the test configuration section.

See also.

[Options menu overview](#)

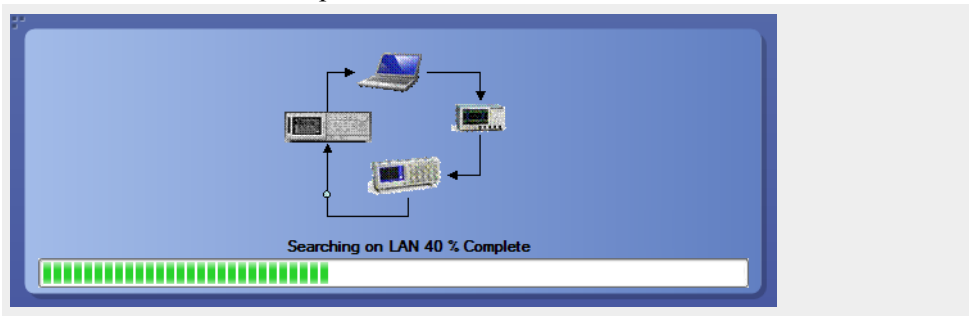
View connected instruments. Use the Instrument Control Settings dialog box to view or search for connected instruments required for the tests. The application uses TekVISA to discover the connected instruments.

To refresh the list of connected instruments:

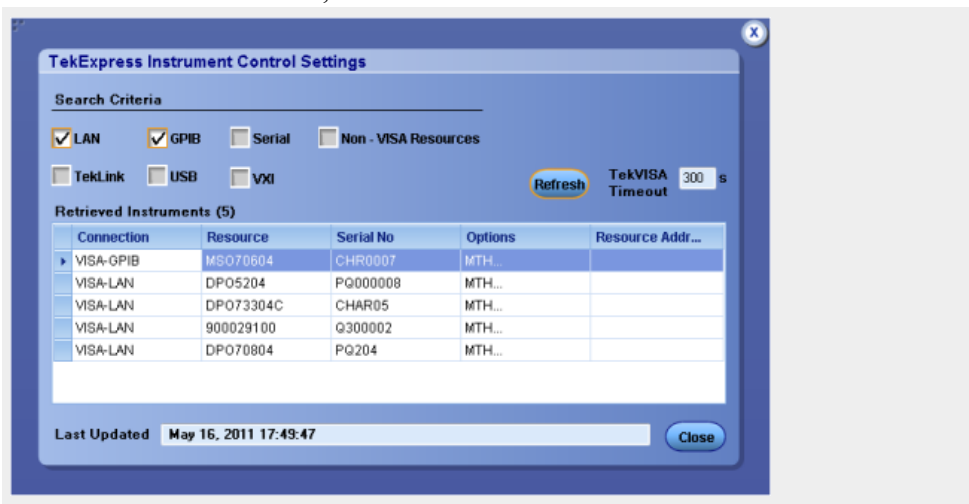
1. From the Options menu, select **Instrument Control Settings**.
2. In the Search Criteria section of the Instrument Control Settings dialog box, select the connection types of the instruments for which to search.

Instrument search is based on the VISA layer, but different connections determine the resource type, such as LAN, GPIB, and USB. For example, if you choose LAN, the search will include all the instruments supported by TekExpress that are communicating over the LAN. If the search does not find any instruments that match a selected resource type, a message appears telling you that no such instruments were found.

3. Click **Refresh**. TekExpress searches for connected instruments.



4. After discovery, the dialog box lists the instrument-related details based on the search criteria you selected. For example, if you selected LAN and GPIB as the search criteria, the application checks for the availability of instruments over LAN, then GPIB.



The details of the instruments are displayed in the Retrieved Instruments table. The time and date of instrument refresh is displayed in the Last Updated field.

See also.

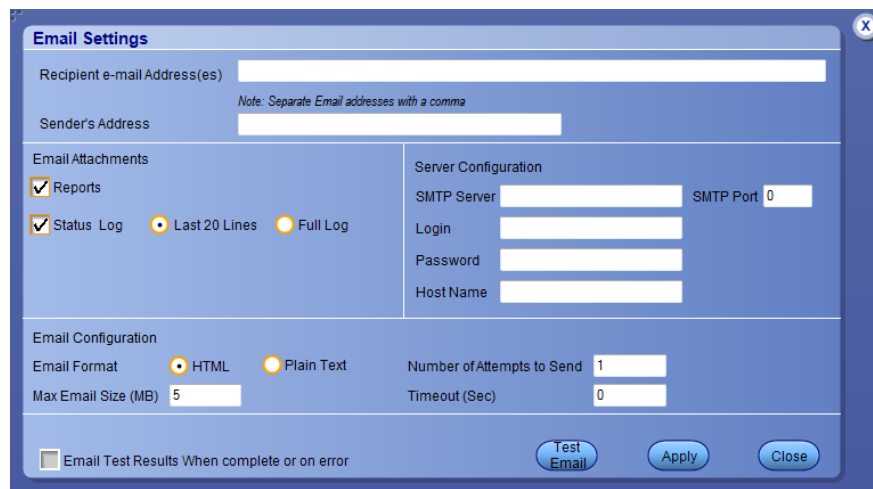
[Configuration test parameters](#)

Equipment connection setup

Email settings. Use the Email Settings utility to *configure email notifications* to receive notifications when a test completes, produces an error, or fails. Select the type of test session information to include with the email, such as test reports and test logs, the email message format, and the email message size limit.

Access this dialog box from the Options menu.

NOTE. *Recipient email address, sender's address, and SMTP Server are mandatory fields.*



See also.

Configure email settings

Options menu

Select test notification preferences

Configure email settings. To be notified by email when a test completes, fails, or produces an error, configure the email settings.

1. **Options > Email Settings** to open the Email Settings dialog box.
2. (Required) For Recipient email Address(es), enter one or more email addresses to which to send the test notification. To include multiple addresses, separate the addresses with commas.
3. (Required) For Sender's Address, enter the email address used by the instrument. This address consists of the instrument name followed by an underscore followed by the instrument serial number, then the @ symbol and the email server used. For example:
DPO72016C_B130099@yourcompany.com.
4. (Required) In the Server Configuration section, type the SMTP Server address of the Mail server configured at the client location, and the SMTP Port number, in the corresponding fields.

If this server requires password authentication, enter a valid login name, password, and host name in the corresponding fields.

NOTE. *If any of the above required fields are left blank, the settings will not be saved and email notifications will not be sent.*

5. In the Email Attachments section, select from the following options:
 - **Reports:** Select to receive the test report with the notification email.
 - **Status Log:** Select to receive the test status log with the notification email. If you select this option, then also select whether you want to receive the full log or just the last 20 lines.
6. In the Email Configuration section:
 - Select the message file format to send: HTML (the default) or plain text.
 - Enter a maximum file size for the email message. Messages with attachments larger than this limit will not be sent. The default is 5 MB.
 - Enter the number in the Number of Attempts to Send field, to limit the number of attempts that the system makes to send a notification. The default is 1. You can also specify a timeout period.
7. Select the **Email Test Results When complete or on error** check box. Use this check box to quickly enable or disable email notifications.
8. To test your email settings, click **Test Email**.
9. To apply your settings, click **Apply**.
10. Click **Close** when finished.

Email settings

Recipient e-mail Address(es)

Sender's Address Note: Separate Email addresses with a comma

Email Attachments

Reports

Status Log Last 20 Lines Full Log

Server Configuration

SMTP Server SMTP Port

Login

Password

Host Name

Email Configuration

Email Format HTML Plain Text

Max Email Size (MB) Number of Attempts to Send

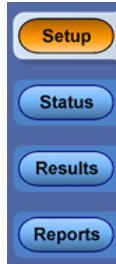
Timeout (Sec)

Email Test Results When complete or on error

Application panels

Application panel overview

Panels group related configuration, test selection, and results settings. Click a button to open the associated panel.



A panel may have one or more tabs that list the selections available in that panel. Controls in a panel can change depending on settings made in that panel or another panel.

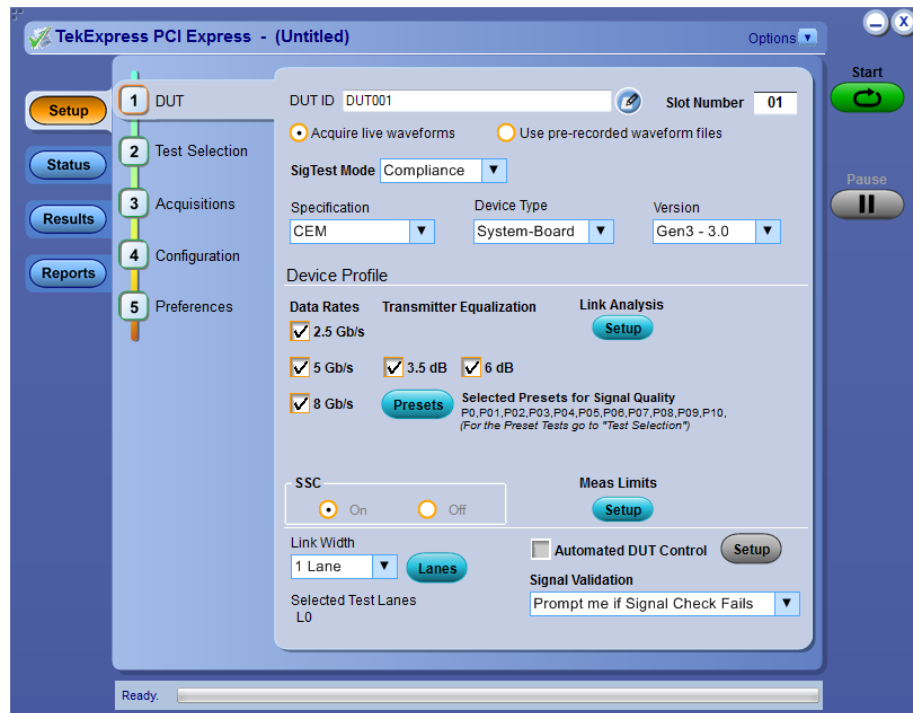
The TekExpress PCIe panels are:

Table 5: Application panels

| Panel Name | Purpose |
|-------------------------|---|
| Setup | <p>The Setup panel shows the test setup controls. Click the Setup button to open this panel. Use this panel to:</p> <ul style="list-style-type: none"> ■ Select DUT parameters. ■ Select the test(s). ■ Set acquisitions parameters for selected tests. ■ Configuration test parameters ■ Select test notification preferences. |
| Status | View the progress and analysis status of the selected tests, and view test logs. |
| Results | View a summary of test results and select result viewing preferences. |
| Reports | Browse for reports, save reports as specific file types, specify report naming conventions, select report content to include (such as summary information, detailed information, user comments, setup configuration, application configuration), and select report viewing options. |

See also.[Application controls](#)[About setting up test](#)

Setup panel **Setup panel overview.** The Setup panel contains sequentially ordered tabs that help guide you through a typical test setup process.



Use the tabs on this panel to:

Set the DUT parameters

Select test(s)

Select acquisition parameters

Set configuration parameters

Select test notification preferences

Set DUT parameters. Use the DUT tab to select parameters for the device under test. The settings are global and apply to all tests for the current session. DUT settings also affect the list of available tests in the Test Selection tab.

Click **Setup > DUT** to access the DUT parameters.

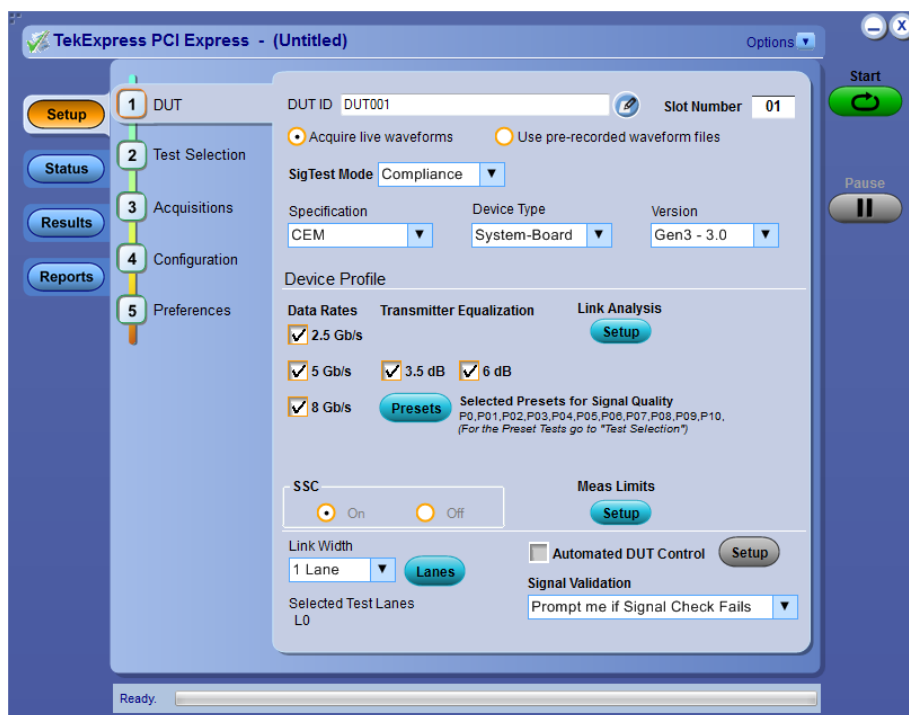



Table 6: DUT tab settings

| Setting | Description |
|--|---|
| DUT ID | Adds an optional text label for the DUT to reports. The default value is DUT001. ¹ |
| Slot Number | The slot parameter (1, 2, 4, 8, 16, or 32) of the DUT. |
|  Comments icon (to the right of the DUT ID field) | Opens a Comments dialog box in which to enter optional text to add to a report. The maximum number of characters is 256. To enable or disable comments appearing on the test report, see Select report options .) |
| Acquire live waveforms | Acquire active signals from the DUT for testing. |

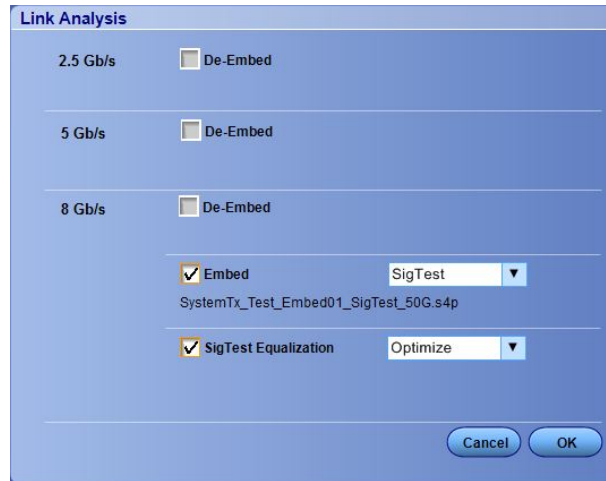
¹ In pre-recorded mode, waveform recall will not be successful if the session name is lengthy, i.e. more than 10 characters.

| Setting | Description |
|--------------------------------|--|
| Use prerecorded waveform files | Run tests on a saved waveform. Open (load) a saved test setup |
| SigTest Mode | <p>Sets the overall testing mode. Select Compliance or User Defined:</p> <ul style="list-style-type: none"> ■ Compliance: Preselects tests and parameters to meet compliance specifications for the selected version, specification, and device type. ■ User Defined: Enables the user to select specific tests and set custom parameters for tests. |
| Specification | <p>PCIe supports the CEM and U.2 (SFF8639) specification.</p> <hr/> <p>NOTE. U.2 (SFF8639) supports Gen3 (3.0) version only.</p> |
| Device Type | Sets the DUT device type (System-Board or Add-in-Card). |
| Version | Sets the DUT generation version. Available versions are Gen 1 (1.0a and 1.1), Gen2 (2.0), and Gen3 (3.0) |
| Device Profile | |
| Data Rates | Sets the data rates to test (2.5 Gb/s, 5 Gb/s, and 8 Gb/s). The data rates available depend on the selected DUT version. |
| Transmitter Equalization | <p>Sets transmitter preemphasis levels. Available for Gen 2 and Gen 3 devices.</p> <p>The application selects both preemphasis levels by default when in the compliance mode for an Add-in-Card.</p> <p>At least one preemphasis level must be selected.</p> |
| Link Analysis | Opens the Link Analysis dialog box to select custom filter files with which to perform link analysis on the source waveforms. Link Analysis dialog box |
| Presets | Opens the Presets dialog box to select the presets (P0-P10) used to perform the signal quality tests. Only available for Gen 3 DUT version. |

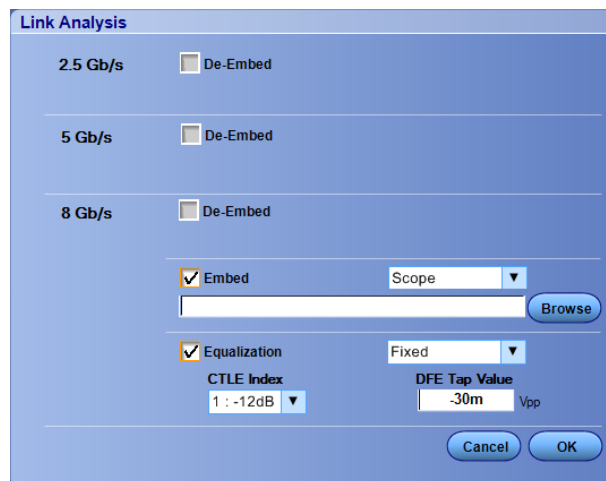
| Setting | | Description |
|--------------------------------|----------------------|--|
| Measurement Limits | Voltage Swing Limits | Sets the lane/link transmitter p-p voltage swing. This affects the limits applied to certain measurements based on the settings and does not change anything on the DUT tab. |
| | Crosstalk Limits | <p>Sets specific eye test limits depending on if the DUT design uses interleaved or non interleaved routing. This affects the limits applied to certain measurements based on the settings and does not change anything on the DUT tab. This is applicable for Gen2.</p> <ul style="list-style-type: none"> ■ When the DUT uses noninterleaved routing, select Crosstalk (noninterleaved routing). ■ When the DUT uses interleaved routing, select No Crosstalk (interleaved routing). |
| SSC (spread spectrum clocking) | | Enables or disables SSC clocking. This affects the limits applied to certain measurements based on the settings and does not change anything on the DUT tab. |
| Link Width | | Sets the link width in Lanes (1,2,4,8, or 16). |
| Lanes | | <p>Opens the Test Lane Setup dialog box to select the lanes to test. Lanes required for compliance testing are colored orange. At least one lane must be selected.</p> <p>The Link Width setting determines the number of lanes that can be tested.</p> <p>Test Lane Setup dialog box</p> |
| Automated DUT Control | | <p>Enables automatic toggling of the DUT into different test modes (generation/equalization). Requires the use of an AFG or AWG instrument. Click Setup to access the Automated DUT Control dialog box</p> |
| Signal validation | | Sets the application to validate acquisition signals and perform the specified action to take when acquired signals do not meet requirements. Select the action from the list. |

See also.[About setting up tests](#)[Select a test](#)

Link analysis dialog box. The Link Analysis dialog box lets you select custom filter files with which to perform link analysis on the source waveforms. The options available change with the each data rate. Selecting a mode enables a file browser with which to select the de-embed file.



- For 2.5 Gb/s and 5 Gb/s data rates, only the De-embed option is available.
- There are two options for the 8 Gb/s data rate Embed mode: “Scope” and “SigTest.” This selection effects the fields shown in the Equalization selections.
- There are two options for the 8 Gb/s data rate equalization mode: “Optimize” and “Fixed”. “Fixed” mode provides fields with which to set the CTLE Index and DFE Tap Value parameters.

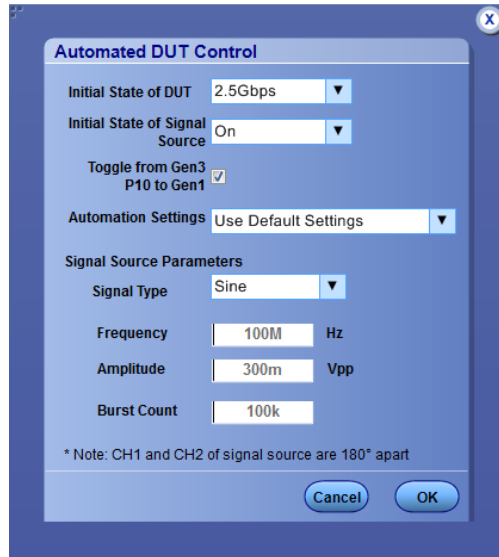


Test lane setup dialog box. The Test Lane Setup dialog box enables setting the link width and specific lanes to test. Lanes required for compliance testing are colored orange. At least one lane must be selected.



- The Link Width parameter sets the lanes that are tested.
- The lanes required for compliance testing are selected by default.
- Click **Select All** to select all the lanes.
- Click **Deselect All** to clear the selected lanes.
- Click **Compliance Lanes** to set all lanes required for compliance testing for the specified link width value.

Automated DUT control setup. The Automated DUT Control dialog box sets the parameters needed for automatic toggling of the DUT into different test modes (generation/equalization). DUT automation requires the use of a signal source (AFG or AWG).



Initial State of DUT: Sets the starting state of the DUT.

Initial State of Signal Source: Sets the AFG/AWG state to **On** (default) or **Off**. The On state enables the AFG/AWG output before the application starts signal acquisition. Some DUTs will toggle to the next signal state when the AFG/AWG initial state is On. Set the initial state to Off for these types of DUTs before running automated tests.

Toggle from Gen3 P10 to Gen1: Sets the DUT to toggle automatically to Gen1 after Gen3 P10 test execution.

Automation Settings: The Automation Settings values are Use Default Settings, Manually Configure Settings, and Custom Settings:

- **Use Default Settings:** The signal source parameters are set to predefined values as recommended by the test specification. The signal source parameter fields are disabled and cannot be edited.
- **Manually Configure Settings:** The signal source parameters are set directly at the AFG or AWG. The signal source parameter fields are disabled and cannot be edited. The PCIe application turns on or off the signal source without changing the settings.
- **Use Custom Settings:** The signal source parameters are set to the values specified in the Signal Source Parameters area. The signal source parameter fields are enabled.

Signal Type: Valid signal types are **Sine** and **Square**.

Frequency, Amplitude: Sets the AFG to output the specified frequency and amplitude values.

Burst Count: Sets the AFG to output the specified signal burst count.

NOTE. *Ch 1 and Ch 2 on the AFG source are set to 180° phase difference in all modes except Manually Configure Settings.*

NOTE. *Using DC Caps or Manual toggle, you can eliminate the automatic toggling issues that is due to DC offset.*

Select tests. Use the **Test Selection** tab to select the Signal Test and Preset Test(s) (for Gen3 only).

NOTE. *All tests are selected by default.*

1. Click **Setup > Test Selection**.
2. Select the test(s) to run:
 - Click + to expand a group of commands. Click the check box adjacent to a test group to select all tests in that group. Click check boxes adjacent to individual tests to select those tests.
 - Click **Deselect All** button to deselect all tests. All tests are selected by default.
 - Click **Select All** button to select all tests.
 - Click **Show MOI** button to open the MOI (Methods of Implementation) document for all measurements.
 - Click **Schematic** button to view a diagram that shows the correct DUT and equipment setup for the selected test. Use to verify your test equipment setup before running the test.
3. For Gen3 testing:
 - (Gen3 only) Click the **Preset Test** tab and select the presets tests.
 - (Gen3 only) Click the **Lanes** button in the Preset Test tab to view and select which lanes to use for preset testing. At least one lane must be selected.

See also.

[Set acquisition parameters](#)

[About setting up tests](#)

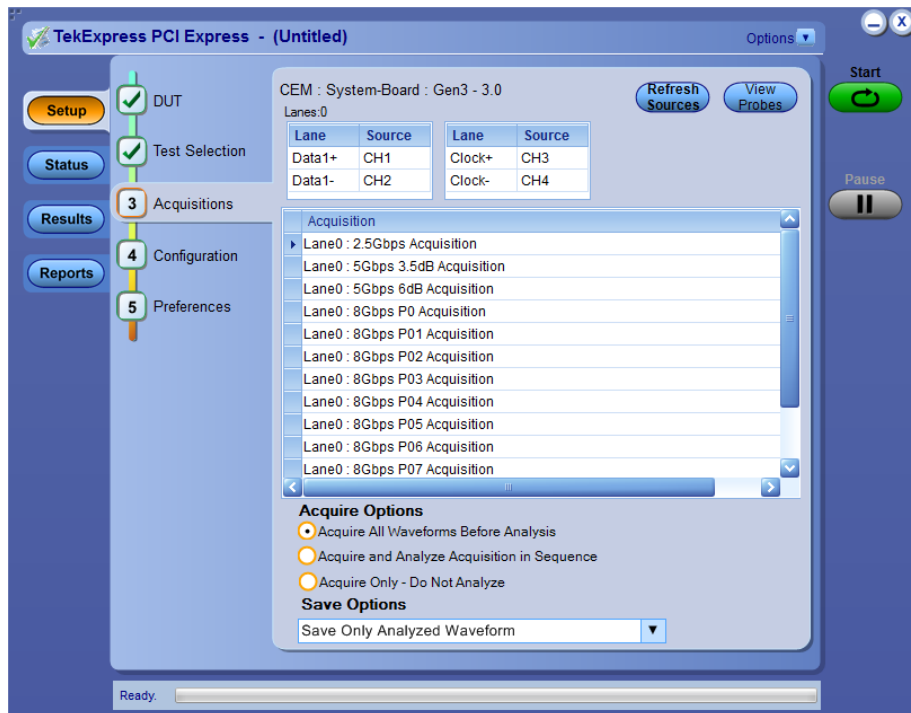
Acquisitions tab.

Set acquisition parameters. Use the **Acquisition** tab in the Setup panel to view and select test acquisition parameters, including the signal source channels, acquisition options, and waveform save options. This panel also shows the signal inputs required for the selected DUT parameters.

Contents displayed on this tab depend on whether you acquire active waveforms or use prerecorded waveform files (as set in the **DUT** tab. Contents displayed on this tab also depend on detected probes and the specified DUT type.

Active waveforms.

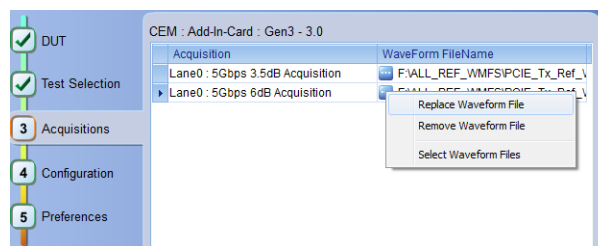
Figure 1: Acquisitions tab: using active waveforms



- Click the *Source* fields to select signal sources for the listed lanes. The number of lanes shown depends on the parameters set in the DUT tab.
- Click **Refresh Sources** to refresh the probe configuration after changing any probes. (This button performs the same function as the Refresh button in the Probe Configuration dialog box.)
- Click **View Probes** to view the detected probe configuration. Use the View Probes dialog box to enable or disable probe signal source access in the application.
- Click the *Acquire Options* controls to set how the application acquires and analyzes signals.
- Click the *Save Options* field to set how the application saves acquired waveforms (save all waveforms, save all waveforms after applying filters, or discard all waveforms after running analysis).

