Tektronix®

TekExpress® USB2 Automated Test Solution

Printable Application Help

077-1429-00

TekExpress® USB2 Automated Test Solution

Printable Application Help

www.tek.com
077-1429-00

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.tek.com* to find contacts in your area.

Table of Contents

Running tests

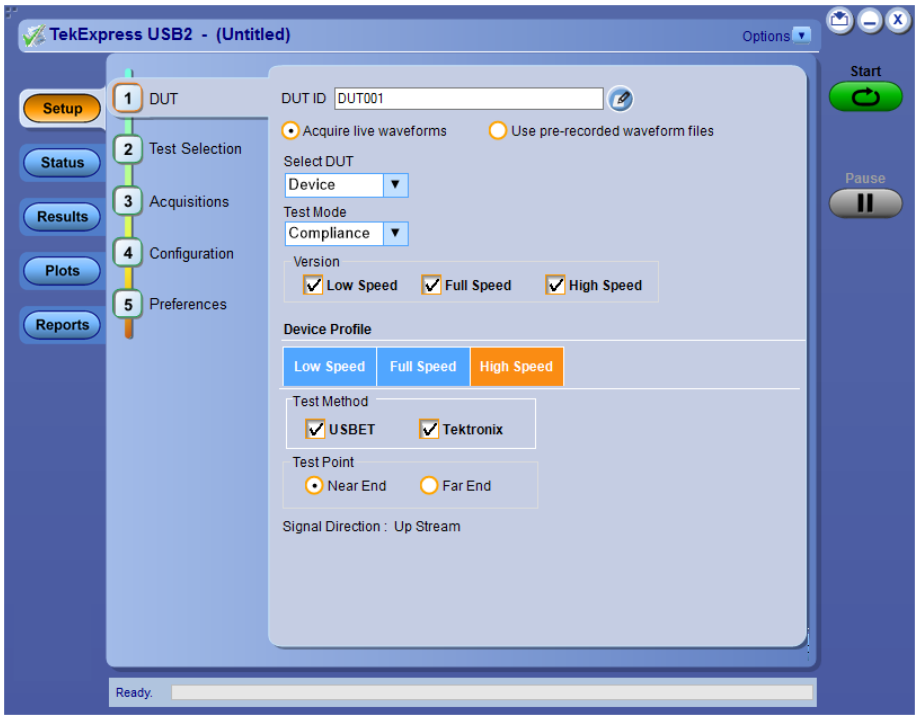
Saving and recalling test setup files

TekExpress USB2 programmatic interface

SCPI commands

Reference

Welcome



Welcome to the TekExpress® USB2 Automated Test Solution application. TekExpress USB2 is a Signal Quality, Non-Signal Quality, Power Measurement and Receiver Sensitivity Measurement solution, which provides an automated, simple, and efficient way to test USB 2.0 interfaces and devices consistent to the requirements of the USB 2.0 specifications.

Key features and benefits

- Comprehensive test coverage; select or deselect individual tests
- Precise debugging and troubleshooting
- USB-IF Signal Quality measurements integrated with TekExpress USB2
- Automated Receiver Sensitivity measurement
- User-friendly interface for plot inspection
- Minimizes user intervention when performing time-consuming testing
- Consolidated report for High Speed, Full Speed and Low Speed measurements
- Complete programmatic interface enables automation scripts to call TekExpress USB2 functions
- Flexible probe configuration

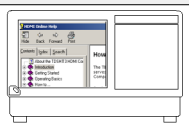
- Comprehensive signal validation check
- Automated DUT test mode control

Getting help and support

Related documentation

The following documents are available as part of the TekExpress® USB2 Automated Test and Compliance Solution application.

Table 1: Product documentation

Item	Purpose	Location
Application Help	Application operation and User Interface help	 Press **F1** on the oscilloscope keyboard to open Application Help.
PDF of the help	Printable version of the compiled help	 www.Tektronix.com PDF file that ships with TekExpress USB2 application (TekExpress USB2.pdf).

See also *Technical support*

Conventions used in help

Online 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 a screen item (button, control, list item): using a mouse or using the touch screen.

Table 2: Icon descriptions

Icon	Meaning
	This icon identifies important information.
	This icon identifies conditions or practices that could result in loss of data.
	This icon identifies additional information that will help you use the application more efficiently.

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 and send the setup files for all the instruments used and the application
- If possible, save and send the TekExpress USB2 setup files, *.TekX (session files and folders), and status messages text file
- If possible, save and send the waveform on which you are performing the measurement as a .wfm file
- If possible, log files of Tektronix HSETT Controller from folder C:\Program Files\Tektronix\TekApplication\Tektronix.\HSETT Controller\

Getting started

Minimum system requirements

The following table shows the minimum system requirements needed for an oscilloscope to run TekExpress USB2.