Prerecorded waveforms.

Figure 2: Acquisitions tab: using prerecorded waveforms



When using prerecorded waveform files, this panel lists available prerecorded waveform files. You can only select the source of the prerecorded waveform file for each test. See [Set acquisition waveform source for prerecorded waveform files](#).

Set acquisition options. Select an **Acquire Option** to set the order in which waveforms are acquired and analyzed:

- **Acquire All Waveforms Before Analysis:** Acquire all waveforms required by tests before performing analysis. All required user interventions (such as connecting to different lanes) are completed, and waveforms acquired, before the analysis is run. You can turn off the DUT after the acquisitions are completed.
- **Acquire and Analyze Acquisition in Sequence:** Acquire waveforms and analyze for each test before proceeding to the next test. Use this setting to stop the testing when an error occurs, investigate and correct DUT, instrumentation connections, or application settings, then restart testing.
- **Acquire Only – Do Not Analyze:** Acquire all waveforms required by tests, and then stop (do not use waveforms to perform test analysis). Use this setting for testing multiple DUTs once the test and application settings are correct. Acquire all required waveforms and save the session for each DUT, and then recall the waveforms at a later point to analyze in *Prerecorded* mode.

See also.

[Set acquisitions signal source](#)

[Set acquisition waveform save options](#)

Set acquisition waveform save options. Select a **Save Option** to set how to save acquired test waveforms:

- **Save All the Waveforms:** Save all waveforms that were acquired for tests.
- **Save Only Analyzed Wfms:** Save waveforms that was used for analysis.
- **No Waveforms Saved – Discard after analysis:** Delete all acquired waveform data after analysis is complete.

Waveforms are saved to a folder that is unique to each session (a session starts when you click the Start button). The folder path is X:\PCI Express\Untitled Session\<<dutid>\<date>_<time>. Images created for each analysis, CSV files with result values, reports, and other information specific to that particular execution are also saved in this folder. When the session is saved, content is moved to that session folder and the “Untitled Session” gets replaced by the session name.

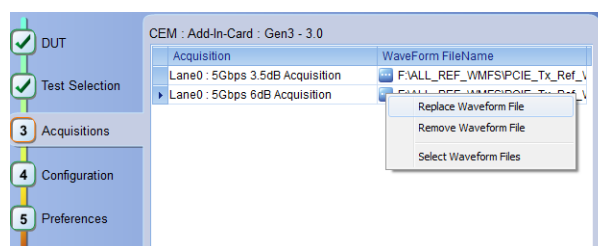
See also.

[Set acquisitions signal source](#)

[Set acquisition waveform source for prerecorded waveform files](#)

[Set acquisition Acquire options](#)

Set acquisition waveform source for prerecorded waveform files. When using prerecorded waveform files, there are no acquisition source selections to make. You can only select the source of the prerecorded waveform files for each test.



If you selected to use a prerecorded waveform file (in the DUT tab), the lane and source fields are not applicable and are not shown. The Acquisition tab instead shows a table of the waveforms used for the required test acquisitions.

You can load a different waveform file for each table item. To load a different waveform file:

1. Click the ellipsis button (⋮) of the waveform file to change.
2. Select the waveform task to perform (replace, remove, or select the waveform file).
3. Use the dialog box to navigate to and select the waveform file with which to replace the current file. You need to select all required differential waveforms for analysis. For example, select one data waveform and one clock waveform for each acquisition (except 2.5 Gbps) for testing a system board.

NOTE. Clock signals are not required for Gen1 (2.5 Gbps data rate) testing.

See also.

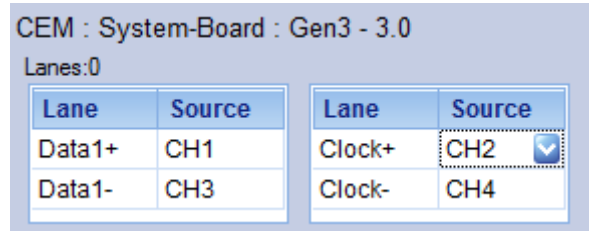
[Set acquisition signal source](#)

[Set acquisition options](#)

[Set acquisition waveform save options](#)

Set acquisition signal source. Use this procedure to set the channel sources for live waveform acquisitions. The number of Lane and Source fields shown depends on the number of lanes selected for testing in the **DUT** tab.

1. Click **Setup > Acquisitions**.
2. Click in the Source column of the field to change.
3. Click the arrow button to list available sources from which to select.



See also.

[Set acquisition options](#)

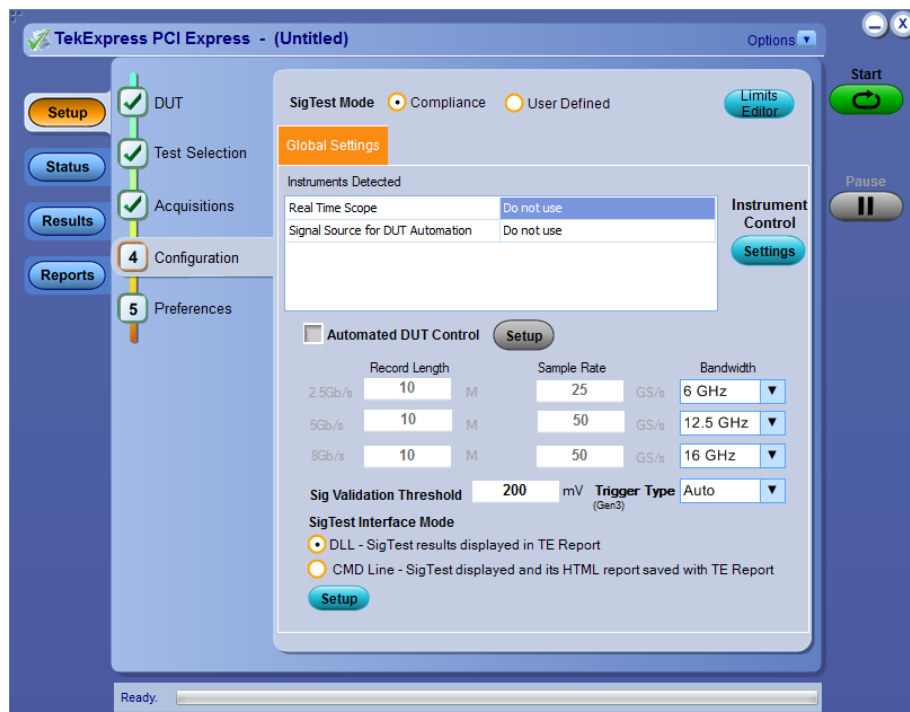
[Set acquisition waveform save options](#)

[Set acquisition waveform source for prerecorded waveform files](#)

Configure test parameters.

About configuring test parameters. Use the **Configuration** tab to view and set global and individual measurement parameters for the selected tests. Which fields are available to edit depends on the selected Sigtest mode (Compliance or User Defined) as set in this tab or the DUT tab.

NOTE. You cannot change test parameters that are grayed out.



See also.

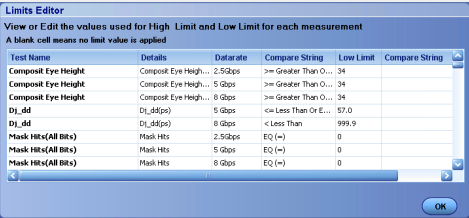
[Configuration tab parameters](#)

[About setting up tests](#)

[About running tests](#)

Configuration tab parameters. The following table lists the Configuration tab settings and parameters.

Table 7: Configuration tab parameters

| Parameter | Description |
|-----------------------|--|
| SigTest Mode | <p>Determines whether test parameters are in compliance or can be edited (User Defined Mode).</p> <ul style="list-style-type: none"> Compliance: Most test parameter values cannot be edited. User Defined: Enables editing of most test parameters. |
| Limits Editor | <p>Shows the upper and lower limits for the applicable measurement using different types of comparisons.</p> <p>In Compliance Mode, use the Limits Editor to view the measurement high and low limits used for selected tests.</p> <p>In User Defined Mode, use the Limits Editor to edit the limit settings.</p>  <p>To edit a value, click that field and either select from the displayed list or enter a new value. Use the bottom scroll bar to view all available fields.</p> |
| Instruments Detected | <p>Displays a list of the connected instruments found during the instrument discovery.</p> <p>Instrument types include equipment such as oscilloscopes and signal generators. Select Instrument Control Settings to <i>refresh the connected instrument list</i>.</p> |
| Automated DUT Control | <p>Enables automatic toggling of test patterns for DUT tests. Requires an AFG instrument as part of the test setup. Click Setup to configure the DUT automation settings.</p> |

| Parameter | Description |
|---------------------------------------|---|
| Record Length, Sample Rate, Bandwidth | <p>These settings apply to all tests selected for the indicated data rate.</p> <ul style="list-style-type: none"> ■ Record Length: Specifies the waveform record length. ■ Sample Rate: Specifies the oscilloscope sample rate to use for all tests. ■ Bandwidth: Specifies the oscilloscope bandwidth to use for all tests. |
| Sig Validation Threshold | Sets the threshold voltage to use for signal validation. |
| Trigger Type (Gen3) | <ul style="list-style-type: none"> ■ Edge ■ Width ■ Auto |
| SigTest Interface Mode | Sets whether to use a Dynamic Link Library (DLL) or command line interface (CMD) for running Sigtest. |
| | NOTE. <i>DLL is applicable for CEM specification only</i> |
| | Click Setup to open the SigTest Module Settings dialog box, where you can specify which revision of PCI-SIG Sigtest library to use for running tests. |

See also.[About acquisition](#)[De-embed using filter files](#)

Set test notification preferences. Use the Preferences tab to set the application action when a test measurement fails:

1. Click **Setup > Preferences**.
2. Select the measurement failure action:
 - Select **On Test Failure, pause the test and let me investigate** to pause the test when a failure occurs. Click the **Status** and **Results** buttons to explore the failure condition. To resume the test, click **Continue**.
 - Select **On Test Failure, stop and notify me of the failure** to stop the test and send an email when a test fails. Click **Email Settings** to verify that **Email Test Results when complete or on error** is selected, and to verify the address to which the email is sent.

Preferences tab parameters. Use the Preferences tab to set the application action when a test measurement fails, and how the application handles pop-up error, warning, and information messages during test sequences.

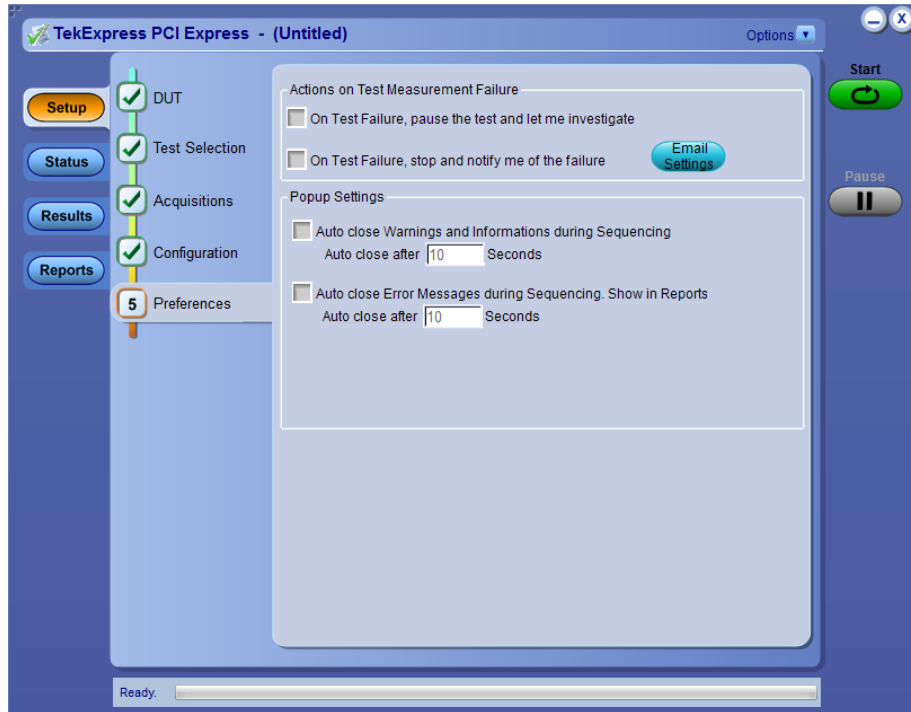


Table 8: Preferences tab parameters

| Parameter | Description |
|--|---|
| Actions on Test Measurement Failure | |
| On Test Failure, stop and notify me of the failure | Stops the test sequence and sends an email when a test fails. |
| On Test Failure, pause the test and let me investigate | Pauses the test when a failure occurs. Click the Status and Results buttons to explore the failure condition. To resume the test, click Continue . |
| Email Settings button | Click Email Settings to open the Email Settings dialog box and verify that Email Test Results when complete or on error is selected, and verify the address to which the email is sent. |
| Popup Settings | |
| Auto close Warnings and Informations during Sequencing | Sets the time for how long the application displays Warning and Information pop-up messages before automatically closing the messages and continues testing by taking the default action. |

| Parameter | Description |
|---|---|
| Auto close Error Messages during Sequencing. Show in Reports | Sets the time for how long the application displays Error pop-up messages before automatically closing the messages and continuing with testing. Message content is added to the test report. |

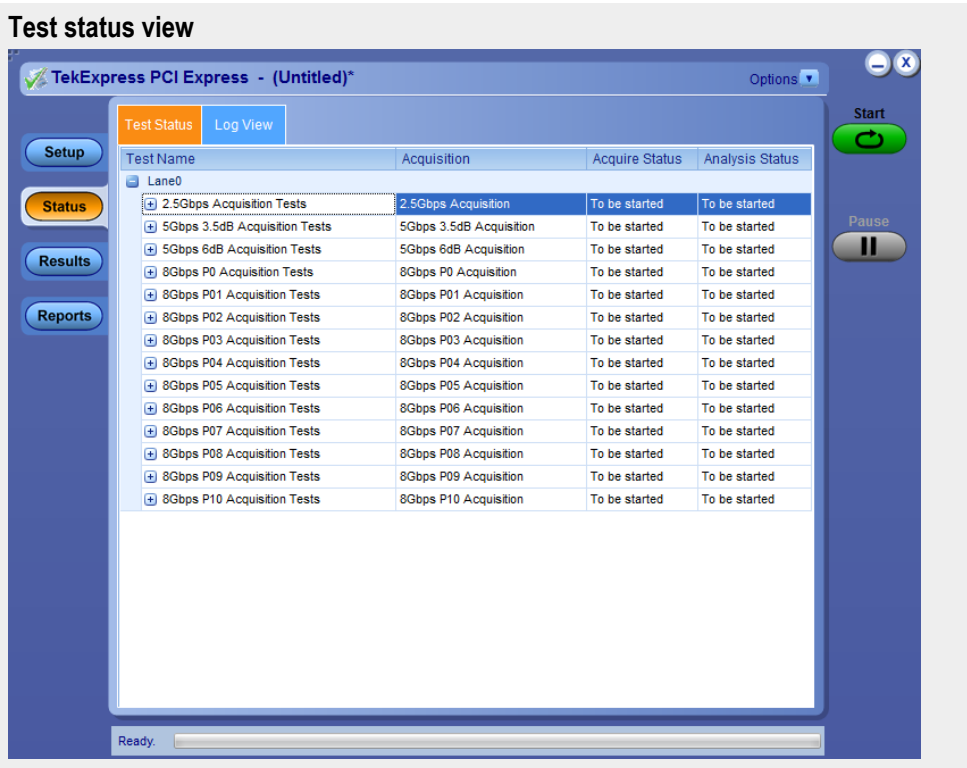
See also.

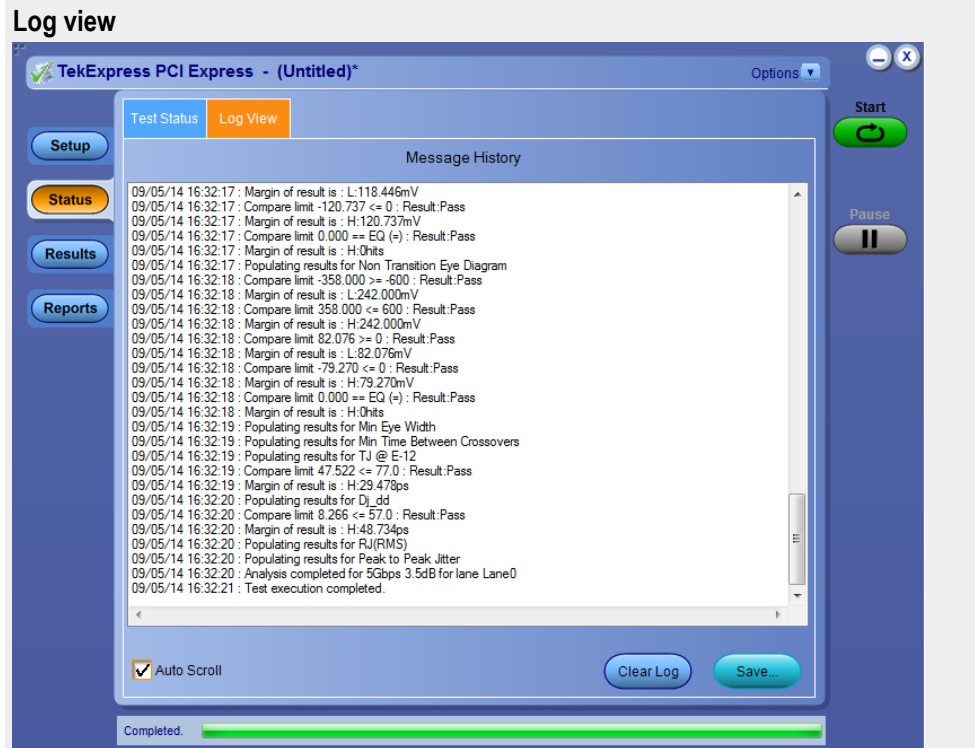
[About setting up tests](#)

[Select report options](#)

Status panel overview

The Status panel provides status on test acquisition and analysis (Test Status tab) and a listing of test tasks performed (Log View tab). The application opens the Test Status tab when you start a test run. You can select the Test Status or the Log View tab to view these items while tests are running.





The Log View display has several viewing options:

- **Message History**: This window timestamps and displays all run messages.
- **Auto Scroll** button : Select this check box to have the program automatically scroll down as information is added to the log during the test.
- **Clear Log** button : Click this button to clear all messages from the display.
- **Save** button : Click this button to save the log file as a text file. A standard Save File window is displayed to name and save the file.

See also.

[Application panel overview](#)

- To enable or disable the wordwrap feature, select **Preferences > Enable Wordwrap**.
- To expand the width of a column, place the cursor over the vertical line that separates the column from the column to the right. When the cursor changes to a double-ended arrow, hold down the mouse button and drag the column to the desired width.
- To group and view the tests by Lane, Test, Equalization, Pass/Fail, use the Preferences option in Results Panel.
- To clear all test results displayed, click **Clear**.

See also.

[View a report](#)

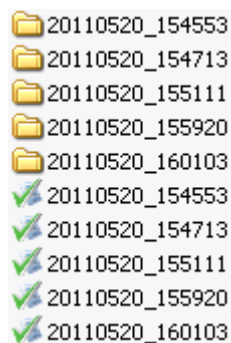
[About panels](#)

View test-related files. Files related to tests are stored in the My TekExpress\PCI Express folder. Each test setup in this folder has a test setup file and a test setup folder, both with the test setup name.

The test setup file is preceded by the TekExpress icon and usually has no visible file name extension.

Inside the test setup folder is another folder named for the DUT ID used in the test sessions. The default is DUT001.

Inside the DUT001 folder are the session folders and files. Each session also has a folder and file pair, both named for the test session using the naming convention (date)_(time). Each session file is stored outside its matching session folder:



Each session folder contains image files of any plots generated from running the test session. If you selected to save all waveforms or ran tests using prerecorded waveform files, these are included here.

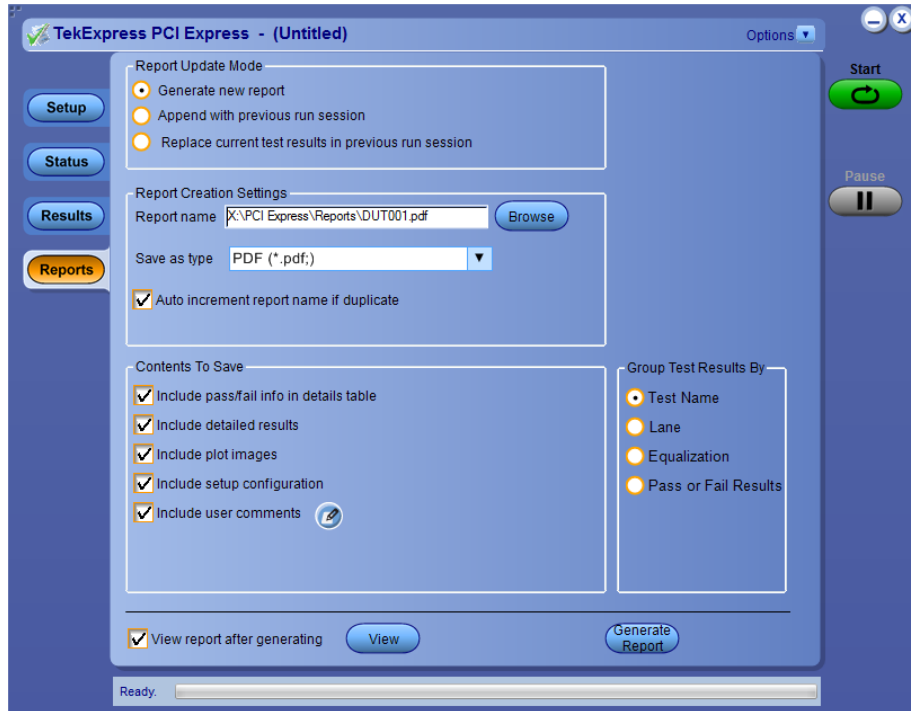
The first time you run a new, unsaved session, the session files are stored in the Untitled Session folder located at ..\My TekExpress\PCI Express. When you name and save the session, the files are placed in a folder with the name that you specify. A copy of the test files stay in the Untitled Session folder until you run a new test or until you close the PCIe application.

See also.

[File name extensions](#)

[Before you click start](#)

Reports panel **Reports panel overview.** Use the Reports panel to browse for reports, name and save reports, select test content to include in reports, and select report viewing options.



For information on setting up reports, see [Select report options](#). For information on viewing reports, see [View a Report](#).

See also.

[About panels](#)

Select report options. Click the **Reports** button and use the Reports panel controls to select which test result information to include in the report, and the naming conventions to use for the report. For example, always give the report a unique name or select to have the same name increment each time you run a particular test.

Select report options before running a test or when creating and saving test setups. Report settings are included in saved test setups.

In the Reports panel, select from the following report options:

Table 9: Report options

| Setting | Description |
|--|---|
| Report Update Mode | |
| Generate new report | Creates a new report. |
| Append with previous run session | Appends the latest test results to the end of the current test results report. |
| Replace current test in previous run session | Replaces the previous test results with the latest test results. Results from newly added tests are appended to the end of the report. |
| Report Creation Settings | |
| Report name | <p>Displays the name and location from which to open a report. The default location is at <i>My TekExpress\PCI Express\Reports</i>. The report file in this folder gets overwritten each time you run a test unless you specify a unique name or select to auto increment the report name, this option is selected by default.</p> <p>Change the report name or location.</p> <p>Do one of the following:</p> <ul style="list-style-type: none"> ■ In the Report Path field, type over the current folder path and name. ■ Double-click in the Report Path field and then make selections from the popup keyboard and click the Enter button. <p>Be sure to include the entire folder path, the file name, and the file extension. For example: C:\Documents and Settings\your user name\My Documents\My TekExpress\PCI Express\DUT001_Test_72.7.1.3.mht.</p> <p>Open an existing report.</p> <p>Click Browse button to locate and select the report file and then click View button at the bottom of the panel.</p> |

| Setting | Description |
|---|--|
| Save as type | <p>Saves a report in the specified file type. Lists supported file types to choose from.</p> <p>NOTE. <i>If you select a file type different from the default, be sure to change the report file name extension in the Report Name field to match. By default, report is generated in PDF format but you can also generate the report in MHT format.</i></p> |
| Auto increment report name if duplicate | <p>Sets the application to automatically increment the name of the report file if the application finds a file with the same name as the one being generated. For example: DUT001, DUT002, DUT003. This option is enabled by default.</p> <p>NOTE. <i>If you choose, say 'TID_345', for 'DUT ID', the next report for a new run in the same session will be named 'TID_346' which may contradict or cause confusion. In this case, it is suggested that you choose 'TID_345_001' for DUT ID, since the next runs in the same session will have report names generated with the last suffix auto-incremented as follows – 'TID_345_002' and so on.</i></p> |
| Contents To Save | |
| Include pass/fail info in details table | Select to include the column labeled Test Result (indicating whether the test passed or failed) in the report. For details, see Report Contents in View a report . |
| Include detailed results | Sets the application to include parameter limits, execution time, and test-specific comments generated during the test. |
| Include plot images | Sets the application to include plotted diagrams such as Eye diagrams. |
| Include setup configuration | Sets the application to include hardware and software information in the summary box at the top of the report. Information includes: the oscilloscope model and serial number, probe model and serial number, the oscilloscope firmware version, SPC and factory calibration status, and software versions for applications used in the measurements. |
| Include user comments | Select to include any comments about the test that you or another user added in the DUT tab of the Setup panel. Comments appear in the Comments section, under the summary box at the beginning of each report. |

| Setting | Description |
|-------------------------------|--|
| Group Test Results by | |
| Group Test Results by | Sets how to group the test results in the results pane and report. |
| Test Name | Select to display the test results by test name. |
| Lane | Select to display the test results by lane. |
| Equalization | Select to display the test results by equalization. |
| Pass or Fail Results | Select to display the test results by pass or fail results. |
| Other | |
| View Report After Generating | Automatically opens the report in a Web browser when the test completes. This option is selected by default. |
| View button | Click to view the most current report. |
| Generate Report button | Generates a new report based on the current analysis results. |

See also.

[View a report](#)

[About setting up tests](#)

View a report. The application automatically generates a report when test analysis is completed and displays the report in your default Web browser (unless the **View Report After Generating** check box is not selected). If this check box is not cleared, or to view a different test report, do the following:

1. Click the **Browse** button and locate and select the report file to view.
2. In the Reports panel, click **View**.

For information on changing the file type, file name, and other report options, see [Select report options](#).

Report contents. A report shows specified test details, as defined in the Reports panel.

NOTE. *NAN (Not A Number) is displayed in the report contents if an invalid waveform was supplied for the test. In PDF generated report, hyperlinks do not work and result table for some measurements maybe spread across a page.*

Setup configuration information

Setup configuration information is listed in the summary box at the beginning of the report. This information includes the oscilloscope model and serial number, and software versions. To exclude this information from a report, clear the **Include Setup Configuration** check box in the Reports panel before running the test.

| Tektronix TekExpress PCI Express Add-In-Card Test Report | | | |
|--|--|------------------------|------------------|
| Setup Information | | | |
| DUT ID | DUT001 | DPOJET Version | 6.2.1.8 |
| Date /Time | 2014-11-04 12:27:01 | Scope Model | MSO72504DX |
| Device Type | PCIe | Scope Serial Number | PQ00006 |
| TekExpress Version | PCI Express 2.2.0.91 Framework: 3.0.1.64 | SPC_FactoryCalibration | PASS/PASS |
| Test Mode | SigTest Compliance | Scope F/W Version | 7.2.0 devBuild 4 |
| Spec Version | Gen3 - 3.0 | Probe1 Model | TCA-SMA |
| SigTest Version | 3.2_0 | Probe2 Model | TCA292D |
| Slot Number | 01 | Probe2 Serial Number | None |
| Overall Execution Time | 0:00:15 | Probe3 Model | TCA292D |
| Overall Test Result | Pass | Probe3 Serial Number | N/A |
| | | Probe4 Model | TCA-SMA |
| DUT COMMENT: | DUT001 | | |
| Test Name Summary Table | | | |
| Unit Interval | | | |
| Mask Hits(All Bits) | | | |
| Composit Eye Height | | | |
| Number Passing Eyes | | | |
| Number Failing Eyes | | | |
| Transition Eye Diagram | | | |
| Non Transition Eye Diagram | | | |
| Min Eye Width | | | |
| Min Time Between Crossovers | | | |
| RMS Jitter (Per Edge) | | | |
| Median Peak Jitter | | | |
| Peak to Peak Jitter | | | |

Test result summary

The Test Result column indicates whether a test passed or failed. If the test passed, the column cell is green. If the test failed, it is red. To exclude this information from a report, clear the **Include pass/fail info in details table** check box in the Reports panel before running the test.

| Unit Interval | | | | | | | | |
|---------------------------------------|-----------|-----------|--------------|----------------|-------------|------------------------|-----------|------------|
| Measurement Details | Lane Name | Data Rate | Equalization | Measured Value | Test Result | Margin | Low Limit | High Limit |
| Mean Unit Interval | Lane0 | 2.5Gbps | - | 400.025 ps | Pass | L:0.145ps H:0.095ps | 399.88 | 400.12 |
| Min Unit Interval | Lane0 | 2.5Gbps | - | 400.025 ps | Pass | L:0.145ps | 399.88 | N.A |
| Max Unit Interval | Lane0 | 2.5Gbps | - | 400.025 ps | Pass | H:0.095ps | N.A | 400.12 |
| COMMENTS | | | | | | | | |
| Back to Summary Table | | | | | | | | |
| Mask Hits(All Bits) | | | | | | | | |
| Measurement Details | Lane Name | Data Rate | Equalization | Measured Value | Test Result | Margin | Low Limit | High Limit |
| Mask Hits | Lane0 | 2.5Gbps | - | 0 hits | Pass | N.A | 0 | 0 |
| COMMENTS | | | | | | | | |
| Back to Summary Table | | | | | | | | |
| Composit Eye Height | | | | | | | | |
| Measurement Details | Lane Name | Data Rate | Equalization | Measured Value | Test Result | Margin | Low Limit | High Limit |
| Composit Eye Height | Lane0 | 2.5Gbps | - | 590.653 mV | Informative | N.A | N.A | N.A |
| COMMENTS | | | | | | | | |
| Back to Summary Table | | | | | | | | |

See also.

[Results panel overview](#)

[View test-related files](#)

Setting up and configuring tests

About setting up tests

Set up tests using the tabs in the [Setup panel](#). Settings in the DUT tab use a top-down, left-to-right logic flow, so that any parameter that affects or acts as a filter for other parameters appears either to the top of or to the left of the affected parameters.

Tests are saved when you save a test setup. To avoid overwriting test results, remember to assign a unique name to the test either before running it or immediately after.

See also

[Test setup overview](#)

[Before you click start](#)

[About test setups](#)

[About running tests](#)

Equipment connection setup

Click the **Setup > Test Selection > Schematic** button to open a PDF file that shows the compliance test setup diagrams (instrument, DUT, and cabling) for supported testing configurations.

See also

[Minimum system requirements](#)

[View connected instruments](#)

[About setting up tests](#)

Running tests

Test setup overview

Test setup includes acquisition and configuration parameters. You can also select report options when setting up tests. Use the options in the *Setup panel* and *Reports panel* to select and configure tests.

1. *Set up equipment.*
2. *Do the prerun checklist.*
3. *Set DUT parameters.*
4. *Select one or more tests.*
5. *Select acquisitions.*
6. *Configuration test parameters.*
7. *Set test measurement notification options.*
8. *Select report options.*

See also

About test setups

Before you click start

About running tests

About running tests

After selecting and configuring tests, review the *Prerun checklist* and then click **Start** to run the tests. While tests are running, you cannot access the Setup or Reports panels. To monitor the test progress, switch back and forth between the Status panel and the Results panel.

The application displays a report when the tests are complete. While the tests are running, other applications may display windows in the background. The TekScope application takes precedence over other applications, but you can switch to other applications by using the **Alt + Tab** key combination. To keep the TekExpress PCIe application on top, select **Keep On Top** from the TekExpress Options menu.

See also [Before you click start](#)
[About configuring tests](#)
[About setting up tests](#)

Prerun checklist

Do the following before you click Start to run a test. If this is the first time you are running a test on a setup, refer to the information in [Before you click start](#).

1. Make sure that all the required instruments are properly warmed up (approximately 20 minutes).
2. Perform Signal Path Compensation (SPC):
 - a. On the oscilloscope main menu, select the **Utilities** menu.
 - b. Select **Instrument Calibration**.
3. Verify that the application is able to find the DUT. If it cannot, perform a search for connected instruments:
 - a. In PCIe, select the **Setup** panel and then click the **Test Selection** tab.
 - b. Select any test and then click **Configure**.
 - c. In the Configuration section, click **Global Settings**.
 - d. In the **Instruments Detected** section, click the drop-down arrow to the right of **Real Time Scope** and make sure that the oscilloscope with the (GPIB8::1::INSTR) designation is in the list.

See also [Equipment connection setup](#)

Saving and recalling test setups

About test setups

TekExpress PCIe opens with the default setup selected. Run a test before or after saving a setup. When you save a setup, the test information, such as the selected oscilloscope, general parameters, acquisition parameters, measurement limits, waveforms (if applicable), and other configuration settings are all saved under the setup name at **X:\PCI Express**.

Use test setups to:

- Run a saved test in prerecorded mode.
- View all the information associated with a saved test, including the log file, the history of the test status as it executed, and the results summary.
- Create a new test setup based on an existing one.
- Run a new session, acquiring live waveforms, using a saved test configuration.

See also

[*About setting up tests*](#)

[*Save a test setup*](#)

[*Recall a saved test setup*](#)

Save a test setup

Save a test setup before or after running a test to save the test configuration. Create a new test setup from any open setup or from the default setup. When you select the default test setup, all parameters are returned to the application's default values.

To save the current setup session to the same setup name, select **Options > Save Test Setup**.

To save the current setup session to a new setup name, select **Options > Save Test Setup As**.

To create and save a new setup from the default test setup:

1. Select **Options > Default Test Setup**.
2. Select **Setup** and set required options and parameters in the tabs (DUT, Test Selection, and so on).
3. Select **Reports** and set your *report options*.
4. Optional: Click **Start** to run the test and verify that it runs correctly and captures the information you want. If it does not, edit the parameters and repeat this step until the test runs to your satisfaction.

Running the test helps verify that all parameters are set correctly, but it is not a necessary step.

5. Select **Options > Save Test Setup**. Enter the file name for the setup file. The application saves the file to X:\PCI Express*<session_name>*.

See also

[About setting up tests](#)

[Test setup overview](#)

[View test-related files](#)

[About configuring tests](#)

Open (load) a saved test setup

These instructions are for recalling saved test setups.

1. Select **Options > Open Test Setup**.
2. Select the setup from the list and click **Open**. Setup files must be located at **X:\PCI Express**.

See also

[About test setups](#)

[Create a new test setup based on an existing one](#)

[Test setups overview](#)

Create a new test setup based on an existing one

Use this method to create a variation on a test setup without having to create the setup from the beginning.

1. Select **Options > Open Test Setup**.
2. Select a setup from the list and then click **Open**.
3. Use the **Setup** and **Reports** panels to modify the parameters to meet your testing requirements.
4. Select **Options > Save Test Setup As**.
5. Enter a test setup name and click **Save**.

NOTE.

- Select **Default Test Setup** before execution of every test and save the session with a valid name; this will prevent overwriting/losing data.
 - When you save session using the option **Save Test Setup**, the "Untitled Session" folder and the "Untitled Session.Tekx" file will get renamed with the session name provided by you. It takes few seconds to store the data in the folder.
 - When you save session using the option **Save Test Setup As**, the "Untitled Session" folder and the "Untitled Session.Tekx" file will get named with session name provided by you. It takes few seconds to store the data in the folder. The same is applicable when you use the **Save as** option to save the already saved session files.
-

See also

[About test setups](#)

[Set DUT parameters](#)

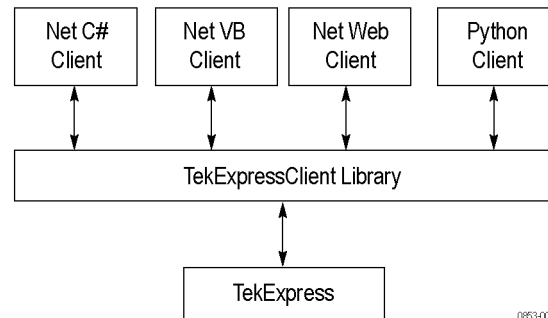
[Configuration parameters](#)

[Select acquisitions](#)

TekExpress programmatic interface

About the programmatic interface

The Programmatic interface seamlessly integrates the TekExpress test automation application with the high-level automation layer. This also lets you control the state of the TekExpress application running on a local or a remote computer.



The following terminology is used in this section to simplify description text:

- **TekExpress Client:** A high-level automation application that communicates with TekExpress using TekExpress Programmatic Interface.
- **TekExpress Server:** The TekExpress application when being controlled by TekExpress Client.

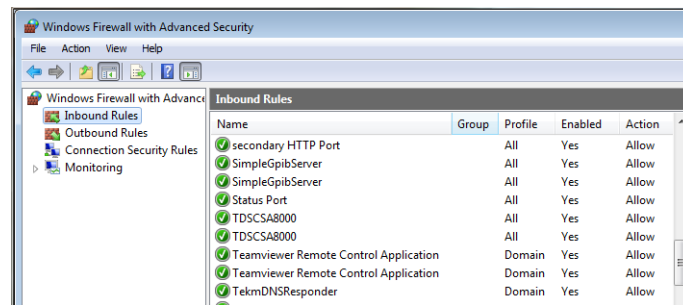
TekExpress leverages .Net Marshalling to enable the Programmatic Interface for TekExpress Client. TekExpress provides a client library for TekExpress clients to use the programmatic interface. The TekExpress client library maintains a reference to the TekExpress Server and this reference allows the client to control the server state.

See also [Requirements for developing TekExpress client](#)

To enable remote access

To access and remotely control an instrument using the TekExpress programmatic interface, you need to change specific firewall settings as follows:

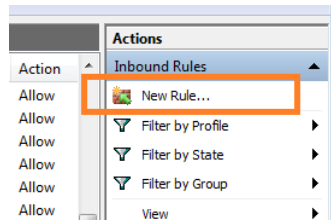
1. Access the Windows Control Panel and open the Windows Firewall tool (**Start > Control Panel > All Control Panel Items > Windows Firewall**).
2. Click **Advance Settings > Inbound Rules**.
3. Scroll through the **Inbound Rules** list to see if the following items (or with a similar name) are shown:
 - TekExpress PCI Express
 - TekExpress



4. If both items are shown, you do not need to set up any rules. Exit the Windows Firewall tool.
5. If one or both are missing, use the following procedure to run the **New Inbound Rule Wizard** and add these executables to the rules to enable remote access to the TekExpress application.
6. On the client side, include controller. exe through which TexExpress PCIe application is remotely controlled. For example, if the application is controlled using python scripts the "ipy64.exe" should be included as part of Inbound rules.

Run the New Inbound Rule Wizard

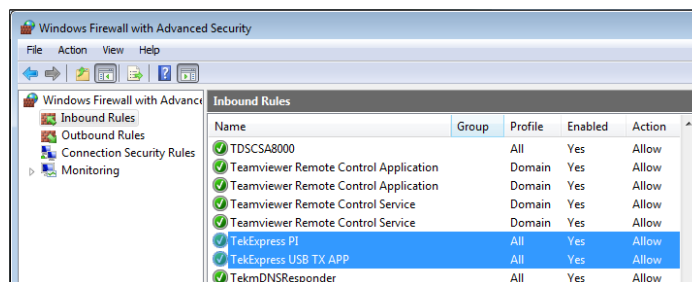
1. Click **New Rule** (in Actions column) to start the **New Inbound Rule Wizard**.



2. Verify that **Program** is selected in the Rule Type panel and click **Next**.
3. Click **Browse** in the Program panel and navigate to and select one of the following TekExpress applications (depending on the one for which you need to create a rule):
4. TekExpress PCI Express.exe
5. TekExpress.exe

NOTE. See [Application directories and content](#) for the path to the application files.

6. Click **Next**.
7. Verify that **Allow the connection** is selected in the Action panel and click **Next**.
8. Verify that all fields are selected (**Domain, Private, and Public**) in the Profile panel and click **Next**.
9. Use the fields in the Name panel to enter a name and optional description for the rule. For example; **TekExpress PCI Express Application**. Add description text to further identify the rule.
10. Click **Finish** to return to the main Windows Firewall screen.
11. Scroll through the Inbound Rules list and verify that the list shows the rule that you just entered.



12. Repeat steps 1 through 11 to enter the other TekExpress executable if it is missing from the list. Enter **TekExpress PI** as the name.
13. Scroll through the Inbound Rules list and verify that the list shows the rule that you just entered.
14. Exit the Windows Firewall tool.

- To use the remote access:**
1. Obtain the IP address of the instrument on which you are running TekExpress PCI Express. For example, 134.64.235.198.
 2. On the PC from which you are accessing the remote instrument, use the instrument IP address as part of the TekExpress PCI Express PI code to access that instrument. For example:

```
object obj = piClient.Connect("134.64.235.198",out clientid);
```

Requirements for developing TekExpress client

While developing TekExpress Client, use the TekExpressClient.dll. The client can be a VB .Net, C# .Net, TestStand, Python, or Web application. The examples for interfaces are in the TekExpress PCI Express\Examples folder.

NOTE. *The TestStand run time engine is no longer installed as of this release of TekExpress PCI Express. You can continue to use TestStand scripts if you install the TestStand run time engine and set up your environment to use it.*

References required

- TekExpressClient.dll has an internal reference to IIdlglib.dll and IRemoteInterface.dll.
- IIdlglib.dll has a reference to TekDotNetLib.dll.
- IRemoteInterface.dll provides the interfaces required to perform the remote automations. It is an interface that forms the communication line between the server and the client.
- IIdlglib.dll provides the methods to generate and direct the secondary dialog messages at the client-end.

NOTE. *The end-user client application does not need any reference to the above mentioned DLL files. It is essential to have these DLLs (IRemoteInterface.dll, IIdlglib.dll and TekDotNetLib.dll) in the same folder as that of TekExpressClient.dll.*

Required steps for a client

The client uses the following steps to use TekExpressClient.dll to programmatically control the server:

Develop a client UI to access the interfaces exposed through the server. This client loads TekExpressClient.dll to access the interfaces. After TekExpressClient.dll is loaded, the client UI can call the specific functions to run the operations requested by the client. When the client is up and running, it does the following to run a remote operation:

1. To connect to the server, the client provides the IP address of the PC where the server is running.
2. The client locks the server application to avoid conflict with any other Client that may try to control the server simultaneously. "Lock" would also disable

all user controls on the server so that server state cannot be changed by manual operation.

If any other client tries to access a server that is locked, it will receive a notification that the server is locked by another client.

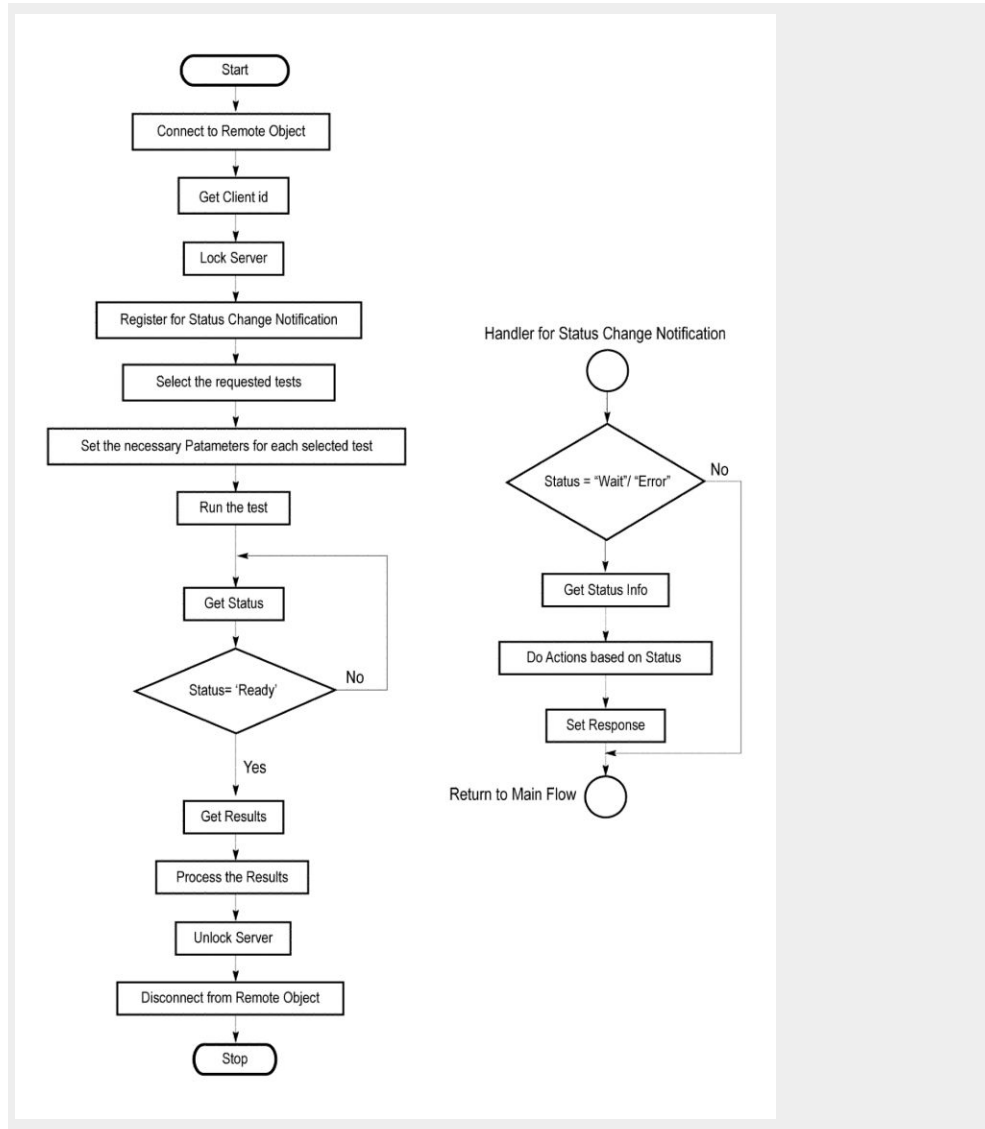
3. When the client has connected to and locked the server, the client can access any of the programmatic controls needed to run the remote automations.
4. After the client operations finish, the client unlocks the server.

See also [PCIe application commands flow](#)

Client programmatic interface example

An example of the client programmatic interface is described and shown as follows:

Process flowchart



1. Connect to a server or remote object using the programmatic interface provided.
2. Get the client ID that is created when connecting to the remote object. This client ID is one of the required parameters to communicate with the server.

NOTE. The server identifies the client with this ID only and rejects any request if the ID is invalid.

3. Lock the server for further operations. This disables the application interface.

NOTE. You can get values from the server or set values from the server to the client only if the application is locked.

4. Register for receiving notifications on status change events on the server. To register you need to give a handler as a parameter.

NOTE. *Whenever there is a change in the status of the server, all the clients registered with the server receive a notification from the server.*

5. Select the tests that you want to run through the programmatic interface.
6. Set the necessary parameters for each test.
7. Run the tests.
8. Poll for the status of the application.

NOTE. *Skip step 8 if you are registered for the status change notification and the status is Ready.*

9. After completing the tests, get the results.
10. Create a report or display the results and verify or process the results.
11. Unlock the server after you complete all the tasks.
12. Disconnect from the remote object.

Handler of status change notification

1. Get the status. If the status is Wait or Error, get the information that contains the title, message description, and the expected responses for the status.
2. Perform the actions based on the status information.
3. Set the response as expected.

See also

[PCIe application commands flow](#)

[Program remote access code example](#)

Program remote access code example

This code example shows how to communicate between a remote PC and TekExpress PCI Express.

Table 10: Remote access code example

| Task | Code |
|-----------------------------------|--|
| Start the application | |
| Connect through an IP address. | <code>m_Client.Connect("localhost") 'True or False clientID = m_Client.getClientID</code> |
| Lock the server | <code>m_Client.LockServer(clientID)</code> |
| Disable the Popups | <code>m_Client.SetVerboseMode(clientID, false)</code> |
| Set the DUT ID | <code>m_Client.SetDutId(clientID, "DUT_Name")</code> |
| Run with set configurations | <code>m_Client.Run(clientID)</code> |
| Wait for the test to complete. | <code>Do Thread.Sleep(500) m_Client.Application_Status(clientID) Select Case status Case "Wait"</code> |
| Get the current state information | <code>mClient.GetCurrentStateInfo(clientID, WaitingMsbBxCaption, WaitingMsbBxMessage, WaitingMsbBxButtontexts)</code> |
| Send the response | <code>mClient.SendResponse(clientID, WaitingMsbBxCaption, WaitingMsbBxMessage, WaitingMsbBxResponse) End Select Loop Until status = "Ready"</code> |
| Save results | <code>Save all results values from folder for current run m_Client.TransferResult(clientID, logDirname)</code> |
| Unlock the server | <code>m_Client.UnlockServer(clientID)</code> |
| Disconnect from server | <code>m_Client.Disconnect()</code> |
| Exit the application | |

Python and C# examples

Example Python and C# test sequence files are located at TekExpress PCI Express\Examples\.

PCIe application commands

PCIe application commands listing

Click a client action link to see the associated command name, description, parameters, return value, and an example.

[Connect through an IP address](#)

[Lock the server](#)

[Disable the popups](#)

[Set or get the DUT ID](#)

[Select the PCIe device](#)

[Select the suite](#)

[Set the PCIe test version](#)

[Set the test mode parameter](#)

[Set the prerecorded waveform execution mode](#)

[Set the data rate parameter](#)

[Set the 5 Gbs preemphasis parameter](#)

[Set the SSC parameter](#)

[Set the voltage swing parameter](#)

[Set the signal quality preset parameter](#)

[Set the preset lanes parameter](#)

[Set the lane source parameter](#)

[Set the preset parameter](#)

[Set the acquisition parameter](#)

[Set the analysis mode parameter](#)

[Set the Sigtest version parameter](#)

[Set the on failure action parameter](#)

[Set the report update mode parameter](#)

[Set the append report parameter](#)

[Set the crosstalk parameter](#)

[Set the waveform save parameter](#)

- Set the link analysis embed/de embed signal parameter*
- Set the link analysis embed/de embed filter file parameter*
- Set the link analysis other filter file parameter*
- Set the link analysis equalization dropdown parameter*
- Set the link analysis CTLE index paramete*
- Set the link analysis DFE parameter*
- Set the DUT auto toggle parameter*
- Set the DUT auto toggle options parameter*
- Set the AFG signal type parameter*
- Set the AFG signal frequency parameter*
- Set the AFG signal amplitude parameter*
- Set the burst count parameter*
- Set the record length parameter*
- Set the sample rate parameter*
- Set the bandwidth parameter*
- Set the signal validation parameter*
- Set the slot number parameter*
- Set SigTest Interface mode*
- Set Sig Validation Threshold*
- Set Toggle from Gen3P10 to Gen1*
- Run with set configurations or stop the run operation*
- Handle error codes*
- Get or set the timeout value*
- Wait for the test to complete*
- After the test is complete*
- Save, recall, or query a saved session*
- Unlock the server*
- Disconnect from the server*

| string id | | | |
|------------------|-------------|------------------|---|
| Name | Type | Direction | Description |
| id | string | IN | Identifier of the client performing the remote function |

Ready: Test configured and ready to start

Running: Test running

Paused: Test paused

Wait: A popup that needs your inputs

Error: An error is occurred

string dutName

| Name | Type | Direction | Description |
|---------|--------|-----------|-----------------------------|
| dutName | string | IN | The new DUT ID of the setup |

out bool saved

| Name | Type | Direction | Description |
|-------|------|-----------|---|
| saved | bool | OUT | Boolean representing whether the current session is saved |

This parameter is used as a check in SaveSession() and SaveSessionAs() functions.

string ipAddress

| Name | Type | Direction | Description |
|-----------|--------|-----------|--|
| ipAddress | string | IN | The ip address of the server to which the client is trying to connect. This is required to establish the connection between the server and the client. |

out string clientID

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientid | string | OUT | Identifier of the client that is connected to the server clientid = unique number + ipaddress of the client. For example, 1065-192.157.98.70 |

NOTE. If the *dutName* parameter is null, the client is prompted to provide a valid DUT ID.

NOTE. The server must be active and running for the client to connect to the server. You can connect any number of clients to the server at a time.

NOTE. When the client is disconnected, it is unlocked from the server and then disconnected. The id is reused.

string dutId

| Name | Type | Direction | Description |
|-------|--------|-----------|---------------------------------------|
| dutId | string | OUT | The DUT ID of the setup. for example, |

The dutId parameter is set after the server processes the request.

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|---|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "PCIe". |

string suite

| Name | Type | Direction | Description |
|-------|--------|-----------|---------------------------------|
| suite | string | IN | Specifies the name of the suite |

string test

| Name | Type | Direction | Description |
|------|--------|-----------|---|
| test | string | IN | Specifies the name of the test to obtain the pass or fail status or a test result value. Append spaces at the end of the test name to differentiate the PCIe DUT generation version for which to return measurements: Gen1: no spaces Gen2: One space Gen3: Two spaces Examples: Gen1: "Unit Interval" Gen2: "Unit Interval " Gen3: "Unit Interval " |

string parameterString

| Name | Type | Direction | Description |
|-----------------|--------|-----------|-----------------------------|
| parameterString | string | IN | Selects or deselects a test |

int rowNr

| Name | Type | Direction | Description |
|-------|------|-----------|--|
| rowNr | int | IN | Specifies the zero based row index of the sub-measurement for obtaining the result value |

NOTE. When the client tries to lock a server that is locked by another client, the client gets a notification that the server is already locked and it must wait until the server is unlocked. If the client locks the server and is idle for a certain amount of time then the server is unlocked automatically from that client.

out string[] status

| Name | Type | Direction | Description |
|--------|--------------|-----------|--|
| status | string array | OUT | The list of status messages generated during the run |

string name

| Name | Type | Direction | Description |
|------|--------|-----------|--|
| name | string | IN | The name of the session being recalled |

The name parameter cannot be empty. If it is empty, the client is prompted to provide a valid name.