Table 3: TekExpress USB2 system requirements

Component	Requirement
Oscilloscope	See *Supported Instruments*
Processor	Same as the oscilloscope
Operating System	Same as the oscilloscope:
Memory	Same as the oscilloscope
Hard Disk	Same as the oscilloscope
Display	Same as the oscilloscope
Firmware	TekScope 1.6.3 and later (Windows 10, 64-bit only)
Software	■ TekExpress Framework version 4.5.0 ■ Iron Python 2.7.3 ■ PyVISA-1.3 ■ Microsoft .NET 4.0 framework ■ Microsoft Internet Explorer 6.0 SP1 or later ■ Adobe Reader 8.0 or equivalent software for viewing portable document format (PDF) files

Supported instruments

Table 4: Required equipments

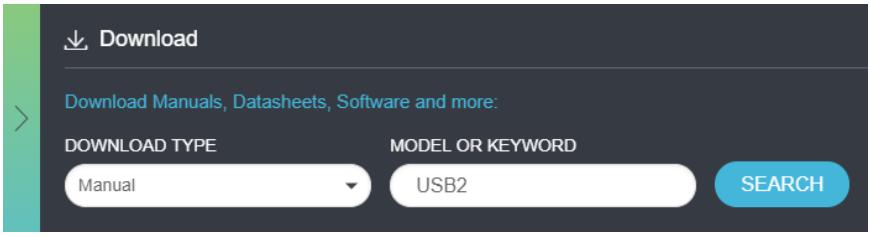
Resource	Model supported
Oscilloscope	Tektronix MSO54, MSO56, MSO58
AWG	AWG70001A, AWG70002A, AWG7102, AWG7122B, AWG7122C, AWG5014B, AWG5014C, AWG5012C, AWG5002C
Probes	■ Two TCA-SMA adapters ■ TCP0030A or TCP202/A probe ■ TDP1500, TDP3500, or P6248 differential probe ■ TAP1500 or P6245 single-ended probe
Host test fixtures	■ USB2SIGQUAL fixture set (USB-IF host/ device SQ) ■ USB2/3_DD fixture (USB-IF Drop-Droop fixture) ■ TDSUSBF (Tektronix fixture)
Device test fixtures	■ USB2SIGQUAL fixture set (USB-IF host/ device SQ) ■ USB2/3_DD fixture (USB-IF Drop-Droop fixture) ■ TDSUSBF (Tektronix fixture)

See also *Minimum system requirements*

Downloading and installing the software

Complete the following steps to download and install the latest TekExpress USB2 application. See Minimum system requirements for compatibility.

1. Go to *www.tek.com*
2. Click **Downloads**. In the Downloads menu, select DOWNLOAD TYPE as Software and enter USB2 in the MODEL OR KEYWORD field and click **SEARCH**.



3. Select the latest version of software and follow the instructions to download. Copy the executable file to the oscilloscope.
4. Double-click the executable and follow the on-screen instructions. The software is installed at C:\Program Files\Tektronix\TekExpress\USB2\.
5. Select **Application** > **TekExpress USB2** from the oscilloscope menu to launch the application.

See also

Minimum system requirements

Supported instruments

Activate the license

Follow the steps to activate the TekExpress USB2 license:

1. From the oscilloscope menu bar, click **Help** > **About**.
2. Click **Install License**; browse and select the license file (.Lic).
3. Follow the prompts of the oscilloscope to activate the license.

NOTE. *Contact Tektronix to purchase the TekExpress USB2 license.*

See also

View version and license information

View software version

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 TekExpress USB2, click **Options** > **About TekExpress**.

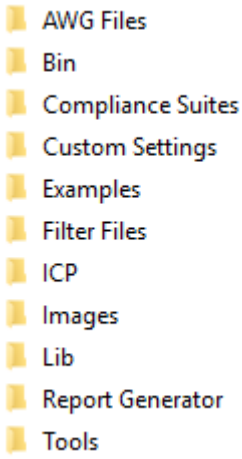
To view the license installed and option key information, from the oscilloscope menu, click **Help** > **About**.