NOTE. When the run is performed, the status of the run is updated periodically using a timer.

string name

| Name | Type | Direction | Description |
|------|--------|-----------|-------------------------------------|
| name | string | IN | The name of the session being saved |

The name parameter cannot be empty. If it is empty, the client is prompted to provide a valid name.

Once the session is saved under ‘name’ you cannot use this method to save the session in a different name. Use SaveSessionAs instead.

string name

| Name | Type | Direction | Description |
|------|--------|-----------|--|
| name | string | IN | The name of the session being recalled |

The same session is saved under different names using this method. The name parameter cannot be empty. If it is empty, the client is prompted to provide a valid name.

bool isSelected

| Name | Type | Direction | Description |
|------------|------|-----------|-----------------------------|
| isSelected | bool | IN | Selects or deselects a test |

string time

| Name | Type | Direction | Description |
|------|--------|-----------|---|
| time | string | IN | The time in seconds that refers to the timeout period |

The time parameter gives the timeout period, which is the time the client is allowed to be locked and idle. After the timeout period if the client is still idle, it gets unlocked.

The time parameter should be a positive integer; otherwise, the client is prompted to provide a valid timeout period.

bool_verbose

| Name | Type | Direction | Description |
|----------|------|-----------|---|
| _verbose | bool | IN | Specifies whether the verbose mode should be turned ON or OFF |

NOTE. When the session is stopped, the client is prompted to stop the session and is stopped at the consent.

string filePath

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| filePath | string | IN | The location where the report must be saved in the client |

NOTE. If the client does not provide the location to save the report, the report is saved at C:\ProgramFiles.

NOTE. When the client is disconnected, the client is unlocked automatically.

out string WaitingMsbBxCaption

| Name | Type | Direction | Description |
|---------|--------|-----------|---|
| caption | string | OUT | The wait state or error state message sent to you |

out string WaitingMsbBxMessage

| Name | Type | Direction | Description |
|---------|--------|-----------|--|
| message | string | OUT | The wait state/error state message sent to you |

out string[] WaitingMsbBxButtontexts

| Name | Type | Direction | Description |
|-------------|--------------|-----------|--|
| buttonTexts | string array | OUT | An array of strings containing the possible response types that you can send |

string WaitingMsbBxResponse

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| response | string | IN | A string containing the response type that you can select (it must be one of the strings in the string array buttonTexts) |

out string clientID

| Name | Type | Direction | Description |
|----------|--------|-----------|--|
| clientID | string | OUT | Identifier of the client that is connected to the server clientID = unique number + IP address of the client. For example, 1065-192.157.98.70 |

Connect through an IP address

| Command name | Parameters | Description | Return value | Example |
|--------------|---|--|--------------------------------------|--|
| Connect() | string ipAddress out string clientID | <p>This method connects the client to the server. Note</p> <p>NOTE. The server must be active and running for the client to connect to the server. You can connect any number of clients to the server at a time.</p> <p>The client provides the IP address to connect to the server. The server provides a unique client identification number when connected to it.</p> | Return value is either True or False | <pre>m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as boolean returnval = m_Client.Connect(ipaddress,m_client ID)</pre> |

string ipAddress

| Name | Type | Direction | Description |
|-----------|--------|-----------|--|
| ipAddress | string | IN | The ip address of the server to which the client is trying to connect. This is required to establish the connection between the server and the client. |

out string clientID

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientid | string | OUT | Identifier of the client that is connected to the server clientid = unique number + ipAddress of the client. For example, 1065-192.157.98.70 |

NOTE. The Fail condition for PI commands occurs in any of the following cases:

If the server is locked, the application displays "Server is locked by another client."

If the session is unlocked, the application displays "Lock session to execute the command."

If the server is not found, the application displays "Server not found-Disconnect!."

If the fail condition is not one of the above types, the application displays "Failed."

Lock the server

| Command name | Parameters | Description | Return value | Example |
|---------------|-----------------|---|--|---|
| LockSession() | string clientID | <p>This method locks the server. Note</p> <p>NOTE. <i>When the client tries to lock a server that is locked by another client, the client gets a notification that the server is already locked and it must wait until the server is unlocked. If the client locks the server and is idle for a certain amount of time then the server is unlocked automatically from that client</i></p> <p>The client must call this method before running any of the remote automations. The server can be locked by only one client.</p> | <p>String value that gives the status of the operation after it was performed</p> <p>The return value is "Session Locked..." on success.</p> | <pre>m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string returnval = m_Client.LockServer(clientID)</pre> |

out string clientID

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientid | string | OUT | Identifier of the client that is connected to the server clientid = unique number + ipaddress of the client. For example, 1065-192.157.98.70 |

NOTE. *The Fail condition for PI commands occurs in any of the following cases:*

If the server is locked, the application displays "Server is locked by another client."

If the session is unlocked, the application displays "Lock session to execute the command."

If the server is not found, the application displays "Server not found-Disconnect!."

If the fail condition is not one of the above types, the application displays "Failed."

Disable the popups

Use these commands to disable popup messages that require user intervention.

| Command name | Parameters | Description | Return value | Example |
|------------------|----------------------------------|--|--|--|
| SetVerboseMode() | string clientID bool _verbose | This method sets the verbose mode to either true or false. When the value is set to true, any message boxes that appear during the application are routed to the client machine that is controlling TekExpress. When the value is set to false, all the message boxes are shown on the server machine. | String that gives the status of the operation after it was performed When Verbose mode is set to true, the return value is "Verbose mode turned on. All dialog boxes will be shown to client". When Verbose mode is set to false, the return value is "Verbose mode turned off. All dialog boxes will be shown to server". | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Verbose mode is turned on return=m_Client.SetVerboseMode(clientID, true) Verbose mode is turned off returnval=m_Client.SetVerboseMode(clientID, false) |

out string clientID

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientid | string | OUT | Identifier of the client that is connected to the server clientid = unique number + ipaddress of the client. For example, 1065-192.157.98.70 |

bool_verbose

| Name | Type | Direction | Description |
|----------|------|-----------|---|
| _verbose | bool | IN | Specifies whether the verbose mode should be turned ON or OFF |

NOTE. *The Fail condition for PI commands occurs in any of the following cases:*

If the server is locked, the application displays "Server is locked by another client."

If the session is unlocked, the application displays "Lock session to execute the command."

If the server is not found, the application displays "Server not found-Disconnect!."

If the fail condition is not one of the above types, the application displays "Failed."

Set or get the DUT ID

| Command name | Parameters | Description | Return value | Example |
|--------------|-----------------------------------|--|---|---|
| SetDutId() | string clientID string dutName | This method changes the DUT ID of the setup. The client must provide a valid DUT ID. | String that gives the status of the operation after it was performed Return value is "DUT Id Changed" on success | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string return=m_Client.SetDutId(clientID,desiredDutId) Note |
| GetDutId() | string clientID string dutId | This method gets the DUT ID of the current setup. | String that gives the status of the operation after it was performed | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string return=m_Client.GetDutId(clientID,out DutId) |

out string clientID

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientid | string | OUT | Identifier of the client that is connected to the server clientId = unique number + ipaddress of the client. For example, 1065-192.157.98.70 |

string dutName

| Name | Type | Direction | Description |
|---------|--------|-----------|-----------------------------|
| dutName | string | IN | The new DUT ID of the setup |

string dutId

| Name | Type | Direction | Description |
|-------|--------|-----------|---------------------------------------|
| dutid | string | OUT | The DUT ID of the setup. for example, |

The dutId parameter is set after the server processes the request.

NOTE. If the dutName parameter is null, the client is prompted to provide a valid DUT ID.

NOTE. The Fail condition for PI commands occurs in any of the following cases:

If the server is locked, the application displays "Server is locked by another client."

If the session is unlocked, the application displays "Lock session to execute the command."

If the server is not found, the application displays "Server not found-Disconnect!."

If the fail condition is not one of the above types, the application displays "Failed."

Select the PCIe device

Syntax: mClient.SelectDevice(clientId, device, true);

| Command name | Parameters | Description | Return value | Example |
|--------------|--------------------|--------------------------|--|---|
| SelectDevice | clientId device | Selects the PCIe device. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Select PCIe device example |

out string clientId

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientId | string | OUT | Identifier of the client that is connected to the server clientId = unique number + ipaddress of the client. For example, 1065-192.157.98.70 |

Select PCIe device example

mClient.SelectDevice("1065-192.157.98.70", "CEM", true);

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|---|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2(SFF-8639)". |

Select the suite **Syntax:** `mClient.SelectSuite(clientId, device, devicesuite, true);`

| Command name | Parameters | Description | Return value | Example |
|--------------|--|---------------------------|--|--|
| SelectSuite | clientId device devicesuite | Sets the suite parameter. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | <code>m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Select suite example</code> |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|--|
| devicesuite | string | IN | Specifies the name of the device suite, enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2(SFF-8639), valid values are Module and Host . |

Select suite example

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "Add-In-Card", true);
```

out string clientId

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientId | string | OUT | Identifier of the client that is connected to the server clientId = unique number + ipaddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|---|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2(SFF-8639)". |

Set the PCIe test version **Syntax:** `mClient.SelectVersion(clientId, device, devicesuite, testversion);`

| Command name | Parameters | Description | Return value | Example |
|---------------|--|--------------------------------------|--|--|
| SelectVersion | clientId device testversion | Sets the PCIe compliance test suite. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Select test version example |

string testversion

| Name | Type | Direction | Description |
|-------------|--------|-----------|---|
| testversion | string | IN | Specifies the version of the PCIe specification test suite. Valid values are Gen1-1.0a , Gen1-1.1 , Gen2-2.0 , and Gen3-3.0 . Declare each value as a single array variable |

out string clientId

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientId | string | OUT | Identifier of the client that is connected to the server clientId = unique number + ipaddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|--|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2 (SFF-8639)". |

Set test version example

```
mClient.SelectVersions("1065-192.157.98.70", "CEM", "Add-In-Card", Gen2-2.0);
```

Set the data rate parameter

Syntax: `mClient.SetGeneralParameter(clientId, device, devicesuite, "", parameterString);`

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|-------------------------------|--|---|
| SetGeneralParameter | clientId device devicesuite parameterString | Sets the data rate parameter. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | <code>m_Client = new Client() //m_Client</code> is a reference to the Client class in the Client DLL. returnval as string Set data rate example |

out string clientId

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientid | string | OUT | Identifier of the client that is connected to the server clientId = unique number + ipaddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|--|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2 (SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|--|
| devicesuite | string | IN | Specifies the name of the device suite enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2 (SFF-8639), valid values are Module and Host . |

string parameterString (for test rate)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|--|
| parameterString | string | IN | A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter is the data rate. Valid values are DataRate2Gb , DataRate5Gb , and DataRate8Gb . The second parameter sets whether to include or exclude the data rate. Valid values are Included and Excluded . String example: "DataRate2Gb \$Included". |

Set data rate example

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board", "", "DataRate2Gb$Included");
```

Set the test mode parameter

Syntax: mClient.SetGeneralParameter(clientId, device, devicesuite, "", **parameterString**);

NOTE. This parameter has to be set before setting the link analysis, sample rate, bandwidth, and record length parameters.

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|---|--|--|
| SetGeneralParameter | clientId device devicesuite parameterString | Sets the Sigtest test mode parameter (compliance or use defined). | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Set test mode example |

out string clientID

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientid | string | OUT | Identifier of the client that is connected to the server clientid = unique number + ipaddress of the client. For example, 1065-192.157.98.70 |

string parameterString (for test mode)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|--|
| parameterString | string | IN | A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter is the data rate. Valid values are TestMode . The second parameter sets the test mode. Valid values are SigTest User Defined and SigTest Compliance . String example: "TestMode\$SigTest Compliance". |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|--|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2 (SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|--|
| devicesuite | string | IN | Specifies the name of the device suit, enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2 (SFF-8639), valid values are Module and Host . |

Set test mode example

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board", "", "TestMode$SigTest Compliance");
```

Set the prerecorded waveform execution mode

Syntax: mClient.SetPreRecorded(clientId, false | true, out error);

| Command name | Parameters | Description | Return value | Example |
|----------------|-----------------------------------|--|--|---|
| SetPreRecorded | clientId device devicesuite | Enables or disables using prerecorded (saved) waveforms for testing. <i>NOTE. There is no command in the PI to specify the prerecorded waveforms. Use this command to run a test session where you have already specified the waveforms files with the application user interface.</i> | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Set PreRecorded mode example |

out string clientID

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientid | string | OUT | Identifier of the client that is connected to the server clientid = unique number + ipaddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|--|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2 (SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|---|
| devicesuite | string | IN | Specifies the name of the device suite, enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2 (SFF-8639), valid values are Module and Host . |

Set SetPreRecorded mode example

```
mClient.SetPreRecorded(clientId, false, out error);
```

Set the 5 Gb/s preemphasis parameter

Syntax: mClient.SetGeneralParameter(clientId, device, devicesuite, "", parameterString);

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|--|--|--|
| SetGeneralParameter | clientId device devicesuite parameterString | Sets the 5 Gb/s preemphasis parameter. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Select 5 Gb/s preemphasis example |

out string clientId

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientId | string | OUT | Identifier of the client that is connected to the server clientId = unique number + ipaddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|---|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2(SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|--|
| devicesuite | string | IN | Specifies the name of the device suite, enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2(SFF-8639), valid values are Module and Host . |

string parameterString (for 5 Gb/s preemphasis)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|--|
| parameterString | string | IN | A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter is the preemphasis value. Valid values are PreEmphasis3dB and PreEmphasis6dB . The second parameter sets whether to include or exclude the preemphasis value. Valid values are Included and Excluded . String example: "DataRate2Gb \$Included". |

Set 5 Gb/s preemphasis example

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board", "", "PreEmphasis3dB$Included");
```

Set the SSC parameter

Syntax: mClient.SetGeneralParameter(clientId, device, devicesuite, "", parameterString);

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|---------------------|--|--|
| SetGeneralParameter | clientId device devicesuite parameterString | Sets the data rate. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Set SSC parameter example |

out string clientID

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientid | string | OUT | Identifier of the client that is connected to the server clientid = unique number + ipaddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|--|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2 (SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|---|
| devicesuite | string | IN | Specifies the name of the device suite, enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2 (SFF-8639), valid values are Module and Host . |

string parameterString (for SSC)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|--|
| parameterString | string | IN | A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter is SSC . The second parameter sets whether to enable or disable SSC. Valid values are On or Off . String example: "SSC \$Off". |

Set SSC example

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board",
"", "SSC$On");
```

Set the voltage swing parameter

Syntax: mClient.SetGeneralParameter(clientId, device, devicesuite, "", parameterString);

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|-----------------------------------|--|---|
| SetGeneralParameter | clientId device devicesuite parameterString | Sets the voltage swing parameter. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Set voltage swing parameter example |

out string clientId

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientid | string | OUT | Identifier of the client that is connected to the server clientId = unique number + ipaddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|--|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2 (SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|---|
| devicesuite | string | IN | Specifies the name of the device suite, enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2 (SFF-8639), valid values are Module and Host . |

string parameterString (for voltage swing)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|---|
| parameterString | string | IN | A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter is VoltageSwing . The second parameter sets the voltage swing amount. Valid values are Full and Reduced . String example: "VoltageSwing \$Reduced". This parameter affects the Signal Quality Preset tests: <ul style="list-style-type: none"> ■ Full selects all Signal quality Preset tests (P0–P10). ■ Reduced selects P01, P03 P04, P05, P06, and P09 (and in 5 Gbps only, 3.5 dB preemphasis is selected). |

Set voltage swing example

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board", "", "VoltageSwing$Full");
```


Set the signal quality preset parameter

Syntax: `mClient.SetGeneralParameter(clientId, device, devicesuite, "", parameterString);`

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|---|--|---|
| SetGeneralParameter | clientId device devicesuite parameterString | Sets the presets for the signal quality test. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | <code>m_Client = new Client() //m_Client</code> is a reference to the Client class in the Client DLL. returnval as string Set signal quality preset example |

out string clientId

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientid | string | OUT | Identifier of the client that is connected to the server clientId = unique number + ipaddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|--|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2 (SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|---|
| devicesuite | string | IN | Specifies the name of the device suite, enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2 (SFF-8639), valid values are Module and Host . |

string parameterString (for signal quality preset)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|---|
| parameterString | string | IN | A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter is SignalPreset . The second parameter sets the preset(s) to enable. Enter the preset, followed by an underscore character for additional preset selections. String example: "SignalPreset \$P0_P4_P5". |

Set signal quality preset example

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board", "", "SignalPreset$P0_P01_P04_");
```

To set all presets at one time:

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board", "", "SignalPreset$P0_P01_P02_P03_P04_P05_P06_P07_P08_P09_P10_");
```

Set the preset lanes parameter

Syntax: mClient.SetGeneralParameter(clientId, device, devicesuite, "", parameterString);

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|--|--|---|
| SetGeneralParameter | clientId device devicesuite parameterString | Sets the lanes required to run the preset test. This should be equal to or a subset of the lanes selected by the SelectedLanes parameter or as set on the application DUT panel. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Set preset lanes example |

out string clientID

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientid | string | OUT | Identifier of the client that is connected to the server clientid = unique number + ipaddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|--|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2 (SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|---|
| devicesuite | string | IN | Specifies the name of the device suite, enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2 (SFF-8639), valid values are Module and Host . |

string parameterString (for preset lanes)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|--|
| parameterString | string | IN | A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter is PresetLanes . The second parameter sets the lane(s) to enable. Enter the lane, followed by an underscore character for additional lane selections. String example: "PresetLanes \$Lane0_Lane1". |

Set preset lanes example

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board",
"", "PresetLanes$Lane0_Lane1_Lane4");
```

If all lanes required, set link width to 16 and then:

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board",
"", "PresetLanes
$Lane0_Lane1_Lane2_Lane3_Lane4_Lane5_Lane6_Lane7_Lane8_Lane9_
Lane10_Lane11_Lane12_Lane13_Lane14_Lane15");
```

Set the lane source parameter

Syntax: mClient.SetGeneralParameter(clientId, device, devicesuite, "", parameterString);

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|---|--|--|
| SetGeneralParameter | clientId device devicesuite parameterString | Sets the signal source and probing mode (single ended or differential) for each lane. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Set lane source example |

out string clientID

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientid | string | OUT | Identifier of the client that is connected to the server clientid = unique number + ipaddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|--|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2 (SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|---|
| devicesuite | string | IN | Specifies the name of the device suite, enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2 (SFF-8639), valid values are Module and Host . |

string parameterString (for lane source)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|---|
| parameterString | string | IN | <p>A string containing four parameters, separated by either a colon or \$, enclosed in quotes.</p> <ul style="list-style-type: none"> ■ The first parameter is Lane<n> Connected to:, where n is the lane number (0–15). ■ The second parameter is Lane<n>:, where n is the lane number (0–15) to which the first parameter is connected. ■ The third parameter is the probing mode. Valid values are Differential, + Single Ended, and – Single Ended. ■ The fourth parameter, separated from the first three by a \$ symbol is CH<n>:, where n is the channel number (1–4) to which the lane parameters are connected. <p>String example: "Lane0 Connected to:Lane0:Differential \$CH1".</p> |

Set lane source example

For differential probing:

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board", "", "Lane0 Connected to:Lane0:Differential$CH1");
```

For single ended probing:

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board",
"", "Lane0 Connected to:Lane0:+ Single Ended$CH1");
```

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board",
"", "Lane0 Connected to:Lane0:- Single Ended$CH1");
```

Set the preset parameter

Syntax: mClient.SetGeneralParameter(clientId, device, devicesuite, "", parameterString);

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|---|--|---|
| SetGeneralParameter | clientId device devicesuite parameterString | Sets the parameter for the Preset test. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Set preset example |

out string clientId

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientId | string | OUT | Identifier of the client that is connected to the server clientId = unique number + IPaddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|--|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2 (SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|---|
| devicesuite | string | IN | Specifies the name of the device suite, enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2 (SFF-8639), valid values are Module and Host . |

string parameterString (for preset)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|--|
| parameterString | string | IN | A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter is Preset . The second parameter sets the preset(s) to enable. Enter the preset(s), followed by an underscore character. String example: "Preset \$P0_P01_P02_P04_". |

8 Gbps preset testing dependencies

Select the required dependent preset for 8 Gbps preset testing along with primary presets:

Table 11:

| Evaluating preset | Dependent preset |
|-------------------|------------------|
| P0 | P04 |
| P01 | P04 |
| P02 | P04 |
| P03 | P04 |
| P04 | -- |
| P05 | P04 |
| P06 | P04 |
| P07 | P02, P04, P05 |
| P08 | P03, P04, P06 |

| Evaluating preset | Dependent preset |
|-------------------|------------------|
| P09 | P04 |
| P10 | P04 |

Set preset test example

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board",
"", "Preset$P0_P02_P04_");
```

To set all presets:

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board",
"", "Preset$P0_P01_P02_P03_P04_P05_P06_P07_P08_P09_P10_");
```

Set the acquisition parameter

Syntax: mClient.SetGeneralParameter(clientId, device, devicesuite, "", parameterString);

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|----------------------------|--|--|
| SetGeneralParameter | clientId device devicesuite parameterString | Sets the acquisition mode. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Set acquisition example |

out string clientId

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientId | string | OUT | Identifier of the client that is connected to the server clientId = unique number + IPaddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|---|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2(SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|--|
| devicesuite | string | IN | Specifies the name of the device suite, enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2(SFF-8639), valid values are Module and Host . |

string parameterString (for acquisition)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|--|
| parameterString | string | IN | A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter is Acquisition . The second parameter sets when acquisitions occur. Valid values are BeforeAnalysis , InSequence , and AcquireOnly . String example:"Acquisition \$BeforeAnalysis". |

Set acquisition example

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board", "Acquisition$BeforeAnalysis")
```

Set the analysis mode parameter

Syntax: `mClient.SetGeneralParameter(clientId, device, devicesuite, "", parameterString);`

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|---|--|---|
| SetGeneralParameter | clientId device devicesuite parameterString | Sets the signal analysis mode (DLL or CLI). | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | <code>m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Set analysis mode example</code> |

out string clientId

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientId | string | OUT | Identifier of the client that is connected to the server clientId = unique number + IPaddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|---|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "PCIe". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|--|
| devicesuite | string | IN | Specifies the name of the PCIe device suite, enclosed in quotes. Valid values are System-Board and Add-In-Card . |

string parameterString (for analysis mode)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|---|
| parameterString | string | IN | A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter is AnalysisMode . The second parameter sets the analysis mode. Valid values are DLL and CLI . String example: "AnalysisMode\$DLL". |

Set analysis mode example

```
mClient.SetGeneralParameter("1065-192.157.98.70", "PCIe", "System-Board", "", "AnalysisMode$DLL");
```

Set the sigtest version parameter

Syntax: mClient.SetGeneralParameter(clientId, device, devicesuite, "", **parameterString**);

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|---|--|---|
| SetGeneralParameter | clientId device devicesuite parameterString | Sets the Sigtest version or source file used for testing. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Set Sigtest version example |

out string clientId

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientId | string | OUT | Identifier of the client that is connected to the server clientId = unique number + IPaddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|--|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2 (SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|---|
| devicesuite | string | IN | Specifies the name of the device suite, enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2 (SFF-8639), valid values are Module and Host . |

string parameterString (for Sigtest version)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|--|
| parameterString | string | IN | <p>A string containing two parameters separated by a \$ symbol, enclosed in quotes.</p> <p>The first parameter is either SigTestVersion (when AnalysisMode is set to DLL) or SigTestPath (when AnalysisMode is set to CLI) .</p> <p>The second parameter sets the SigTest version source. Valid values are the SigTest version installed on the instrument (when AnalysisMode is set to DLL), or the full path to the SigTest executable file (when AnalysisMode is set to CLI).</p> <p>String example: "SigTestVersion \$3_2_0". "SigTestPath\$C:\Program Files (x86)\SigTest 3.2.6\SigTest.exe".</p> |

Set Sigtest version example

For when AnalysisMode is set to DLL:

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board", "", "SigTestVersion$3_2_0");
```

For when AnalysisMode is set to CLI:

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board", "", "SigTestPath$C:\Program Files (x86)\SigTest 3.2.6\SigTest.exe");
```

Set the on failure action parameter

Syntax: `mClient.SetGeneralParameter(clientId, device, devicesuite, "", parameterString);`

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|--|--|---|
| SetGeneralParameter | clientId device devicesuite parameterString | Sets the action taken when the application encounters a test failure. Note: Email settings must be entered in the application (with the user interface) before using this command. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | <code>m_Client = new Client() //m_Client</code> is a reference to the Client class in the Client DLL. returnval as string Set on failure action example |

out string clientId

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientId | string | OUT | Identifier of the client that is connected to the server clientId = unique number + IPaddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|--|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2 (SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|---|
| devicesuite | string | IN | Specifies the name of the device suite, enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2 (SFF-8639), valid values are Module and Host . |

string parameterString (for on failure action)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|--|
| parameterString | string | IN | A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter sets the failure action. Valid values are On Failure Stop and Notify or On Failure Pause . The second parameter enables or disables the on failure action. Valid values are True and False . String example:"On Failure Stop and Notify \$True". |

Set on failure action example

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board", "", "On Failure Stop and Notify$False");
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board", "", "On Failure Pause$True");
```

Set the report update mode parameter

Syntax: mClient.SetGeneralParameter(clientId, device, devicesuite, "", parameterString);

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|--|---|--|
| SetGeneralParameter | clientId device devicesuite parameterString | Sets how the application saves the current test run report. Note: Changes to the test report update mode must be made before running a test session. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Set report update example |

out string clientID

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientID | string | OUT | Identifier of the client that is connected to the server clientID = unique number + IPaddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|--|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2 (SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|---|
| devicesuite | string | IN | Specifies the name of the device suite, enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2 (SFF-8639), valid values are Module and Host . |

string parameterString (for report update mode)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|---|
| parameterString | string | IN | A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter is Report Update Mode . The second parameter sets how the report is saved in relation to the last report from the session. Valid values are Append , New , and Replace . String example: "Report Update Mode \$Append". |

Set report update mode example

To create a new report for each test run:

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board", "", "AReport Update Mode$New");
```

Set the append report parameter

Syntax: mClient.SetGeneralParameter(clientId, device, devicesuite, "", **parameterString**);

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|--|--|--|
| SetGeneralParameter | clientId device devicesuite parameterString | Enables or disables appending the current test session report to the previous test report. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Set append report example |

out string clientID

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientID | string | OUT | Identifier of the client that is connected to the server clientID = unique number + IPaddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|---|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "PCIe". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|--|
| devicesuite | string | IN | Specifies the name of the PCIe device suite, enclosed in quotes. Valid values are System-Board and Add-In-Card . |

string parameterString (for append report mode)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|---|
| parameterString | string | IN | A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter is Append Report . The second parameter enables or disables appending the report. Valid values are True and False . String example: "Append Report\$True". |

Set append report mode example

```
mClient.SetGeneralParameter("1065-192.157.98.70", "PCIe", "System-Board", "", "Append Report$False");
```

Set the crosstalk parameter

Syntax: mClient.SetGeneralParameter(clientId, device, devicesuite, "", parameterString);

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|---|--|---|
| SetGeneralParameter | clientId device devicesuite parameterString | Sets the DUT crosstalk type (interleaved or noninterleaved signal routing). | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Set crosstalk mode example |

out string clientId

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientId | string | OUT | Identifier of the client that is connected to the server clientId = unique number + IPaddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|--|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2 (SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|---|
| devicesuite | string | IN | Specifies the name of the device suite, enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2 (SFF-8639), valid values are Module and Host . |

string parameterString (for crosstalk mode)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|---|
| parameterString | string | IN | A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter is CrossTalk . The second parameter sets the crosstalk mode. Valid values are On and Off . String example: "CrossTalk \$On." |

Set crosstalk mode example

To set Crosstalk mode on (same as selecting the application control **Crosstalk (noninterleaved routing)**):

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board",
"", "CrossTalk$On");
```

To set Crosstalk mode off (same as selecting the application control **No Crosstalk (interleaved routing)**):

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board",
"", "CrossTalk$Off");
```

Set the waveform save parameter

Syntax: mClient.SetGeneralParameter(clientId, device, devicesuite, "", parameterString);

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|---------------------------------------|--|--|
| SetGeneralParameter | clientId device devicesuite parameterString | Sets the presets for the Preset test. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Set waveform save parameter example |

out string clientID

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientID | string | OUT | Identifier of the client that is connected to the server clientID = unique number + IPaddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|--|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2 (SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|---|
| devicesuite | string | IN | Specifies the name of the device suite, enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2 (SFF-8639), valid values are Module and Host . |

string parameterString (for waveform save)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|---|
| parameterString | string | IN | A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter is SaveOptions . The second parameter sets when acquisitions occur. Valid values are Save All the Waveforms, Save Waveforms after applying Filters, No Waveforms saved - Discard after analysis, and Analyze Immediately - No Waveforms saved . String example: "SaveOptions \$Save All the Waveforms". |

Set waveform save parameter example

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board", "", "SaveOptions$Save Waveforms after applying Filters");
```

Set the link analysis embed-de-embed signal parameter

Syntax: mClient.SetGeneralParameter(clientId, device, devicesuite, "", parameterString);

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|---|--|--|
| SetGeneralParameter | clientId device devicesuite parameterString | Embeds or de-embeds the 2.5, 5, and 8 Gb/s signals. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Set link analysis embed/de-embed signal example |

out string clientID

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientID | string | OUT | Identifier of the client that is connected to the server clientID = unique number + IPaddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|--|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2 (SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|---|
| devicesuite | string | IN | Specifies the name of the device suite, enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2 (SFF-8639), valid values are Module and Host . |

string parameterString (for embed/de-embed signal)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|---|
| parameterString | string | IN | A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter values are DeEmbed2Gb , DeEmbed5Gb , DeEmbed8Gb , Embed8Gb , or Equalization8Gb . The second parameter sets whether to include or exclude the first parameter. Valid values are Included and Excluded . String example: "Embed8Gb \$Included". |

Set embed/de-embed signal example

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board",
"", "Embed2Gb$Included");
```

**Set the link analysis
embed-de-embed filter file
parameter**

Syntax: mClient.SetGeneralParameter(clientId, device, devicesuite, "",
parameterString);

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|--|--|---|
| SetGeneralParameter | clientId device devicesuite parameterString | Embeds or de-embeds the specified filter file. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Set link analysis embed/de-embed filter file example |

out string clientID

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientID | string | OUT | Identifier of the client that is connected to the server clientID = unique number + IPaddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|--|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2 (SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|---|
| devicesuite | string | IN | Specifies the name of the device suite, enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2 (SFF-8639), valid values are Module and Host . |

string parameterString (for embed/de-embed filter file)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|---|
| parameterString | string | IN | <p>A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter value is Filterfile2Gb, Filterfile5Gb, FilterfileEmbed8Gb, or FilterfileDeEmbed8Gb.</p> <p>The second parameter specifies the .flt filter file name to load. String example: "FilterFileEmbed8Gb\$C:\PCI_Filters\Gen1.flt". PCIe does not include any filter files.</p> |

Set embed/de-embed filter file example

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board", "", "FilterFile5Gb$file.flt");
```

Set the link analysis other filter file parameter

Syntax: mClient.SetGeneralParameter(clientId, device, devicesuite, "", **parameterString**);

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|--|--|---|
| SetGeneralParameter | clientId device devicesuite parameterString | Loads the specified Sigtest or user-defined filter file. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Set load other filter file example |

out string clientID

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientID | string | OUT | Identifier of the client that is connected to the server clientID = unique number + IPaddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|--|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2 (SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|---|
| devicesuite | string | IN | Specifies the name of the device suite, enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2 (SFF-8639), valid values are Module and Host . |

string parameterString (for other filter file)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|--|
| parameterString | string | IN | A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter value is EmbedDropdown . Valid second parameter values are SigTest and Scope . String example: "EmbedDropdown\$SigTest". |

Set load other filter file example

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board",
"", "EmbedDropdown$Scope");
```

**Set the link analysis
equalization dropdown
parameter**

Syntax: mClient.SetGeneralParameter(clientId, device, devicesuite, "",
parameterString);

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|--|--|--|
| SetGeneralParameter | clientId device devicesuite parameterString | Loads the specified Sigtest or user-defined filter file. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Set equalization dropdown example |

out string clientId

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientId | string | OUT | Identifier of the client that is connected to the server clientId = unique number + IPaddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|--|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2 (SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|---|
| devicesuite | string | IN | Specifies the name of the device suite, enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2 (SFF-8639), valid values are Module and Host . |

string parameterString (for equalization dropdown)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|--|
| parameterString | string | IN | A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter value is EqualizationDropdown . Valid second parameter values are Optimize and Fixed . String example: "EqualizationDropdown\$Optimize". |

Set equalization dropdown example

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board", "", "EqualizationDropdown$Fixed");
```

Set the link analysis CTLE index parameter

Syntax: `mClient.SetGeneralParameter(clientId, device, devicesuite, "", parameterString);`

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|--------------------------------|--|--|
| SetGeneralParameter | clientId device devicesuite parameterString | Sets the CTLE Index parameter. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | <code>m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Set CTLE index example</code> |

out string clientId

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientId | string | OUT | Identifier of the client that is connected to the server clientId = unique number + IPAddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|--|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2 (SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|---|
| devicesuite | string | IN | Specifies the name of the device suite, enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2 (SFF-8639), valid values are Module and Host . |

string parameterString (for CTLE index)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|---|
| parameterString | string | IN | A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter value is CTLE Index . Valid second parameter values are 1: -12db, 2: -11db, 3: -10db, 4: -9db, 5: -8db, and 6: -7db . String example: "CTLE Index\$3 : -10dB". |

Set CTLE index example

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board", "", "CTLE Index$5 : -8dB");
```

Set the link analysis DFE parameter

Syntax: mClient.SetGeneralParameter(clientId, device, devicesuite, "", **parameterString**);

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|---------------------|--|--|
| SetGeneralParameter | clientId device devicesuite parameterString | Sets the DFE value. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Set DFE example |

out string clientId

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientId | string | OUT | Identifier of the client that is connected to the server clientId = unique number + IPaddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|--|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2 (SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|---|
| devicesuite | string | IN | Specifies the name of the device suite, enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2 (SFF-8639), valid values are Module and Host . |

string parameterString (for DFE)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|--|
| parameterString | string | IN | A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter value is DFE . The second parameter specifies the DFE value, using standard scientific notation. String example:"DFE \$30e-3". |

Set DFE example

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board", "", "DFE$30e-3");
```

Set the DUT auto toggle parameter

Syntax: mClient.SetGeneralParameter(clientId, device, devicesuite, "", parameterString);

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|---|--|--|
| SetGeneralParameter | clientId device devicesuite parameterString | Enables DUT Automation if AFG is connected. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Set DUT auto toggle example |

out string clientId

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientId | string | OUT | Identifier of the client that is connected to the server clientId = unique number + IPaddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|--|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2 (SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|---|
| devicesuite | string | IN | Specifies the name of the device suite, enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2 (SFF-8639), valid values are Module and Host . |

string parameterString (for DUT auto toggle)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|--|
| parameterString | string | IN | A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter value is EnableDUTAutomation . The second parameter values are Included and Excluded . String example: "EnableDUTAutomation\$Included". |

Set DUT auto toggle example

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board", "", "EnableDUTAutomation$Included");
```

Set the DUT auto toggle options parameter

Syntax: mClient.SetGeneralParameter(clientId, device, devicesuite, "", parameterString);

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|--|--|--|
| SetGeneralParameter | clientId device devicesuite parameterString | Sets DUT Automation settings preference. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Set DUT auto toggle options example |

out string clientId

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientId | string | OUT | Identifier of the client that is connected to the server clientId = unique number + IPaddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|--|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2 (SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|---|
| devicesuite | string | IN | Specifies the name of the device suite, enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2 (SFF-8639), valid values are Module and Host . |

string parameterString (for DUT auto toggle options)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|--|
| parameterString | string | IN | A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter value is Automation Settings . The second parameter values are Use Default Settings , Manually Configure Settings , and Use Custom Settings . String example: "Automation Settings\$Use Default Settings". |

Set DUT auto toggle options example

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board", "", "Automation Settings$Use Custom Settings");
```

Set AFG signal type parameter

Syntax: `mClient.SetGeneralParameter(clientId, device, devicesuite, "", parameterString);`

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|--|--|---|
| SetGeneralParameter | clientId device devicesuite parameterString | Sets the signal type (square or sine wave) from AFG or AWG source. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | <code>m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Set signal source example</code> |

out string clientId

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientId | string | OUT | Identifier of the client that is connected to the server clientId = unique number + IPAddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|---|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2(SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|--|
| devicesuite | string | IN | Specifies the name of the device suite, enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2(SFF-8639), valid values are Module and Host . |

string parameterString (for AFG/AWG signal type)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|---|
| parameterString | string | IN | A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter value is Signal Type . The second parameter values are Square and Sine . String example: "Signal Type\$Sine". |

Set AFG/AWG signal type example

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board", "", "Signal Type$Square");
```

Set the AFG signal frequency parameter

Syntax: mClient.SetGeneralParameter(clientId, device, devicesuite, "", **parameterString**);

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|--|--|---|
| SetGeneralParameter | clientId device devicesuite parameterString | Specifies the AFG source signal frequency. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Set AFG signal frequency example |

out string clientId

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientId | string | OUT | Identifier of the client that is connected to the server clientId = unique number + IPaddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|---|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "PCIe". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|--|
| devicesuite | string | IN | Specifies the name of the PCIe device suite, enclosed in quotes. Valid values are System-Board and Add-In-Card . |

string parameterString (for AFG signal frequency)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|---|
| parameterString | string | IN | A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter value is AFGFrequency . The second parameter specifies the frequency of the AFG source, using standard scientific notation. String example: "AFGFrequency\$80e6". |

Set AFG signal frequency example

```
mClient.SetGeneralParameter("1065-192.157.98.70", "PCIe", "System-Board", "", "AFGFrequency$80e6");
```

Set the AFG signal amplitude parameter

Syntax: mClient.SetGeneralParameter(clientId, device, devicesuite, "", parameterString);

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|--|--|---|
| SetGeneralParameter | clientId device devicesuite parameterString | Specifies the AFG source signal amplitude. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Set AFG signal amplitude example |

string parameterString (for AFG signal amplitude)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|--|
| parameterString | string | IN | A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter value is AFGAmplitude . The second parameter specifies the amplitude of the AFG source, using standard scientific notation. String example:"AFGAmplitude\$80". |

out string clientId

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientId | string | OUT | Identifier of the client that is connected to the server clientId = unique number + IPaddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|---|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "PCIe". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|--|
| devicesuite | string | IN | Specifies the name of the PCIe device suite, enclosed in quotes. Valid values are System-Board and Add-In-Card . |

Set AFG signal amplitude example

```
mClient.SetGeneralParameter("1065-192.157.98.70", "PCIe", "System-Board", "", "AFGAmplitude$60");
```

Set the burst count parameter

Syntax: mClient.SetGeneralParameter(clientId, device, devicesuite, "", parameterString);

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|--|--|--|
| SetGeneralParameter | clientId device devicesuite parameterString | Specifies the burst count parameters related to the AFG. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Set burst count example |

out string clientId

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientId | string | OUT | Identifier of the client that is connected to the server clientId = unique number + IPAddress of the client. For example, 1065-192.157.98.70 |

string parameterString (for burst count)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|---|
| parameterString | string | IN | A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter value is Burst Count . The second parameter specifies the size of the burst count. String example: "Burst Count\$100k". |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|--|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2 (SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|---|
| devicesuite | string | IN | Specifies the name of the device suite, enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2 (SFF-8639), valid values are Module and Host . |

Set burst count example

```
mClient.SetGeneralParameter("1065-192.157.98.70", "PCIe", "System-Board", "", "Burst Count$100k");
```

Set the record length parameter

Syntax: `mClient.SetGeneralParameter(clientId, device, devicesuite, "", parameterString);`

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|--|--|---|
| SetGeneralParameter | clientId device devicesuite parameterString | Sets the acquisition record length of the specified signal rate. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | <code>m_Client = new Client() //m_Client</code> is a reference to the Client class in the Client DLL. returnval as string Set record length example |

out string clientId

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientId | string | OUT | Identifier of the client that is connected to the server clientId = unique number + IPAddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|--|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2 (SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|---|
| devicesuite | string | IN | Specifies the name of the device suite, enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2 (SFF-8639), valid values are Module and Host . |

string parameterString (for record length)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|--|
| parameterString | string | IN | A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter value is RecordLength2Gb , RecordLength5Gb , or RecordLength8Gb . The second parameter sets the record length using standard scientific notation. String example: "RecordLength5Gb\$10e6". |

Set record length example

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board", "", "RecordLength2Gb$10e6");
```

Set the sample rate parameter

Syntax: mClient.SetGeneralParameter(clientId, device, devicesuite, "", **parameterString**);

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|--|--|--|
| SetGeneralParameter | clientId device devicesuite parameterString | Sets the sampling rate of the specified signal type. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Set sample rate example |

out string clientId

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientId | string | OUT | Identifier of the client that is connected to the server clientId = unique number + IPaddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|--|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2 (SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|---|
| devicesuite | string | IN | Specifies the name of the device suite, enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2 (SFF-8639), valid values are Module and Host . |

string parameterString (for sampling rate)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|--|
| parameterString | string | IN | A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter value is SampleRate2Gb , SampleRate5Gb , or SampleRate8Gb . The second parameter sets the sample rate using standard scientific notation. String example: "SampleRate2Gb\$25e9". |

Set sample rate example

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board", "", "SampleRate8Gb$25e9");
```

Set the bandwidth parameter

Syntax: mClient.SetGeneralParameter(clientId, device, devicesuite, "", parameterString);

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|--|--|--|
| SetGeneralParameter | clientId device devicesuite parameterString | Sets the bandwidth of the specified signal type. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Set bandwidth example |

out string clientId

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientId | string | OUT | Identifier of the client that is connected to the server clientId = unique number + IPaddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|---|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2(SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|---|
| devicesuite | string | IN | Specifies the name of the device suite, enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2 (SFF-8639), valid values are Module and Host . |

string parameterString (for bandwidth)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|--|
| parameterString | string | IN | A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter value is Bandwidth2Gb , Bandwidth5Gb , or Bandwidth8Gb . The second parameter bandwidth frequency using standard scientific notation. String example:"Bandwidth2Gb\$b\$6e9". |

Set bandwidth example

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board", "", "Bandwidth8Gb$6e9");
```

Set the signal validation parameter

Syntax: mClient.SetGeneralParameter(clientId, device, devicesuite, "", parameterString);

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|--|--|--|
| SetGeneralParameter | clientId device devicesuite parameterString | Sets the signal check (validation) action. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Set signal validation example |

out string clientId

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientId | string | OUT | Identifier of the client that is connected to the server clientId = unique number + IPaddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|--|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2 (SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|---|
| devicesuite | string | IN | Specifies the name of the device suite, enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2 (SFF-8639), valid values are Module and Host . |

string parameterString (for signal validation)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|---|
| parameterString | string | IN | A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter value is Signal Validation . The second parameter sets the signal check (validation) action. Valid values are Prompt me if Signal Check Fails, Skip Test if Signal Check Fails, or Turn Off Signal Check . String example: "Signal Validation\$Prompt me if Signal Check Fails". |

Set signal validation example

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board", "", "Signal Validation$Turn Off Signal Check");
```


Set the slot number parameter

Syntax: `mClient.SetGeneralParameter(clientId, device, devicesuite, "", parameterString);`

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|----------------------------|--|---|
| SetGeneralParameter | clientId device devicesuite parameterString | Sets the test slot number. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | <code>m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Set slot number example</code> |

out string clientId

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientId | string | OUT | Identifier of the client that is connected to the server clientId = unique number + IPAddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|--|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2 (SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|---|
| devicesuite | string | IN | Specifies the name of the device suite, enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2 (SFF-8639), valid values are Module and Host . |

string parameterString (for select slot number)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|---|
| parameterString | string | IN | A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter value is SlotNumber . The second parameter specifies the slot number. Valid values are 01 through 08 . String example: "SlotNumber \$02". |

Set slot number example

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board", "", "SlotNumber$05");
```

Set SigTest Interface Mode

Syntax: mClient.SetGeneralParameter(clientId, device, devicesuite, "", **parameterString**);

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|----------------------------|--|---|
| SetGeneralParameter | clientId device devicesuite parameterString | Sets the test slot number. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Set SigTest Interface mode example |

out string clientId

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientId | string | OUT | Identifier of the client that is connected to the server clientId = unique number + IPaddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|---|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2(SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|--|
| devicesuite | string | IN | Specifies the name of the device suite, enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2(SFF-8639), valid values are Module and Host . |

string parameterString (to set SigTest interface mode)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|---|
| parameterString | string | IN | A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter value is AnalysisMode . The second parameter specifies the slot number. Valid values are CLI through DLL . String example: "AnalysisMode\$CLI". |

Set SigTest interface mode example

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board", "AnalysisMode$CLI");
```

Set the trigger type parameters

Syntax: mClient.SetGeneralParameter(clientId, device, devicesuite, "", parameterString);

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|--|--|--|
| SetGeneralParameter | clientId device devicesuite parameterString | Sets the acquisition record length of the specified signal rate. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Set record length example |

out string clientId

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientId | string | OUT | Identifier of the client that is connected to the server clientId = unique number + IPaddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|--|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2 (SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|---|
| devicesuite | string | IN | Specifies the name of the device suite, enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2 (SFF-8639), valid values are Module and Host . |

string parameterString (for trigger type)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|--|
| parameterString | string | IN | A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter value is Trigger Type . The second parameter is Auto , Edge , or Width . sets the trigger type using standard scientific notation. String example:"Trigger Type\$Auto". |

Set trigger type example

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board",
"", "Trigger Type$Auto");
```

NOTE. *Trigger Type is applicable for Version Gen3 (3.0) only.*

Set Sig Validation Threshold

Syntax: mClient.SetGeneralParameter(clientId, device, devicesuite, "", **parameterString**);

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|----------------------------|--|---|
| SetGeneralParameter | clientId device devicesuite parameterString | Sets the test slot number. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Set Sig Validation Threshold example |

out string clientID

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientID | string | OUT | Identifier of the client that is connected to the server clientID = unique number + IPaddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|---|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2(SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|--|
| devicesuite | string | IN | Specifies the name of the device suite, enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2(SFF-8639), valid values are Module and Host . |

string parameterString (to set signal validation threshold)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|--|
| parameterString | string | IN | A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter value is Signal Validation Threshold(mV) . The second parameter specifies the slot number. Valid values are 50 through 400 . String example: "Signal Validation Threshold(mV)\$325". |

Set signal validation threshold example

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board", "Signal Validation Threshold(mV)$325");
```

Set Toggle from Gen3P10 to Gen1

Syntax: mClient.SetGeneralParameter(clientId, device, devicesuite, "", parameterString);

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|----------------------------|--|--|
| SetGeneralParameter | clientId device devicesuite parameterString | Sets the test slot number. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Set Toggle from Gen3P10 to Gen1 example |

out string clientId

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientId | string | OUT | Identifier of the client that is connected to the server clientId = unique number + IPaddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|---|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "PCIe". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|--|
| devicesuite | string | IN | Specifies the name of the PCIe device suite, enclosed in quotes. Valid values are System-Board and Add-In-Card . |

string parameterString (for select slot number)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|--|
| parameterString | string | IN | A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter value is Toggle from Gen3P10 to Gen1 . The second parameter specifies the slot number. Valid values are On through Off . String example: "Toggle from Gen3P10 to Gen1\$Off". |

Set Toggle from Gen3P10 to Gen1

```
mClient.SetGeneralParameter("1065-192.157.98.70", "PCIe", "System-Board", "Toggle from Gen3P10 to Gen1$Off");
```


Set the group test results by parameters

Syntax: `mClient.SetGeneralParameter(clientId, device, devicesuite, "", parameterString);`

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|--|--|---|
| SetGeneralParameter | clientId device devicesuite parameterString | Sets the acquisition record length of the specified signal rate. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | <code>m_Client = new Client() //m_Client</code> is a reference to the Client class in the Client DLL. returnval as string Set record length example |

out string clientId

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientId | string | OUT | Identifier of the client that is connected to the server clientId = unique number + IPAddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|--|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2 (SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|---|
| devicesuite | string | IN | Specifies the name of the device suite, enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2 (SFF-8639), valid values are Module and Host . |

string parameterString (for group test results)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|---|
| parameterString | string | IN | A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter value is Report Group Mode . The second parameter is Test Name, Lane Name, Equalization or Test Result . sets the group test results using standard scientific notation. String example: "Report Group Mode\$Test Result". |

Set group test results example

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board", "", "Report Group Mode$Test Result");
```

Set the report creation path parameter

Syntax: mClient.SetGeneralParameter(clientId, device, devicesuite, "", **parameterString**);

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|--|--|--|
| SetGeneralParameter | clientId device devicesuite parameterString | Sets the acquisition record length of the specified signal rate. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Set record length example |

out string clientID

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientID | string | OUT | Identifier of the client that is connected to the server clientID = unique number + IPaddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|--|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2 (SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|---|
| devicesuite | string | IN | Specifies the name of the device suite, enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2 (SFF-8639), valid values are Module and Host . |

string parameterString (for report creation path)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|--|
| parameterString | string | IN | A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter value is Report Path . The second parameter is X:\PCI Express \Reports\DUT001.pdf . sets the record length using standard scientific notation. String example: "Report Path\$X:\PCI Express \Reports\DUT001.pdf". |

Set report creation path example

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board", "", "Report Path$X:\PCI Express\Reports\DUT001.pdf");
```

Set the report contents to save parameters

Syntax: mClient.SetGeneralParameter(clientId, device, devicesuite, "", **parameterString**);

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|--|--|--|
| SetGeneralParameter | clientId device devicesuite parameterString | Sets the acquisition record length of the specified signal rate. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Set record length example |

out string clientId

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientId | string | OUT | Identifier of the client that is connected to the server clientId = unique number + IPaddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|--|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2 (SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|---|
| devicesuite | string | IN | Specifies the name of the device suite, enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2 (SFF-8639), valid values are Module and Host . |

string parameterString (for report contents to save)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|--|
| parameterString | string | IN | <p>A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter value are:</p> <ul style="list-style-type: none"> ■ Include Pass/Fail Results Summary ■ Include Detailed Results ■ Include Plot Images ■ Include Setup Configuration ■ Include User Comment <p>.</p> <p>The second parameter is True, or False. sets the report contents to save using standard scientific notation. String example:"Include Pass/Fail Results Summary\$True".</p> |

Set report contents to save example

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board",
"", "Include Detailed Results$True");
```

Set the report creation type parameter

Syntax: `mClient.SetGeneralParameter(clientId, device, devicesuite, "", parameterString);`

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|--|--|---|
| SetGeneralParameter | clientId device devicesuite parameterString | Sets the acquisition record length of the specified signal rate. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | <code>m_Client = new Client() //m_Client</code> is a reference to the Client class in the Client DLL. returnval as string Set record length example |

out string clientId

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientId | string | OUT | Identifier of the client that is connected to the server clientId = unique number + IPAddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|--|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2 (SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|---|
| devicesuite | string | IN | Specifies the name of the device suite, enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2 (SFF-8639), valid values are Module and Host . |

string parameterString (for report creation path)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|--|
| parameterString | string | IN | A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter value is Save As Type . The second parameter is PDF or Web Archive . sets the record length using standard scientific notation. String example: "Save As Type\$Web Archive". |

Set report creation path example

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board", "", "Save As Type$Web Archive");
```

Set the auto increment report name if duplicate parameter

Syntax: mClient.SetGeneralParameter(clientId, device, devicesuite, "", **parameterString**);

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|--|--|--|
| SetGeneralParameter | clientId device devicesuite parameterString | Sets the acquisition record length of the specified signal rate. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Set record length example |

out string clientId

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientId | string | OUT | Identifier of the client that is connected to the server clientId = unique number + IPaddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|--|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2 (SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|---|
| devicesuite | string | IN | Specifies the name of the device suite, enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2 (SFF-8639), valid values are Module and Host . |

string parameterString (for report creation path)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|--|
| parameterString | string | IN | A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter value is Auto increment report name if duplicate . The second parameter is True or False . sets the record length using standard scientific notation. String example: "Auto increment report name if duplicate\$True". |

Set report creation path example

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board", "", "Auto increment report name if duplicate$True");
```

Set the view report after generating parameter

Syntax: mClient.SetGeneralParameter(clientId, device, devicesuite, "", parameterString);

| Command name | Parameters | Description | Return value | Example |
|---------------------|---|--|--|--|
| SetGeneralParameter | clientId device devicesuite parameterString | Sets the acquisition record length of the specified signal rate. | String that gives the status of the operation after it was performed. The return value is "" (an empty String) on success. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Set record length example |

out string clientId

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientId | string | OUT | Identifier of the client that is connected to the server clientId = unique number + IPaddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|--|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2 (SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|---|
| devicesuite | string | IN | Specifies the name of the device suite, enclosed in quotes. For CEM, valid values are System-Board and Add-In-Card . For U.2 (SFF-8639), valid values are Module and Host . |

string parameterString (for view report after generating)

| Name | Type | Direction | Description |
|-----------------|--------|-----------|---|
| parameterString | string | IN | A string containing two parameters separated by a \$ symbol, enclosed in quotes. The first parameter value is View Report After Generating . The second parameter is True , or False . sets the record length using standard scientific notation. String example:"View Report After Generating\$True". |

Set record length example

```
mClient.SetGeneralParameter("1065-192.157.98.70", "CEM", "System-Board", "", "View Report After Generating$True");
```

Run with set configurations or stop the run operation

| Command name | Parameters | Description | Return value | Example |
|--------------|-----------------|--|--|--|
| Run() | string clientID | Runs the selected tests Note After the server is set up and configured, run it remotely using this function. | String that gives the status of the operation after it was performed. The return value is "Run started..." on success. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string returnval=m_Client.Run(clientID) |
| Stop() | string clientID | Stops the running tests. Note | String that gives the status of the operation after it was performed The return value is "Stopped..." on success. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string returnval=m_Client.Stop(clientID) |

out string clientID

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientID | string | OUT | Identifier of the client that is connected to the server clientID = unique number + IPaddress of the client. For example, 1065-192.157.98.70 |

NOTE. When the run is performed, the status of the run is updated periodically using a timer.

NOTE. When the session is stopped, the client is prompted to stop the session and is stopped at the consent.

NOTE. The Fail condition for PI commands occurs in any of the following cases:

If the server is locked, the application displays "Server is locked by another client."

If the session is unlocked, the application displays "Lock session to execute the command."

If the server is not found, the application displays "Server not found-Disconnect!."

If the fail condition is not one of the above types, the application displays "Failed."

Handle error codes

The return value of the remote automations at the server-end is OP_STATUS, which changes to a string value depending on its code, and returned to the client. The values of OP_STATUS are as follows:

| Code | Value | Description |
|------|-----------|--|
| -1 | FAIL | The operation failed |
| 1 | SUCCESS | The operation succeeded |
| 2 | NOT FOUND | Server not found |
| 3 | LOCKED | The server is locked by another client, so the operation cannot be performed |

| Code | Value | Description |
|------|--------|---|
| 4 | UNLOCK | The server is not locked; lock the server before performing the operation |
| 0 | NULL | Nothing |

Save recall or query a saved session

| Command name | Parameters | Description | Return value | Example |
|---------------------|-----------------------------------|--|--|--|
| CheckSessionSaved() | string clientID out bool saved | This method checks whether the current session is saved. | Return value is either True or False | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string returnval=m_Client.CheckSessionSaved(m_clientID, out savedStatus) |
| RecallSession() | string clientID string name | Recalls a saved session. The client provides the session name. | String that gives the status of the operation after it was performed The return value is "Session Recalled..." | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string returnval=m_Client.RecallSession(clientID, savedSessionName) |
| SaveSession() | string clientID string name | Saves the current session. The client provides the session name. | String that gives the status of the operation after it was performed The return value is "Session Saved..."/"Failed..." | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string returnval=m_Client.SaveSession(clientID, desiredSessionName) |
| SaveSessionAs() | string clientID string name | Saves the current session under a different name every time this method is called. The client provides the session name. | String that gives the status of the operation after it was performed The return value is "Session Saved..." | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string returnval=m_Client.SaveSessionAs(clientID, desiredSessionName) |

out string clientID

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientid | string | OUT | Identifier of the client that is connected to the server clientid = unique number + ipaddress of the client. For example, 1065-192.157.98.70 |

out bool saved

| Name | Type | Direction | Description |
|-------|------|-----------|---|
| saved | bool | OUT | Boolean representing whether the current session is saved |

This parameter is used as a check in SaveSession() and SaveSessionAs() functions.

string name

| Name | Type | Direction | Description |
|------|--------|-----------|--|
| name | string | IN | The name of the session being recalled |

The name parameter cannot be empty. If it is empty, the client is prompted to provide a valid name.

NOTE. *The Fail condition for PI commands occurs in any of the following cases:*

If the server is locked, the application displays "Server is locked by another client."

If the session is unlocked, the application displays "Lock session to execute the command."

If the server is not found, the application displays "Server not found-Disconnect!."

If the fail condition is not one of the above types, the application displays "Failed."

Get or set the timeout value

| Command name | Parameters | Description | Return value | Example |
|--------------|--------------------------------|---|--|--|
| GetTimeOut() | string clientID | Returns the current timeout period set by the client | String that gives the status of the operation after it was performed The default return value is 1800000. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string returnval=m_Client.GetTimeOut() |
| SetTimeOut() | string clientID string time | Sets a timeout period specified by the client. After this timeout period expires, the server is unlocked automatically. | String that gives the status of the operation after it was performed On success the return value is "TimeOut Period Changed". | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string returnval=m_Client.SetTimeOut(clientID, desiredTimeOut) |

out string clientID

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientID | string | OUT | Identifier of the client that is connected to the server clientID = unique number + IPaddress of the client. For example, 1065-192.157.98.70 |

string time

| Name | Type | Direction | Description |
|------|--------|-----------|---|
| time | string | IN | The time in seconds that refers to the timeout period |

The time parameter gives the timeout period, which is the time the client is allowed to be locked and idle. After the timeout period if the client is still idle, it gets unlocked.

The time parameter should be a positive integer; otherwise, the client is prompted to provide a valid timeout period.

NOTE. *The Fail condition for PI commands occurs in any of the following cases:*

If the server is locked, the application displays "Server is locked by another client."

If the session is unlocked, the application displays "Lock session to execute the command."

If the server is not found, the application displays "Server not found-Disconnect!."

If the fail condition is not one of the above types, the application displays "Failed."

Wait for the test to complete

The commands in this group execute while tests are running. The GetCurrentStateInfo() and SendResponse() commands are executed when the application is running and in the wait state.

| Command name | Parameters | Description | Return value | Example |
|-----------------------|---|---|--|---|
| ApplicationStatus() | string clientID | This method gets the status of the server application. The states are Running, Paused, Wait, and Error | String value that gives the status of the server application | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string returnval=m_Client.ApplicationStatus(clientID) |
| QueryStatus() | string clientID out string[] status | An interface for the user to transfer Analyze panel status messages from the server to the client | String that gives the status of the operation after it was performed On success the return value is "Transferred...". | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Query status example |
| GetCurrentStateInfo() | string clientID out string WaitingMsbBxCaption out string WaitingMsbBxMessage out string[] WaitingMsbBxButtonContexts | This method gets the additional information of the states when the application is in Wait or Error state. Except client ID, all the others are Out parameters. If the application is in Wait state, and the caption and button texts are retrieved, if the caption is "Unable to Trigger," then Cancel is an expected response. Note: If cancel is sent to skip the acquisition, then the string must have a space at the end (example: "Cancel "). Otherwise OK is considered as the response. | This command does not return any value. This function populates the Out parameters that are passed when invoking this function. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL m_Client.GetCurrentStateInfo(clientID, WaitingMsbBxCaption, WaitingMsbBxMessage, WaitingMsbBxButtonContexts) |

| Command name | Parameters | Description | Return value | Example |
|---|--|--|---|--|
| SendResponse() <i>NOTE. This command is used when the application is running and is in the wait or error state.</i> | string clientID out string WaitingMsbBxCaption out string WaitingMsbBxMessage string WaitingMsbBxResponse | After receiving the additional information using the method GetCurrentStateInfo(), the client can decide which response to send and then send the response to the application using this function. The response should be one of the strings that was received earlier as a string array in the GetCurrentStateInfo function. The _caption and _message should match the information received earlier in the GetCurrentStateInfo function. | This command does not return any value. | <pre>m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL m_Client.SendResponse(clientID, WaitingMsbBxCaption, WaitingMsbBxMessage, WaitingMsbBxResponse)</pre> |

out string clientID

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientid | string | OUT | Identifier of the client that is connected to the server clientid = unique number + ipaddress of the client. For example, 1065-192.157.98.70 |

out string[] status

| Name | Type | Direction | Description |
|--------|--------------|-----------|--|
| status | string array | OUT | The list of status messages generated during the run |

out string WaitingMsbBxCaption

| Name | Type | Direction | Description |
|---------|--------|-----------|---|
| caption | string | OUT | The wait state or error state message sent to you |

out string WaitingMsbBxMessage

| Name | Type | Direction | Description |
|---------|--------|-----------|--|
| message | string | OUT | The wait state/error state message sent to you |

out string[] WaitingMsbBxButtontexts

| Name | Type | Direction | Description |
|-------------|--------------|-----------|--|
| buttonTexts | string array | OUT | An array of strings containing the possible response types that you can send |

string WaitingMsbBxResponse

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| response | string | IN | A string containing the response type that you can select (it must be one of the strings in the string array buttonTexts) |

Ready: Test configured and ready to start

Running: Test running

Paused: Test paused

Wait: A popup that needs your inputs

Error: An error is occurred

NOTE. The Fail condition for PI commands occurs in any of the following cases:

If the server is locked, the application displays "Server is locked by another client."

If the session is unlocked, the application displays "Lock session to execute the command."

If the server is not found, the application displays "Server not found-Disconnect!."

If the fail condition is not one of the above types, the application displays "Failed."

Query status example

```
returnVal=m_Client.QueryStatus(clientID, out statusMessages)
if ((OP_STATUS)returnVal == OP_STATUS.SUCCESS)
return "Status updated..."
else
return CommandFailed(returnVal)
```

After the test is complete

| Command name | Parameters | Description | Return value | Example |
|--|---|---|---|--|
| GetPassFailStatus() | string clientID string device string suite string test | This method gets the pass or fail status of the measurement after test completion. <i>NOTE. Execute this command after completing the measurement.</i> | String that gives the status of the operation after it was performed. Returns the pass or fail status in the form of a string | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string returnval=m_Client.GetPassFailStatus(clientID, device, devicesuite, test) Get pass/fail status for a measurement example |
| GetResultsValue() | string clientID string device string suite string test string parameterString | This method gets the result values of the measurement after the run. | String that gives the status of the operation after it was performed. Returns the result value in the form of a string | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string returnval=m_Client.GetResultsValue(clientID, device, devicesuite, test, parameterString) Get pass/fail status for a measurement example |
| To get all results of a test, first get the value by passing the column name, for example 'Value,' and then pass true value for the last argument. This command will return a value string which contains values for all of the test details in a comma-separated format. See example at end of table. | | | | |

| Command name | Parameters | Description | Return value | Example |
|----------------------|--|--|--|--|
| GetReportParameter() | string clientID string device string suite string test string parameterString | This method gets the general report details such as oscilloscope model and TekExpress version. | The return value is the oscilloscope model, TekExpress application version, or PCIe application version. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string Oscilloscope Model returnval=m_Client.GetReportParameter(clientID,"Scope Model") TekExpress Version returnval=m_Client.GetReportParameter(clientID,"TekExpress Version") PCIe Version returnval=m_Client.GetReportParameter(clientID,"Application Version") |

| Command name | Parameters | Description | Return value | Example |
|--|------------------------------------|--|--|--|
| TransferResult() <i>NOTE. The target folder must have write permission when transferring the results and images from the server machine to the client machine. Otherwise the transfer will fail.</i> | string clientID string filePath | This method transfers the report generated after the run. The report contains the summary of the run. The client must provide the location where the report is to be saved at the client-end. | String that gives the status of the operation after it was performed. Transfers all the result values in the form of a string. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string returnval=m_Client.TransferReport(clientID,"C:\Report") |
| TransferImages() <i>NOTE. The target folder must have write permission when transferring the results and images from the server machine to the client machine. Otherwise the transfer will fail.</i> | string clientID string filePath | This method transfers all the images (screen shots) from the specified client and folder for the current run (for a suite or measurement). <i>NOTE. Every time you click Start, a folder is created in the X: drive. Transfer the waveforms before clicking Start.</i> | String that gives the status of the operation after it was performed. Transfers all the images in the form of a string. | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string returnval=m_Client.TransferImages(clientID, "C:\Waveforms") |

getResultsValue example:

```
string values = mClient.GetResultsValue(clientId, device, devicesuite, test,
"Details", true);

string[] allDetailsList = values.Split(',');

values = mClient.GetResultsValue(clientId, device, devicesuite, test, "Value",
true);

string[] allValuesList = values.Split(',');

for(int index=0;index<allDetailsList.Length;index++)

    Console.WriteLine("Value for "+allDetailsList[index].ToString()+ " =
"+allValuesList[index].ToString
());
```

NOTE. The Fail condition for PI commands occurs in any of the following cases:

If the server is locked, the application displays "Server is locked by another client."

If the session is unlocked, the application displays "Lock session to execute the command."

If the server is not found, the application displays "Server not found-Disconnect!."

If the fail condition is not one of the above types, the application displays "Failed."

out string clientID

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| clientid | string | OUT | Identifier of the client that is connected to the server clientid = unique number + ipaddress of the client. For example, 1065-192.157.98.70 |

string device

| Name | Type | Direction | Description |
|--------|--------|-----------|--|
| device | string | IN | Specifies the name of the device. For PCIe testing, set this to "CEM" or "U.2 (SFF-8639)". |

string devicesuite

| Name | Type | Direction | Description |
|-------------|--------|-----------|--|
| devicesuite | string | IN | Specifies the name of the PCIe device suite, enclosed in quotes. For "CEM", valid values are System-Board and Add-In-Card . For "U.2 (SFF-8639)", valid values are Module and Host . |

string test

| Name | Type | Direction | Description |
|------|--------|-----------|---|
| test | string | IN | Specifies the name of the test to obtain the pass or fail status or a test result value. Append spaces at the end of the test name to differentiate the PCIe DUT generation version for which to return measurements: Gen1: no spaces Gen2: One space Gen3: Two spaces Examples: Gen1: "Unit Interval" Gen2: "Unit Interval " Gen3: "Unit Interval " |

string filePath

| Name | Type | Direction | Description |
|----------|--------|-----------|---|
| filePath | string | IN | The location where the report must be saved in the client |

NOTE. If the client does not provide the location to save the report, the report is saved at C:\ProgramFiles.

string parameterString

| Name | Type | Direction | Description |
|-----------------|--------|-----------|--|
| parameterString | string | IN | Specifies the oscilloscope model, TekExpress version, or application version |

string parameterString "Value"

| Name | Type | Direction | Description |
|-----------------|--------|-----------|--|
| parameterString | string | IN | Specifies to return the measured value for the indicated test. Enter "Value" for this argument |

Get pass/fail status for a measurement example

This example returns the pass/fail status for the **Gen1** transition eye diagram measurement:

```
returnval=m_Client.GetPassFailStatus(clientId, device, devicesuite, "Transition Eye Diagram")
```

This example returns the results for the **Gen3** unit interval measurement:

```
returnval=m_Client.GetPassFailStatus(clientId, device, devicesuite, "Unit Interval ")
```

Note the two blank spaces between the end of the measurement name and the closing quote for that parameter. The test parameter uses blank spaces at the end of the test name to differentiate the PCIe DUT generation version for which to return measurements:

Gen1: no spaces

Gen2: One space

Gen3: Two spaces

Examples:

Gen1: "Unit Interval"

Gen2: "Unit Interval "

Gen3: "Unit Interval "

Unlock the server

| Command name | Parameters | Description | Return value | Example |
|-----------------|-----------------|---|---|--|
| UnlockSession() | string clientID | This method unlocks the server from the client. The ID of the client to be unlocked must be provided. Note | String that gives the status of the operation after it was performed. The return value is "Session UnLocked..." | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string returnval=m_Client.UnlockServer(clientID) |

NOTE. When the client is disconnected, the client is unlocked automatically.

out string clientID

| Name | Type | Direction | Description |
|----------|--------|-----------|--|
| clientID | string | OUT | Identifier of the client that is connected to the server clientID = unique number + IP address of the client. For example, 1065-192.157.98.70 |

NOTE. The Fail condition for PI commands occurs in any of the following cases:

If the server is locked, the application displays "Server is locked by another client."

If the session is unlocked, the application displays "Lock session to execute the command."

If the server is not found, the application displays "Server not found-Disconnect!."

If the fail condition is not one of the above types, the application displays "Failed."

Disconnect from the server

| Command name | Parameters | Description | Return value | Example |
|--------------|-----------------|--|--|--|
| Disconnect() | string clientID | This method disconnects the client from the server. Note | Integer value that gives the status of the operation after it was performed 1 for Success -1 for Failure | m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string returnval=m_Client.Disconnect(m_clientID) |

NOTE. When the client is disconnected, it is unlocked from the server and then disconnected. The Id is reused.

out string clientID

| Name | Type | Direction | Description |
|----------|--------|-----------|--|
| clientID | string | OUT | Identifier of the client that is connected to the server clientID = unique number + IP address of the client. For example, 1065-192.157.98.70 |

NOTE. The Fail condition for PI commands occurs in any of the following cases:

If the server is locked, the application displays "Server is locked by another client."

If the session is unlocked, the application displays "Lock session to execute the command."

If the server is not found, the application displays "Server not found-Disconnect!."

If the fail condition is not one of the above types, the application displays "Failed."

SCPI commands

About SCPI command

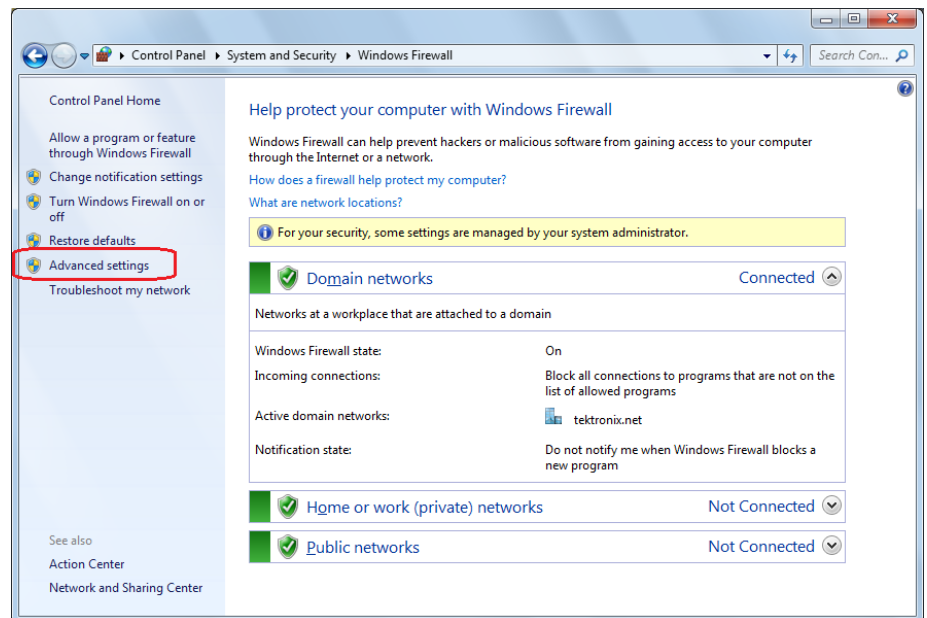
You can use Standard Commands for Programmable Instruments (SCPI) to communicate with the TekExpress application.

Socket configuration for SCPI commands

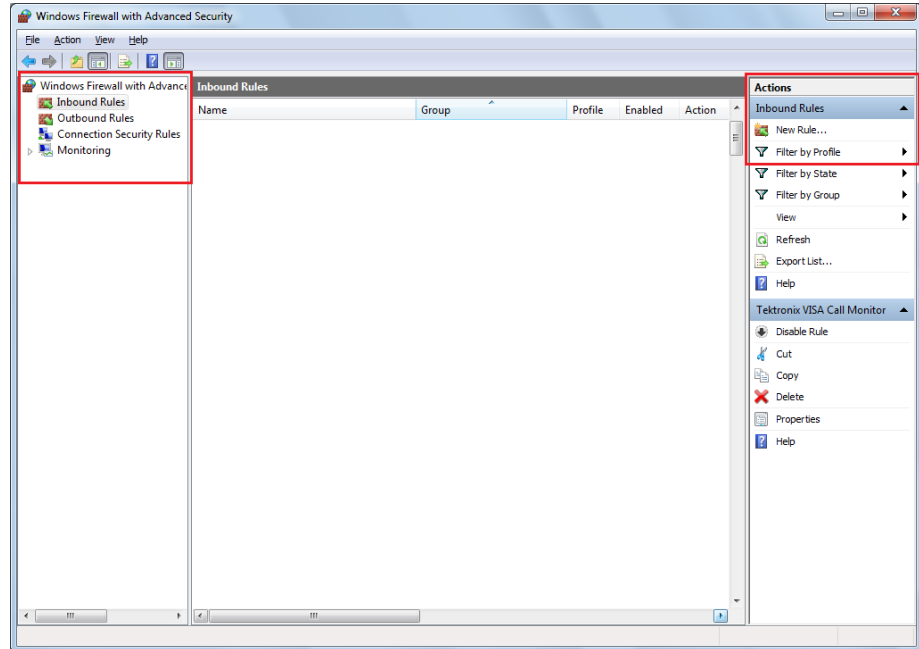
This section describes the steps for TCP/IP socket configuration and TekVISA configuration to execute the SCPI commands.

TCP/IP socket configuration

1. Click **Start > Control Panel > System and Security > Windows Firewall > Advanced settings**

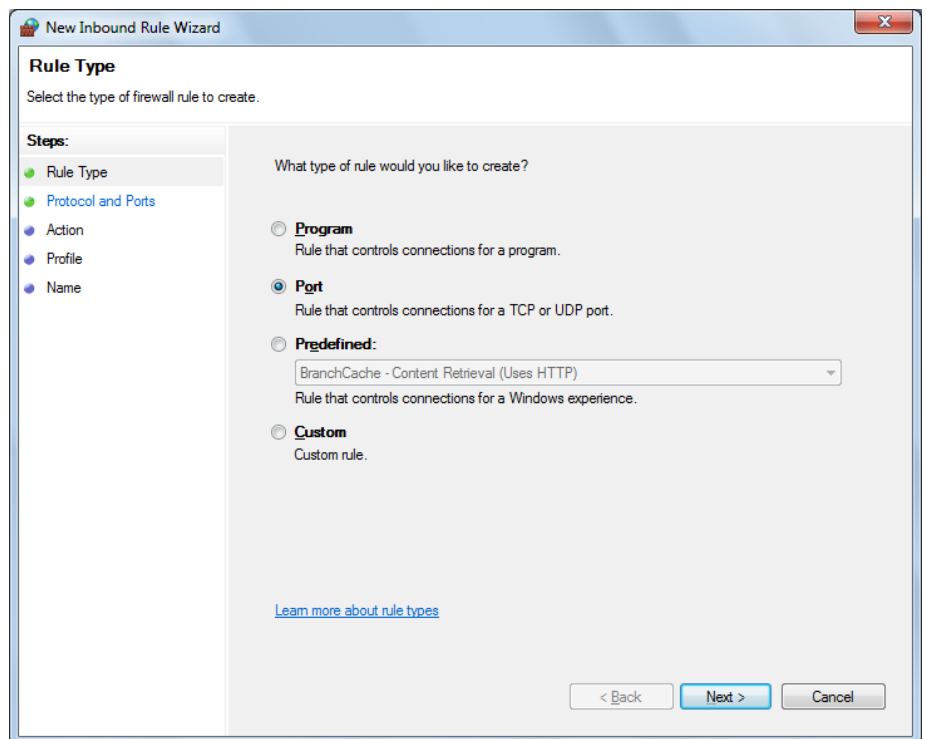


2. In Windows Firewall with Advanced Security menu, select **Windows Firewall with Advanced Security on Local Computer > Inbound Rules** and click New Rule...

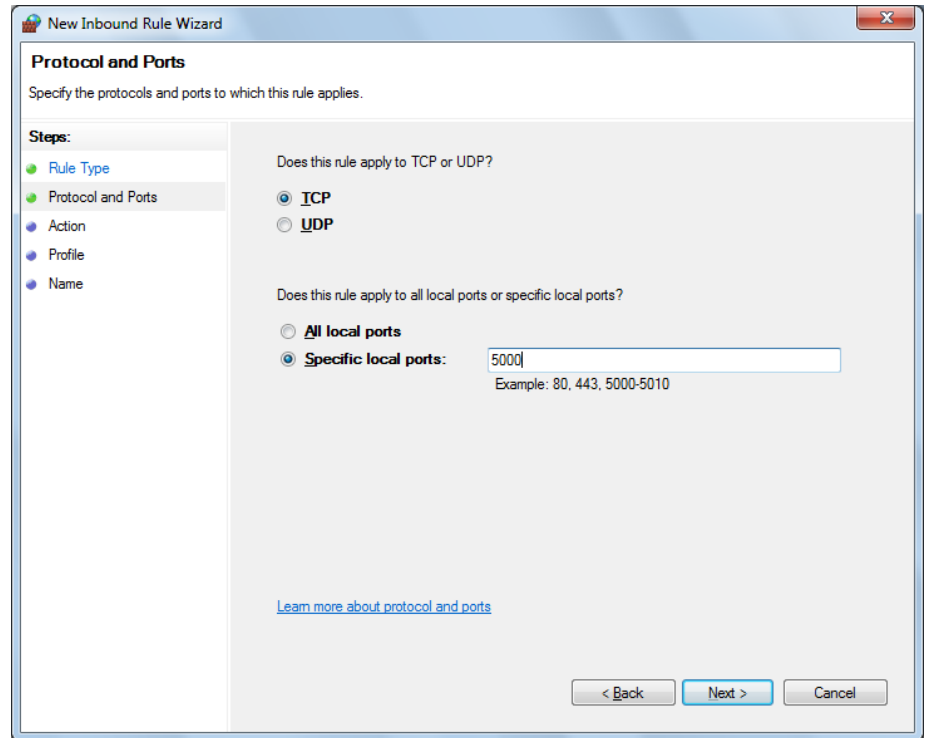


3. In New Inbound Rule Wizard menu

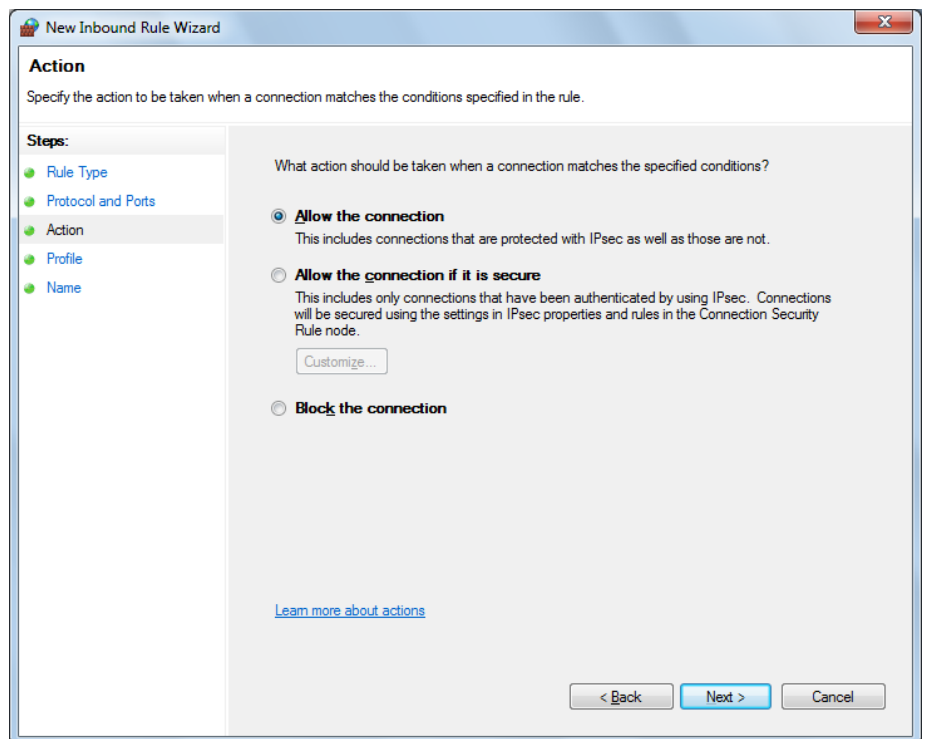
a. Select **Port** and click **Next**



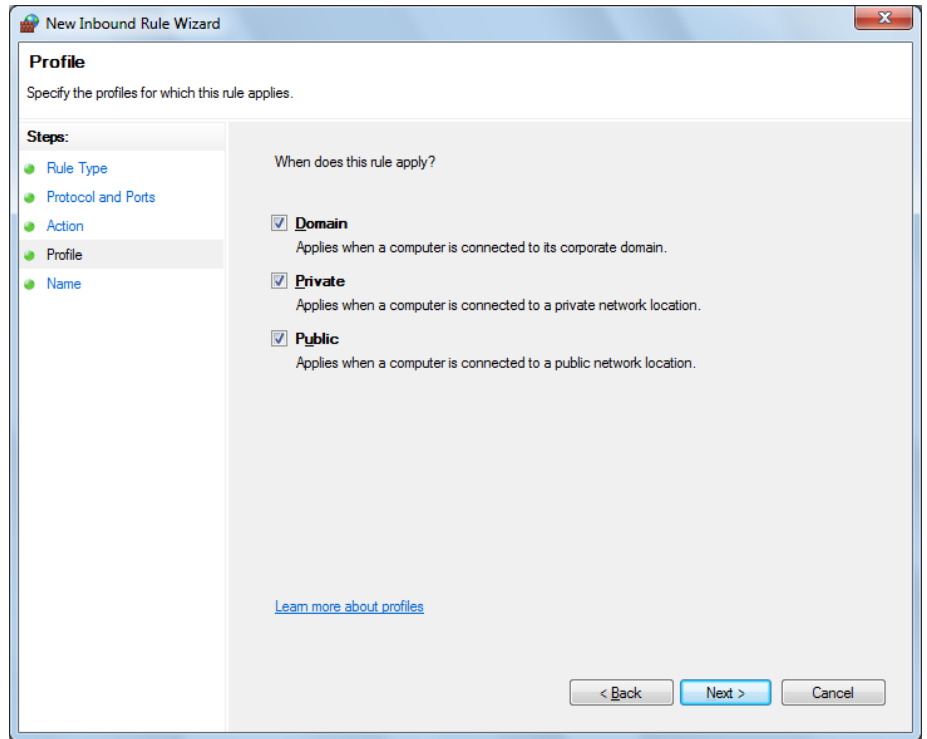
- b. Select **TCP** as rule apply and enter 5000 for **Specific local ports** and click **Next**



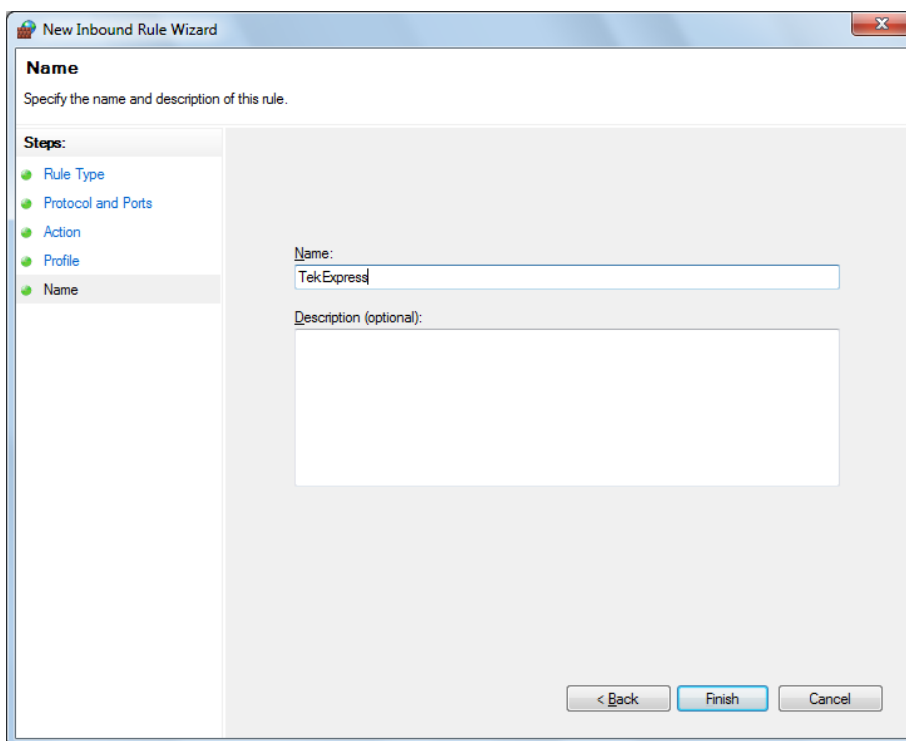
c. Select **Allow the connection** and click **Next**



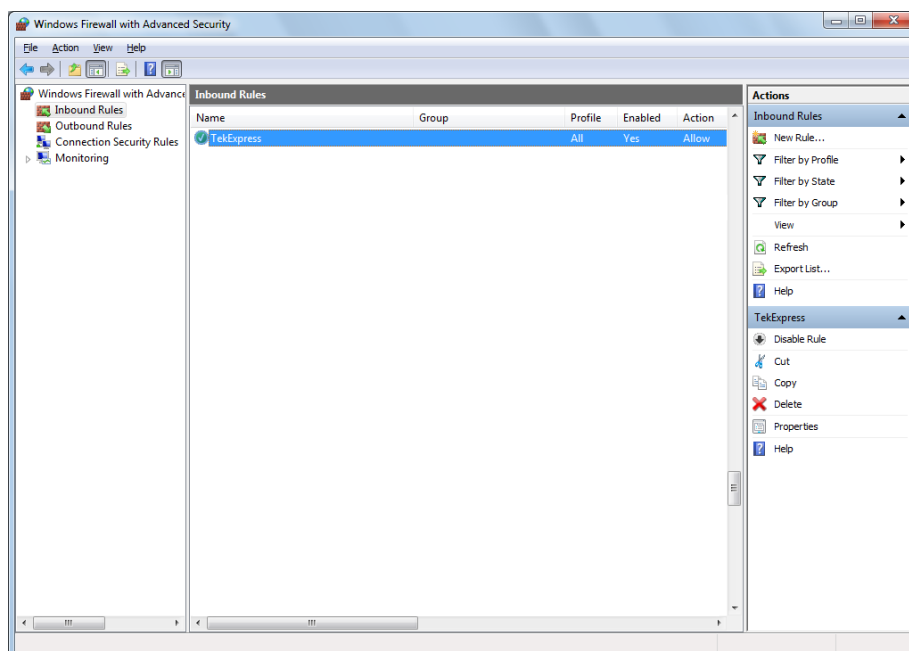
d. Select **Domain**, **Private**, **Public** and click **Next**



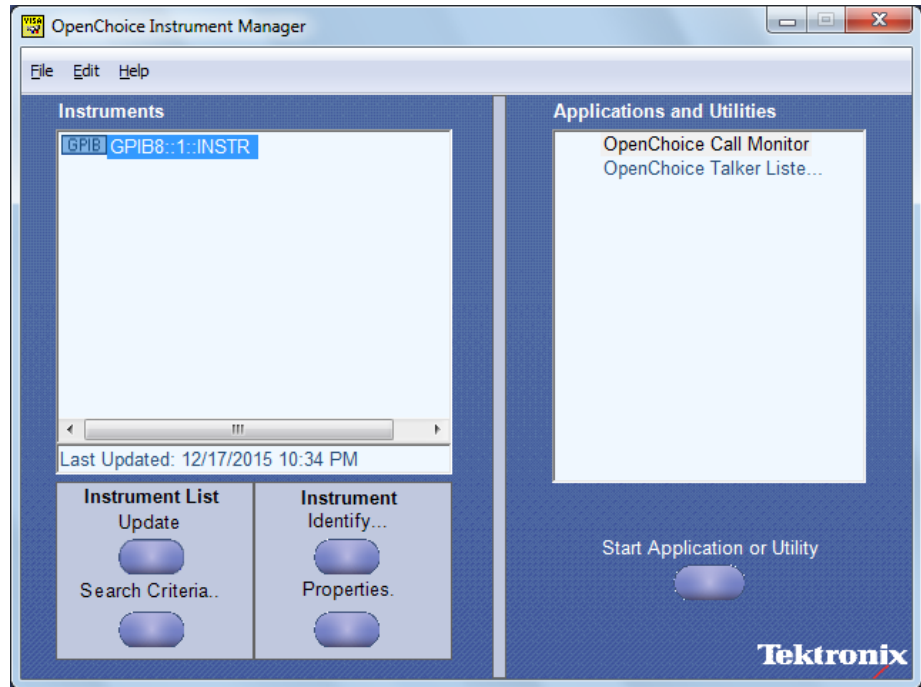
- e. Enter **Name**, Description (optional), and click **Finish**




4. Check whether the Rule name is displayed in **Windows Firewall with Advanced Security menu > Inbound Rules**



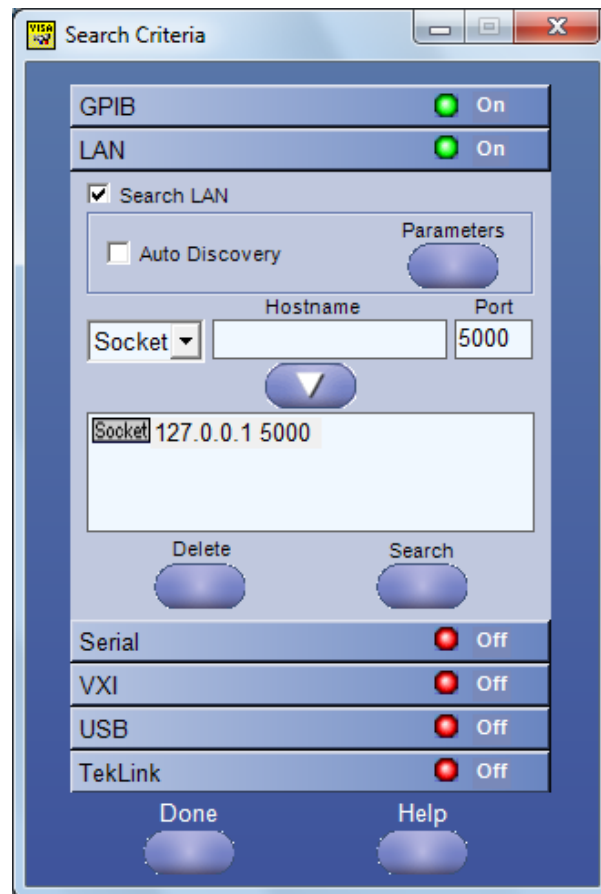
- TekVISA configuration** 1. Click **Start > All Programs > TekVISA > OpenChoice Instrument Manager**



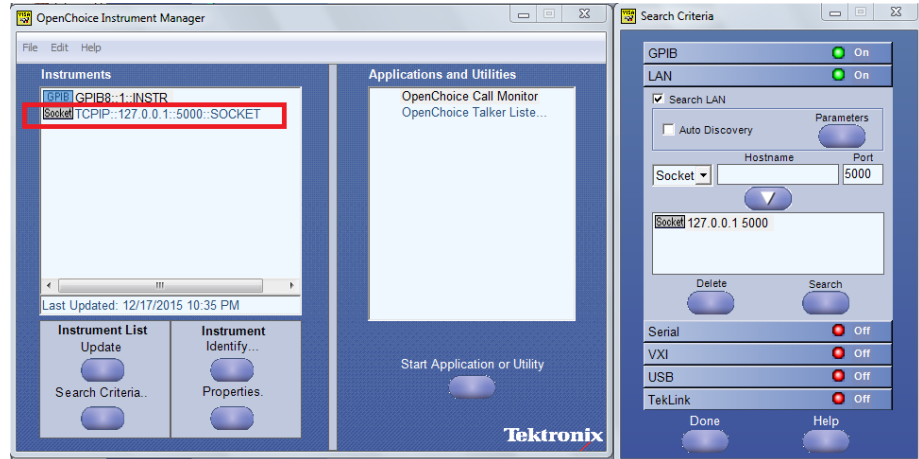
2. Click **Search Criteria**. In Search Criteria menu, click **LAN** to Turn-on. Select **Socket** from the drop-down list, enter the IP address of the

TekExpress device in **Hostname** and type **Port** as 5000. Click  to configure the IP address with Port.

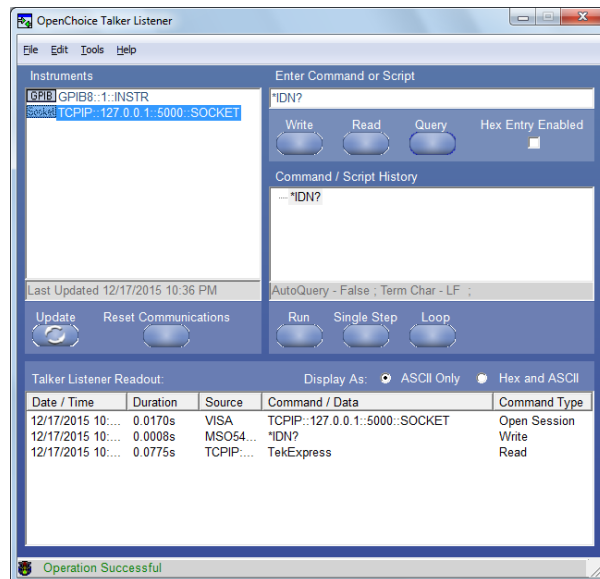
Enter the Hostname as 127.0.0.1 if the TekVISA and TekExpress application are in the same system, else enter the IP address of the TekExpress application system.



- Click **Search** to setup the TCPIP connection with the host. Check whether the TCPIP host name is displayed in **OpenChoice Instrument Manager > Instruments**



- Double-click **OpenChoice Takler Listener** and enter the Command ***IDN?** in command entry field and click **Query**. Check that the Operation is successful and Talker Listener Readout displays the Command / Data.



TEKEXP:*IDN?

This command queries the active TekExpress application name running on the scope.

Syntax TEKEXP:*IDN?\n

Inputs NA

Outputs Returns active TekExpress application name running on the scope

Example If TEKEXP:*IDN?\n Returns "TekExpApp", then TekExpApp is the name of the active TekExpress application running on the scope.

TEKEXP:*OPC?

This command queries the execution status of the last executed command.

Syntax TEKEXP:*OPC?\n

Inputs NA

Outputs 0 - last command execution is not complete
 1 - last command execution is complete

Example If `TEKEXP:*OPC?` returns 0, then the last command execution is not complete.

TEKEXP:ACQUIRE_MODE

This command sets the acquire mode as live or pre-recorded.

Syntax `TEKEXP:ACQUIRE_MODE {LIVE | PRE-RECORDED}\n`

Inputs {LIVE | PRE-RECORDED}

Outputs NA

Example `TEKEXP:ACQUIRE_MODE PRE-RECORDED\n` sets the acquire mode as pre-recorded.

TEKEXP:ACQUIRE_MODE?

This command queries the acquire mode type.

Syntax `TEKEXP:ACQUIRE_MODE?\n`

Inputs NA

Outputs {LIVE | PRE-RECORDED}

Example If `TEKEXP:ACQUIRE_MODE?` returns "LIVE", then acquire mode is set to live.

TEKEXP:EXPORT

This command returns all the bytes of data to the specified file.

| Syntax | Outputs |
|---|--|
| <code>TEKEXP:EXPORT REPORT</code> | Returns the report file in bytes |
| <code>TEKEXP:EXPORT WFM,"<FileName>"</code> | Returns the specified waveform file in bytes |
| <code>TEKEXP:EXPORT IMAGE,"<FileName>"</code> | Returns the specified image file in bytes |

Inputs FileName - Specifies the file name

Example `TEKEXP:EXPORT REPORT` returns the report file in bytes. This can be written into another file for further analysis.

NOTE. This command can be executed through scripts only as this command requires FileName parameter to be parsed from `TEKEXP:INFO?` command.

TEKEXP:INFO?

This command queries the information about the file(s).

| Syntax | Outputs |
|----------------------------------|--|
| <code>TEKEXP:INFO? REPORT</code> | <ReportFileSize>,"<ReportFileName.mht>" |
| <code>TEKEXP:INFO? WFM</code> | <WfmFile1Size>,"<WfmFileName1.wfm>";<WfmFile2Size>,"<WfmFileName2.wfm>";... |
| <code>TEKEXP:INFO? IMAGE</code> | <Image1FileSize>,"<Image1FileName>";<Image2FileSize>,"<Image2FileName>" ;... |

Example If TEKEXP:INFO? REPORT\n returns “100”, “ReportFileName.mht”, then 100 bytes is the filesize and ReportFileName is the filename.
If TEKEXP:INFO? WFM\n returns “100”, “WfmFileName1.wfm”; “200”, “WfmFileName2.wfm”.

TEKEXP:INSTRUMENT

This command sets the value for the selected instrument type.

Syntax TEKEXP:INSTRUMENT "<InstrumentType>",<Value>"\n

Inputs InstrumentType
Value



TIP. Check [Command parameters list](#) for InstrumentType and Value parameters.

Outputs NA

Example TEKEXP:INSTRUMENT "<Real Time Scope>",<111>"\n sets the instrument value as 111 for the selected instrument type XXX.

TEKEXP:INSTRUMENT?

This command queries the instrument selected for the specified instrument type.

Syntax `TEKEXP:INSTRUMENT? "<InstrumentType>"\n`

Inputs `InstrumentType`



TIP. Check [Command parameters list](#) for `InstrumentType` parameters.

Outputs Returns the instrument selected for the specified instrument type

Example If `TEKEXP:INSTRUMENT? "Real Time Scope"\n` returns "InstrumentA", then InstrumentA is the selected instrument for the instrument type Real Time Scope.

TEKEXP:LASTERROR?

This command queries the last error string occurred for the current TCP session. If there are no errors since startup, or since the last call to `TEKEXP:LASTERROR?\n`, this command returns an empty string.

Syntax `TEKEXP:LASTERROR?\n`

Inputs NA

Outputs <string>

Example If TEKEXP:LASTERROR?\n returns "ERROR: ErrorA", then ErrorA is the last error occurred.

TEKEXP:LIST?

This command queries the list of available device, suite, test, version or instrument.

| Syntax | Outputs |
|---|---|
| TEKEXP:LIST? DEVICE\n | Returns the list of available device(s) as comma separated values. |
| TEKEXP:LIST? SUITE\n | Returns the list of available suite(s) as comma separated values. |
| TEKEXP:LIST? TEST\n | Returns the list of available test(s) as comma separated values. |
| TEKEXP:LIST? VERSION\n | Returns the list of available version(s) as comma separated values. |
| TEKEXP:LIST? INSTRUMENT,"<InstrumentType>\n | Returns the list of available instruments' for the given Instrument type as comma separated values. |

Inputs InstrumentType



TIP. Check [Command parameters list](#) for InstrumentType parameters.

Example If TEKEXP:LIST? DEVICE\n returns "TX-Device,RX-Device" then TX-Device, RX-Device are the available device.

If TEKEXP:LIST? INSTRUMENT,"Real Time Scope"\n returns Instrument1, Instrument2 then Instrument1, Instrument2 are the list of available instruments.

TEKEXP:MODE

This command sets the execution mode as compliance or user defined.

Syntax `TEKEXP:MODE {COMPLIANCE | USER-DEFINED}\n`

Inputs `{COMPLIANCE | USER-DEFINED}`

Outputs `NA`

Example `TEKEXP:MODE COMPLIANCE\n` to set the execution mode as compliance

TEKEXP:MODE?

This command queries the execution mode type.

Syntax `TEKEXP:MODE?\n`

Inputs `NA`

Outputs `{COMPLIANCE | USER-DEFINED}`

Example If `TEKEXP:MODE?\n` returns `COMPLIANCE`, then the execution mode is compliance.

TEKEXP:POPUP

This command sets the response to the active popup shown in the application.

Syntax TEKEXP:POPUP "<PopupResponse>"\n

Inputs PopupResponse

Outputs NA

Example TEKEXP:POPUP "PopupResponseA"\n sets PopupResponseA as the response to active popup in the application.

TEKEXP:POPUP?

This command queries the active popup information shown in the application.

Syntax TEKEXP:POPUP?\n

Inputs NA

Outputs Returns the active popup information in the application.

Example If TEKEXP:POPUP?\n returns "PopupMessageA", then PopupMessageA is the active popup information shown in the application.

TEKEXP:REPORT

This command generates the report for the current session.

Syntax `TEKEXP:REPORT GENERATE\n`

Inputs `GENERATE`

Outputs `NA`

Example `TEKEXP:REPORT GENERATE\n` generates report for the current session.

TEKEXP:REPORT?

This command queries the queried header field value in the report.

Syntax `TEKEXP:REPORT? "<HeaderField>"\n`

Inputs `HeaderField` - Specifies to return the measured value for the indicated test.



TIP. Check **Report** for *HeaderField* parameters.

Outputs Returns the queried header field value in the report

Example If TEKEXP:REPORT? “Master Scope Information”\n returns "DPO73304SX", then DPO73304SX is the scope model.

If TEKEXP:REPORT? “DUT ID”\n returns "DNI_DUTID", then DNI_DUTID is the DUT ID.

TEKEXP:RESULT?

This command queries the result available in report summary/details table.

| Syntax | Outputs |
|---|--|
| TEKEXP:RESULT? "<TestName>"\n | Return Pass/Fail status of the test. |
| TEKEXP:RESULT? "<TestName>",<ColumnName>"\n | Returns all the row values of the specified column for the test. |
| TEKEXP:RESULT? "<TestName>",<ColumnName>,<RowNumber>"\n | Returns the column value for the specified row number. |

Inputs

- TestName - Specifies the name of the test for which to obtain the test result value.
- ColumnName - Specifies the column name for the measurement
- RowNumber - Specifies the row number of the measurement



TIP. Check *Report* for TestName, ColumnName, and RowNumber parameters.

Example If TEKEXP:RESULT? "Period using SCOPE (Acquire-Analyze Combined)"\n returns Pass, then the test result is Pass.

If TEKEXP:RESULT? "Period using SCOPE (Acquire-Analyze Combined)","Margin",1\n returns "L:-50.000ps H:2000.000ps", then L:-50.000ps H:2000.000ps is the value.

If TEKEXP:RESULT? "Measured Value","Clock Frequency",1\n returns "2.9600413", then 2.9600413 is the value.

TEKEXP:SELECT

This command selects the device, suite, version, or test.

Syntax `TEKEXP:SELECT <string1>,<string2>,<string4>\n`
`TEKEXP:SELECT TEST,<string3>,<string4>\n`

Inputs `<string1> = {DEVICE | SUITE | VERSION}`
`<string2> = {DeviceName | SuiteName | VersionName}`
`<string3> = {"<TestName>" | ALL | REQUIRED }`
`<string4> = {TRUE | FALSE}`



TIP. Check [Command parameters list](#) for *DeviceName*, *SuiteName*, *VersionName*, and *TestName* parameters.

Outputs NA

Example `TEKEXP:SELECT DEVICE, DeviceA, "TestNameA"\n` selects DeviceA, TestNameA.

TEKEXP:SELECT?

This command queries the name of the selected device, suite, version, or test.

Syntax TEKEXP:SELECT? {DEVICE | SUITE | TEST | VERSION}\n

Inputs {DEVICE | SUITE | TEST | VERSION}

Outputs Returns the name of the selected device, suite, version, or test.

Example If TEKEXP:SELECT? DEVICE\n returns "TX-Device", then TX-Device is the type of the selected device.

TEKEXP:SETUP

This command sets the value of the current setup.

| Syntax | Outputs |
|-------------------------------------|--------------------------|
| TEKEP:SETUP DEFAULT\n | Restore to default Setup |
| TEKEP:SETUP OPEN,"<SessionName>"\n | Open the session |
| TEKEXP:SETUP SAVE\n | Save the session |
| TEKEXP:SETUP SAVE,"<SessionName>"\n | Save the session |

Inputs SessionName - The name of the session

Example TEKEP:SETUP DEFAULT\n restore to default setup.

TEKEXP:STATE

This command sets the execution state of the application.

Syntax `TEKEXP:STATE {RUN | STOP | PAUSE | RESUME}\n`

Inputs `{RUN | STOP | PAUSE | RESUME}`

Outputs `NA`

Example `TEKEXP:STATE STOP\n` sets the execution state of the application to stop.

TEKEXP:STATE?

This command queries the current setup state.

| Syntax | Outputs |
|----------------------------------|--|
| <code>TEKEXP:STATE?</code> | <code>RUNNING PAUSED WAIT ERROR READY STOPPED</code> |
| <code>TEKEXP:STATE? SETUP</code> | <code>SAVED NOT_SAVED</code> |

Example If `TEKEXP:STATE?\n` returns as `READY`, then the application is ready to run next measurement.

If `TEKEXP:STATE? SETUP\n` returns as `NOT_SAVED`, then the current setup is not saved.

TEKEXP:VALUE

This command sets the value of parameters of type General, Acquire, Analyze, or DUTID.

Syntax `TEKEXP:VALUE GENERAL,"<ParameterName>","<Value>"\n`
`TEKEXP:VALUE ACQUIRE,"<AcquireType>", "<ParameterName>",`
`"<Value>"\n`
`TEKEXP:VALUE ANALYZE,"<ParameterName>" "<Value>"\n`
`TEKEXP:VALUE DUTID,"<Value>"\n`

Inputs `ParameterName` - Specifies the parameter name
`AcquireType` - Specifies the acquire type
`Value` - Specifies the value to set



TIP. Check [Command parameters list](#) for `ParameterName`, `AcquireType`, and `Value` parameters.

Outputs NA

Example `TEKEXP:VALUE GENERAL,"Signal Type"\n` sets the parameter name as Signal Type for GENERAL.

TEKEXP:VALUE?

This command queries the value of the parameter for type General, Acquire, Analyze, or DUTID.

| Syntax | Outputs |
|--|--|
| TEKEXP:VALUE? GENERAL,"<ParameterName>"\n | Returns the value of Parameter for type GENERAL |
| TEKEXP:VALUE? ACQUIRE,"<AcquireType>","<ParameterName>" \n | Returns the value of Parameter for type ACQUIRE |
| TEKEXP:VALUE? ANALYZE,"<ParameterName>"\n | Returns the value of Parameter for type ANALYZE |
| TEKEXP:VALUE? DUTID\n | Returns the DUTID value |

Inputs ParameterName - Specifies the parameter name
AcquireType - Specifies the acquire type



TIP. Check [Command parameters list](#) for ParameterName and AcquireType parameters.

Outputs Returns the value of Parameter for type GENERAL | ACQUIRE | ANALYZE | DUTID.

Example If TEKEXP:VALUE? GENERAL,"Signal Type"\n returns "N1N0", then N1N0 is the value of parameter Signal Type for GENERAL.

Command parameters list

This section provides the parameters list for the SCPI commands.

| Parameters | Description |
|----------------|--|
| InstrumentType | Specifies the instrument type. Valid values are: <ul style="list-style-type: none"> ■ Alternate Real Time Scope ■ Real Time Scope |
| Value | Specifies the value parameters. <ul style="list-style-type: none"> ■ For InstrumentType, valid values are: <ul style="list-style-type: none"> ■ Comment ■ For DUTID, valid values are: <ul style="list-style-type: none"> ■ Comment |
| DeviceName | Specifies the device name. Valid values are CEM, U.2(SFF-8639) |
| SuiteName | Specifies the suite name. For CEM, valid values are System-Board, Add-In-Card For U.2(SFF-8639), valid values are Host, Module |
| VersionName | Specifies the version name. Valid values are: <ul style="list-style-type: none"> ■ Gen1-1.0a (Applicable only for DeviceName = CEM) ■ Gen1-1.1 (Applicable only for DeviceName = CEM) ■ Gen2-2.0 (Applicable only for DeviceName = CEM) ■ Gen3-3.0 |
| TestName | Specifies the test measurement name. <ul style="list-style-type: none"> ■ Unit Interval ■ Mask Hits(All Bits) ■ Composit Eye Height ■ Transition Eye Diagram ■ Non Transition Eye Diagram ■ Min Eye Width ■ Min Time Between Crossovers ■ TJ @ E-12 ■ Dj_dd ■ RJ(RMS) ■ Peak to Peak Jitter |

ParameterName and Value for General

Specifies the ParameterName and Value for General. The configuration parameters available are not same for measurements.

Table 12: ParameterName and Value for General

| ParameterName | Value |
|----------------|--|
| DataRate2Gb | <ul style="list-style-type: none"> ■ Included ■ Excluded |
| DataRate5Gb | <ul style="list-style-type: none"> ■ Included ■ Excluded |
| DataRate8Gb | <ul style="list-style-type: none"> ■ Included ■ Excluded |
| PreEmphasis3dB | <ul style="list-style-type: none"> ■ Included ■ Excluded |
| PreEmphasis6dB | <ul style="list-style-type: none"> ■ Included ■ Excluded |
| SSC | <ul style="list-style-type: none"> ■ On ■ Off |
| VoltageSwing | <ul style="list-style-type: none"> ■ Full ■ Reduced |
| Link Widths | 1 Lane |
| SignalPreset | P0 For multiple signal preset, specify as P0_P1_P2 |
| Preset | P0 For multiple preset, specify as P0_P1_P2 |
| Acquisition | BeforeAnalysis |
| SaveOptions | Save All the Waveforms |
| DeEmbed2Gb | <ul style="list-style-type: none"> ■ Included ■ Excluded |
| DeEmbed5Gb | <ul style="list-style-type: none"> ■ Included ■ Excluded |
| DeEmbed8Gb | <ul style="list-style-type: none"> ■ Included ■ Excluded |

| ParameterName | Value |
|---------------------------------|--|
| Embed8Gb | <ul style="list-style-type: none"> ■ Included ■ Excluded |
| Equalization8Gb | <ul style="list-style-type: none"> ■ Included ■ Excluded |
| Filterfile2Gb | Filterfile2Gb.ftt |
| Filterfile5Gb | Filterfile5Gb.ftt |
| FilterfileDeEmbed8Gb | FilterfileDeEmbed8Gb.ftt |
| FilterfileEmbed8Gb | FilterfileEmbed8Gb.ftt |
| EmbedDropdown | <ul style="list-style-type: none"> ■ SigTest ■ Scope |
| EqualizationDropdown | <ul style="list-style-type: none"> ■ Optimize ■ Fixed |
| CTLE Index | 1 : -12dB |
| DFE | -30e-3 |
| EnableDUTAutomation | <ul style="list-style-type: none"> ■ Included ■ Excluded |
| Automation Settings | Use Default Settings |
| Signal Type | <ul style="list-style-type: none"> ■ Square ■ Sine |
| RecordLength2Gb | 2.5e6 |
| RecordLength5Gb | 10e6 |
| RecordLength8Gb | 10e6 |
| SampleRate2Gb | 50e9 |
| SampleRate5Gb | 50e9 |
| SampleRate8Gb | 50e9 |
| Bandwidth2Gb | 7e9 |
| Bandwidth5Gb | 12.5e9 |
| Bandwidth8Gb | 12.5e9 |
| Signal Validation | Turn Off Signal Check |
| SlotNumber | 05 |
| Toggle from Gen3P10 to Gen1 | <ul style="list-style-type: none"> ■ On ■ Off |
| Signal Validation Threshold(mV) | 200 |

| ParameterName | Value |
|--|---|
| AnalysisMode | <ul style="list-style-type: none"> ■ DLL ■ CLI |
| Report Update Mode | <ul style="list-style-type: none"> ■ New ■ Append ■ Replace |
| Auto increment report name if duplicate | TRUE or FALSE |
| Include Pass/Fail Results Summary | TRUE or FALSE |
| Include Detailed Results | TRUE or FALSE |
| Include Plot Images | TRUE or FALSE |
| Include Setup Configuration | TRUE or FALSE |
| Include Complete Application Configuration | TRUE or FALSE |
| Include User Comments | TRUE or FALSE |
| Save As Type | <ul style="list-style-type: none"> ■ Web Archive (*.mht;*.mhtml) ■ PDF (*.pdf;) |
| View Report After Generating | TRUE or FALSE |
| Report Group Mode | <ul style="list-style-type: none"> ■ Test Name ■ Lane Name ■ Test Result ■ Equalization |
| Create report at the end | <ul style="list-style-type: none"> ■ Included ■ Excluded |
| DUTID Comment | User comment |
| Number of retries for instrument IO errors | 0 to 5 |
| Run Test More than Once | TRUE or FALSE |
| Number of Runs | 1 to 100 |
| On Failure Rerun | TRUE or FALSE |
| Number of Reruns On Failure | 1 to 100 |
| Time between retries (seconds) | 5 to 60 |
| Timer Warning Info Message Popup | <ul style="list-style-type: none"> ■ "True" ■ "FALSE" |

SCPI commands

| ParameterName | Value |
|---|--|
| Timer Warning Info Message Popup Duration | 0 to 20 |
| Timer Error Message Popup | <ul style="list-style-type: none">■ "True"■ "False" |
| Timer Error Message Popup Duration | 0 to 20 |
| On Failure Stop and Notify | TRUE or FALSE |

Reference

De-embed using filter files

TekExpress PCIe provides an option to de-embed the signal path using filter files. You create the filter files. The filter files are .flt files composed of de-embed filter coefficients for a particular sampling rate. A filter file created for one sampling rate might not work for other sampling rates, so it is important to understand at what sampling rate the measurements are being performed.

Also, the de-embedding filters might differ based on the type of input. For example, if a single ended input is made using a matched SMA cable pair, a filter file for de-embedding a single SMA cable must be provided, since matched SMA cables mostly have similar s-parameters. So in this case, the same filter file is used to de-embed the SMA cable pair.

The maximum sampling rate provided on any channel combination on MSO/DPO/DSA70000/C/D/DX series oscilloscopes is 50 GS/s in realtime mode. The maximum sampling rate provided on Ch1-Ch3 and Ch2-Ch4 channel combinations on MSO/DPO/DSA70000C/D/DX series oscilloscopes is 100 GS/s, provided only 2 channels are on at a given time.

See also

[*Common test parameters and values*](#)

[*Configuration test parameters*](#)

Setup files

TekExpress PCI Express package contains setup files (*.TekX) which can be used at PCIE Gen3 workshop for compliance tests.

Table 13: Setup files configuration details

| Setup files (*.TekX) | Configuration details (exclusively used in Gen1/2/3 Gold Suite of PCI-SIG Work Shop (WS)) |
|----------------------|---|
| WorkShop_CEM_AIC_x1 | <ul style="list-style-type: none"> ■ Specification - CEM ■ Device Type - Add-In-Card ■ Version - Gen3 - 3.0 ■ Data Rates - 2.5 Gbps, 5 Gbps (Tx equalization 3.5dB, 6 dB) and 8 Gbps ■ Signal Quality Preset Selection - P0, P7 and P8 for L0 and All Presets from P0 to P10 for Lane0 ■ Link Width - 1 Lane (Selected test lane: L0) ■ Automated DUT Control - unchecked ■ Signal Validation - Pattern Decoding ■ SigTest Interface Mode: CMD line - SigTest displayed and its HTML report saved with TE Report |
| WorkShop_CEM_AIC_x2 | <ul style="list-style-type: none"> ■ Specification - CEM ■ Device Type - Add-In-Card ■ Version - Gen3 - 3.0 ■ Data Rates - 2.5 Gbps, 5 Gbps (Tx equalization 3.5dB, 6 dB) and 8 Gbps ■ Signal Quality Preset Selection - P0, P7 and P8 for L0 and All Presets from P0 to P10 for Lane0 ■ Link Width - 2 Lanes (Selected test lane: L0) ■ Automated DUT Control - unchecked ■ Signal Validation - Pattern Decoding ■ SigTest Interface Mode: CMD line - SigTest displayed and its HTML report saved with TE Report |
| WorkShop_CEM_AIC_x4 | <ul style="list-style-type: none"> ■ Specification - CEM ■ Device Type - Add-In-Card ■ Version - Gen3 - 3.0 ■ Data Rates - 2.5 Gbps, 5 Gbps (Tx equalization 3.5dB, 6 dB) and 8 Gbps ■ Signal Quality Preset Selection - P0, P7 and P8 for L0 and All Presets from P0 to P10 for Lane0 ■ Link Width - 4 Lanes (Selected test lane: L0, L03) ■ Automated DUT Control - unchecked ■ Signal Validation - Pattern Decoding ■ SigTest Interface Mode: CMD line - SigTest displayed and its HTML report saved with TE Report |

| Setup files (*.TekX) | Configuration details (exclusively used in Gen1/2/3 Gold Suite of PCI-SIG Work Shop (WS)) |
|----------------------|---|
| WorkShop_CEM_AIC_x8 | <ul style="list-style-type: none"> ■ Specification - CEM ■ Device Type - Add-In-Card ■ Version - Gen3 - 3.0 ■ Data Rates - 2.5 Gbps, 5 Gbps (Tx equalization 3.5dB, 6 dB) and 8 Gbps ■ Signal Quality Preset Selection - P0, P7 and P8 for L0 and All Presets from P0 to P10 for Lane0 ■ Link Width - 8 Lanes (Selected test lane: L0, L03, L07) ■ Automated DUT Control - unchecked ■ Signal Validation - Pattern Decoding ■ SigTest Interface Mode: CMD line - SigTest displayed and its HTML report saved with TE Report |
| WorkShop_CEM_AIC_x16 | <ul style="list-style-type: none"> ■ Specification - CEM ■ Device Type - Add-In-Card ■ Version - Gen3 - 3.0 ■ Data Rates - 2.5 Gbps, 5 Gbps (Tx equalization 3.5dB, 6 dB) and 8 Gbps ■ Signal Quality Preset Selection - P0, P7 and P8 for L0 and All Presets from P0 to P10 for Lane0 ■ Link Width - 16 Lanes (Selected test lane: L0, L07, L15) ■ Automated DUT Control - unchecked ■ Signal Validation - Pattern Decoding ■ SigTest Interface Mode: CMD line - SigTest displayed and its HTML report saved with TE Report |
| WorkShop_CEM_SYB_x1 | <ul style="list-style-type: none"> ■ Specification - CEM ■ Device Type - Add-In-Card ■ Version - Gen3 - 3.0 ■ Data Rates - 2.5 Gbps, 5 Gbps (Tx equalization 3.5dB, 6 dB) and 8 Gbps ■ Signal Quality Preset Selection - P0, P7 and P8 for L0 and All Presets from P0 to P10 for Lane0 ■ Link Width - 1 Lane (Selected test lane: L0) ■ Automated DUT Control - unchecked ■ Signal Validation - Pattern Decoding ■ SigTest Interface Mode: CMD line - SigTest displayed and its HTML report saved with TE Report |

| Setup files (*.TekX) | Configuration details (exclusively used in Gen1/2/3 Gold Suite of PCI-SIG Work Shop (WS)) |
|----------------------|--|
| WorkShop_CEM_SYB_x2 | <ul style="list-style-type: none"> ■ Specification - CEM ■ Device Type - Add-In-Card ■ Version - Gen3 - 3.0 ■ Data Rates - 2.5 Gbps, 5 Gbps (Tx equalization 3.5dB, 6 dB) and 8 Gbps ■ Signal Quality Preset Selection - P0, P7 and P8 for L0 and All Presets from P0 to P10 for Lane0 ■ Link Width - 2 Lanes (Selected test lane: L0) ■ Automated DUT Control - unchecked ■ Signal Validation - Pattern Decoding ■ SigTest Interface Mode: CMD line - SigTest displayed and its HTML report saved with TE Report |
| WorkShop_CEM_SYB_x4 | <ul style="list-style-type: none"> ■ Specification - CEM ■ Device Type - Add-In-Card ■ Version - Gen3 - 3.0 ■ Data Rates - 2.5 Gbps, 5 Gbps (Tx equalization 3.5dB, 6 dB) and 8 Gbps ■ Signal Quality Preset Selection - P0, P7 and P8 for L0 and All Presets from P0 to P10 for Lane0 ■ Link Width - 4 Lanes (Selected test lane: L0, L03) ■ Automated DUT Control - unchecked ■ Signal Validation - Pattern Decoding ■ SigTest Interface Mode: CMD line - SigTest displayed and its HTML report saved with TE Report |
| WorkShop_CEM_SYB_x8 | <ul style="list-style-type: none"> ■ Specification - CEM ■ Device Type - Add-In-Card ■ Version - Gen3 - 3.0 ■ Data Rates - 2.5 Gbps, 5 Gbps (Tx equalization 3.5dB, 6 dB) and 8 Gbps ■ Signal Quality Preset Selection - P0, P7 and P8 for L0 and All Presets from P0 to P10 for Lane0 ■ Link Width - 8 Lanes (Selected test lane: L0,L03,L07) ■ Automated DUT Control - unchecked ■ Signal Validation - Pattern Decoding ■ SigTest Interface Mode: CMD line - SigTest displayed and its HTML report saved with TE Report |

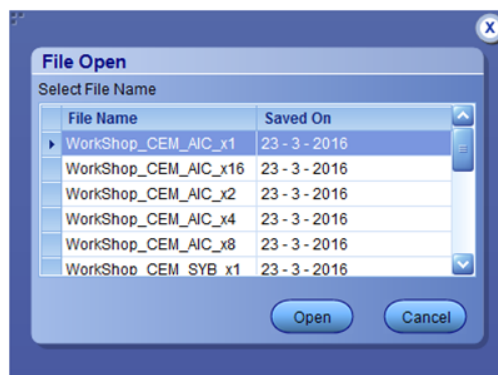
| Setup files (*.TekX) | Configuration details (exclusively used in Gen1/2/3 Gold Suite of PCI-SIG Work Shop (WS)) |
|-----------------------|---|
| WorkShop_CEM_SYB_x16 | <ul style="list-style-type: none"> ■ Specification - CEM ■ Device Type - Add-In-Card ■ Version - Gen3 - 3.0 ■ Data Rates - 2.5 Gbps, 5 Gbps (Tx equalization 3.5dB, 6 dB) and 8 Gbps ■ Signal Quality Preset Selection - P0, P7 and P8 for L0 and All Presets from P0 to P10 for Lane0 ■ Link Width - 16 Lanes (Selected test lane: L0,L07,L15) ■ Automated DUT Control - unchecked ■ Signal Validation - Pattern Decoding ■ SigTest Interface Mode: CMD line - SigTest displayed and its HTML report saved with TE Report |
| WorkShop_U2_Module_x1 | <ul style="list-style-type: none"> ■ Specification - U.2 (SFF8639) ■ Device Type - Module ■ Version - Gen3 - 3.0 ■ Data Rates - 8 Gbps ■ Signal Quality Preset Selection - P0, P7 and P8 for L0 and All Presets from P0 to P10 for Lane0 ■ Link Width - 1 Lane (Selected test lane: L0) ■ Automated DUT Control - unchecked ■ Signal Validation - Pattern Decoding ■ SigTest Interface Mode: CMD line - SigTest displayed and its HTML report saved with TE Report |
| WorkShop_U2_Module_x2 | <ul style="list-style-type: none"> ■ Specification - U.2 (SFF8639) ■ Device Type - Module ■ Version - Gen3 - 3.0 ■ Data Rates - 8 Gbps ■ Signal Quality Preset Selection - P0, P7 and P8 for L0 and All Presets from P0 to P10 for Lane0 ■ Link Width - 2 Lanes (Selected test lane: L0) ■ Automated DUT Control - unchecked ■ Signal Validation - Pattern Decoding ■ SigTest Interface Mode: CMD line - SigTest displayed and its HTML report saved with TE Report |

| Setup files (*.TekX) | Configuration details (exclusively used in Gen1/2/3 Gold Suite of PCI-SIG Work Shop (WS)) |
|-----------------------|--|
| WorkShop_U2_Module_x4 | <ul style="list-style-type: none"> ■ Specification - U.2 (SFF8639) ■ Device Type - Module ■ Version - Gen3 - 3.0 ■ Data Rates - 8 Gbps ■ Signal Quality Preset Selection - P0, P7 and P8 for L0 and All Presets from P0 to P10 for Lane0 ■ Link Width - 4 Lanes (Selected test lane: L0,L03) ■ Automated DUT Control - unchecked ■ Signal Validation - Pattern Decoding ■ SigTest Interface Mode: CMD line - SigTest displayed and its HTML report saved with TE Report |
| WorkShop_U2_Host_x1 | <ul style="list-style-type: none"> ■ Specification - U.2 (SFF8639) ■ Device Type - Host ■ Version - Gen3 - 3.0 ■ Data Rates - 8 Gbps ■ Signal Quality Preset Selection - P0, P7 and P8 for L0 and All Presets from P0 to P10 for Lane0 ■ Link Width - 1 Lane (Selected test lane: L0) ■ Automated DUT Control - unchecked ■ Signal Validation - Pattern Decoding ■ SigTest Interface Mode: CMD line - SigTest displayed and its HTML report saved with TE Report |

| Setup files (*.TekX) | Configuration details (exclusively used in Gen1/2/3 Gold Suite of PCI-SIG Work Shop (WS)) |
|----------------------|---|
| WorkShop_U2_Host_x2 | <ul style="list-style-type: none"> ■ Specification - U.2 (SFF8639) ■ Device Type - Host ■ Version - Gen3 - 3.0 ■ Data Rates - 8 Gbps ■ Signal Quality Preset Selection - P0, P7 and P8 for L0 and All Presets from P0 to P10 for Lane0 ■ Link Width - 2 Lanes (Selected test lane: L0) ■ Automated DUT Control - unchecked ■ Signal Validation - Pattern Decoding ■ SigTest Interface Mode: CMD line - SigTest displayed and its HTML report saved with TE Report |
| WorkShop_U2_Host_x4 | <ul style="list-style-type: none"> ■ Specification - U.2 (SFF8639) ■ Device Type - Host ■ Version - Gen3 - 3.0 ■ Data Rates - 8 Gbps ■ Signal Quality Preset Selection - P0, P7 and P8 for L0 and All Presets from P0 to P10 for Lane0 ■ Link Width - 4 Lanes (Selected test lane: L0, L03) ■ Automated DUT Control - unchecked ■ Signal Validation - Pattern Decoding ■ SigTest Interface Mode: CMD line - SigTest displayed and its HTML report saved with TE Report |

How to open a setup file

1. Click **Options > Open Test Setup**
2. Select the TekExpress Setup File as per your required configuration. Check Setup files configuration details table for configuration details.



3. Make the configuration details and start the test execution.

4. Click **Save Test Setup As** and save the setup.

NOTE. *You cannot edit the TekExpress Test Setup files as they are in **Read Only** mode.*

NOTE. *The setup files path is C:\Program Files (x86)\Tektronix\TekExpress\TekExpress PCI Express\Setup Files*

Index

8 Gbps preset testing dependencies, 99

A

Acquire live waveforms, 25
Acquire parameters
 including in test reports, 47
 viewing in reports, 49
Acquisition tab, 32
Acquisitions tab
 refresh channel sources info, 36
 set channel sources, 36
 set waveform save options, 34
 set waveform source (prerecorded files), 35
 setAcquire options, 33
 view detected probes, 36
Activate the license, 9
Advanced View, 51
Application version, 9
ApplicationStatus(), 166
Automate RF switch, 32

B

Bandwidth, 38

C

CheckSessionSaved(), 162
Code example, remote access, 66
Command buttons, 15
Comments, 25
Compliance Mode, 38, 51
Compliance View, 51
Configuration parameters, 38
Configuring email notifications, 21
Connect(), 75
Connected instruments
 searching for, 19
Connection requirements, 51

D

De-embedding, 207

Detailed log view, 41
Device parameters, 25
Device profile connections, 51
Device profiles, 25
Disable Popups command, 77
Disconnect(), 176
DPOJET Plug-In, 5
DUT ID, 25
DUT parameters, 25
DUT type
 10GBase-KR, 25
 40GBase-KR4, 25

E

Email notifications, 21
Email settings, 20
Enable remote access, 60
Equipment setup, 51
Evaluation mode, 13
Exiting the application, 14

F

File name extensions, 12
Filter files, 207
Firewall (remote access), 60
Free trials, 13, 51

G

GetCurrentStateInfo(), 166
GetDutId(), 79
GetPassFailStatus(), 169
GetReportParameter(), 170
GetResultsValue(), 169
GetTimeOut(), 164
Global settings, 38

H

Help conventions, 2
Host tests, 51

I

- Inbound Rule Wizard (remote access), 60
- Installing the software
 - DPOJET Plug-In, 5
 - TekExpress application for PCIe, 5
- Instruments
 - discovering connected, 18
 - viewing connected, 19
- Instruments detected, 38
- Interface, 59
- Interface error codes, 160

K

- Keep On Top, 13
- Key, 13

L

- lanes, 32
- Lanes
 - number of, 25
- License, 13
- License activation, 9
- License agreement, 9
- Limits Editor, 38
- Loading a test setup, 57
- LockSession(), 76
- Log view
 - save file, 41
- Log view tab, 41

M

- Measurement limits, 38
- Menus, 16
- Mode
 - Compliance, 38
 - User Defined, 38
- Module tests, 51
- My TekExpress folder
 - files stored in, 45
 - location of, 7

- mapping, 8

N

- New Inbound Rule Wizard, 60
- Notifications (test failure), 40

O

- Opening a saved test setup, 57
- Options menu
 - Instrument control settings, 18
 - Keep On Top, 13
- Oscilloscopes supported, 4
- Overall test result, 43

P

- Panels, 22
- Pass/Fail summary
 - viewing, 49
- Pass/Fail Summary
 - including in reports, 48
- PI cmnds
 - 5 Gb/s preemphasis, 88
 - acquisition mode, 101
 - AFG signal amplitude, 132
 - AFG signal frequency, 130
 - analysis mode, 103
 - append report mode, 110
 - application status, 165
 - bandwidth parameter, 138
 - burst count (AFG), 133
 - crosstalk, 112
 - data rate, 83
 - DUT auto toggle, 126
 - DUT auto toggle options, 127
 - get current state info, 165
 - group test results, 149
 - lane source, 96
 - link analysis CTLE index, 123
 - link analysis DFE, 124
 - link analysis embed/de-embed filter file, 117
 - link analysis embed/de-embed signal, 115

- link analysis equalization dropdown, 121
- link analysis other filter file, 119
- on failure action, 107
- PCIe test version, 82
- preset, 99
- preset lanes, 94
- query status, 165
- record length, 135
- report contents to save, 152
- report creation path, 150, 156
- report creation type, 155
- report update mode, 108
- sample rate, 136
- select device, 80
- select suite, 81
- send response, 165
- set verbose mode, 77
- signal quality preset, 93
- signal source signal type, 129
- signal validation, 139
- Sigtest version, 104
- slot number, 141, 142, 145, 147
- SSC, 89
- test mode, 84
- trigger type, 144
- view report after generating, 158
- voltage swing, 91
- waveform save, 113
- Plot images
 - including in reports, 48
 - viewing, 49
- Preferences menu, 43
- Preferences parameters, 40
- Preferences tab, 24, 40
- Prerecorded waveform files
 - selecting run sessions for, 25
- prerecorded waveform files (acquisitions), 35
- Prerun checklist, 54
- Preset test dependencies (8 Gbps), 99
- Probes (acquisitions), 36
- Program example, 66
- Programmatic interface, 59

Q

- QueryStatus(), 166

R

- Reactivate the license, 9
- Real time oscilloscope, 38
- Recalling a test setup, 57
- RecallSession(), 162
- Record length, 38
- Refresh sources (acquisitions), 36
- Related documentation, 1
- Remote access firewall settings, 60
- Report name, 47
- Report options, 47
- Report sections, 49
- Reports
 - adding user comments to, 25
 - receiving in email notifications, 21
- Reports panel, 22, 46
- Resource file, 13
- Results panel, 43
- RF switch, 32
- Run(), 159

S

- Sample Rate, 38
- Sampling rate, 207
- SaveSession(), 162
- SaveSessionAs(), 162
- Saving test setups, 56
- Saving tests, 45
- SCPI commands
 - Command parameters list, 202
 - TEKEXP:*IDN?, 187
 - TEKEXP:*OPC?, 187
 - TEKEXP:ACQUIRE_MODE, 188
 - TEKEXP:ACQUIRE_MODE?, 188
 - TEKEXP:EXPORT, 189
 - TEKEXP:INFO?, 189
 - TEKEXP:INSTRUMENT, 191
 - TEKEXP:INSTRUMENT?, 190

- TEKEXP:LASTERROR?, 191
- TEKEXP:LIST?, 192
- TEKEXP:MODE, 193
- TEKEXP:MODE?, 193
- TEKEXP:POPUP, 194
- TEKEXP:POPUP?, 194
- TEKEXP:REPORT, 195
- TEKEXP:REPORT?, 195
- TEKEXP:RESULT?, 196
- TEKEXP:SELECT, 197
- TEKEXP:SELECT?, 198
- TEKEXP:SETUP, 198
- TEKEXP:STATE, 199
- TEKEXP:STATE?, 199
- TEKEXP:VALUE, 200
- TEKEXP:VALUE?, 201

Selecting test report contents, 47

Selecting tests, 31

SendResponse(), 167

Server, 62

Session files, 45

Session folders, 45

Set acquisition Acquire options, 33

Set remote access, 60

Set SetPreRecorded (PI cmnds), 86

Set waveform save options, 34

SetDutId(), 79

SetTimeout(), 164

Setting up equipment, 51

Setting up tests, 51, 53

Setup files, 208

Setup panel, 22, 24

Setup panel views, 25

SetVerboseMode(), 77

Software installation, 5

Software version, 9

Sources (acquisitions), 36

Status panel, 41

Stop(), 159

Support, 2

System requirements, 3

T

Technical support, 2

TekExpress application for PCIe, 5

TekExpress client, 59

TekExpress client requirements, 62

TekExpress server, 59

Test failure action, 40

Test groups, 31

Test parameters (Configuration tab), 38

Test reports, 49

Test results

- emailing, 21

Test selection controls, 31

Test setup files, 45

Test setup steps, 53

Test setups

- creating, 57

- load, 57

- open, 57

- recalling, 57

- saving, 56

Test Status commands, 165

Test status tab, 41

Test-related files, 45

Tests

- running, 53

- selecting, 31

- setting up, 51

TestStand client example, 67

TransferImages(), 171

TransferResult(), 171

U

UnlockSession(), 175

Untitled session folder, 7

User account setting (Windows 7), 5

User comments

- location in reports, 49

User Comments

- including in reports, 48

User Defined Mode, 38, 51

V

View probes (acquisitions), 36

W

Waveform files

locating and storing, 45
Windows 7 user account setting, 5