See also

Activate the license

Application directories and their contents

The TekExpress USB2 application files are installed at the following location:
C:\Program Files (x86)\Tektronix\TekExpress\TekExpress USB2



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

Table 5: Application directories and usage

Directory names	Usage
AWG Files	Contains files required for Receiver Sensitivity measurement
Bin	Contains TekExpress USB2 application libraries
Compliance Suites	Contains compliance-specific files
Custom Settings	Contains custom settings files
Examples	Contains various support files
Filter Files	Contains filter files required for executing receiver sensitivity measurement
ICP	Contains instrument and TekExpress USB2 application-specific interface libraries
Images	Contains images
Lib	Contains utility files specific to the TekExpress USB2 application
Report Generator	Contains style sheets for report generation
Tools	Contains instrument and TekExpress USB2 application-specific files

See also

View test-related files

File name extensions

File name extensions

The TekExpress USB2 application uses the following file name extensions:

File name extension	Description
.py	Python sequence file
.xml	Test-specific configuration information (encrypted) files Application log files
.wfm	Test waveform files
.mht	Test result reports (default) Test reports can also be *saved in HTML format*
.xslt	Style sheet used to generate reports
.pdf	Test result reports Application help document
.csv	Test result reports Plot data

See also

View test-related files

Application directories and their contents

Where test files are stored

When you launch TekExpress USB2 for the first time, it creates the following folders on the oscilloscope:

- C:\Users\<username>\Documents\My TekExpress\USB2
- C:\Users\<username>\Documents\My TekExpress\USB2\Untitled Session

Every time you launch TekExpress USB2, an Untitled Session folder is created in the USB2 folder. The Untitled Session folder is automatically deleted when you exit the application. To preserve your test session files, save the test setup before exiting the 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, the application creates a .TekX file, and a folder named for the session that contains associated files, on the oscilloscope X: drive.*

See also

Application directories and usage

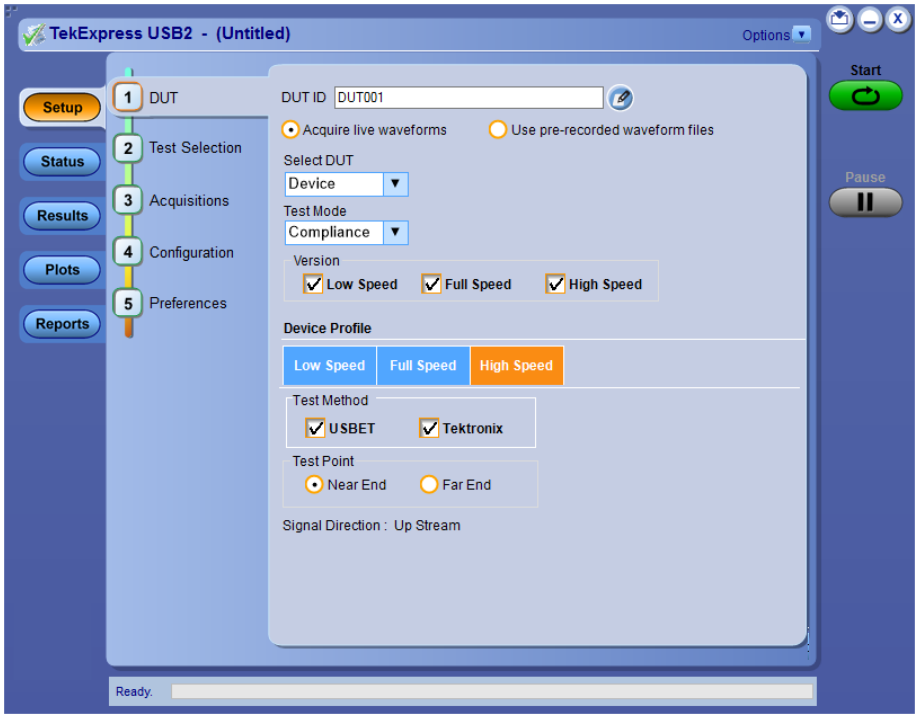
File name extensions

Operating basics

Launch the application

To launch the TekExpress USB2 application, select **Application** > **TekExpress USB2** from the oscilloscope menu.

The oscilloscope opens the **TekExpress USB2** application:



When you first run the application after installation, the application checks for Resources.xml located in the C:\Users\<username>\Documents\My TekExpress \USB2 folder. The Resources.xml file gets mapped to the X: drive when the application launches. Session files are then stored inside the X:\USB2 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 to detect connected instruments before launching TekExpress USB2.

To keep the TekExpress USB2 application window on top, select **Keep On Top** from the *Options menu*. If the application goes behind the oscilloscope application, click **Application > TekExpress USB2** to move the application to be in front.

See also *Application controls*

Application panel overview

Application panels overview

TekExpress USB2 uses panels to group related configuration, test, 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.

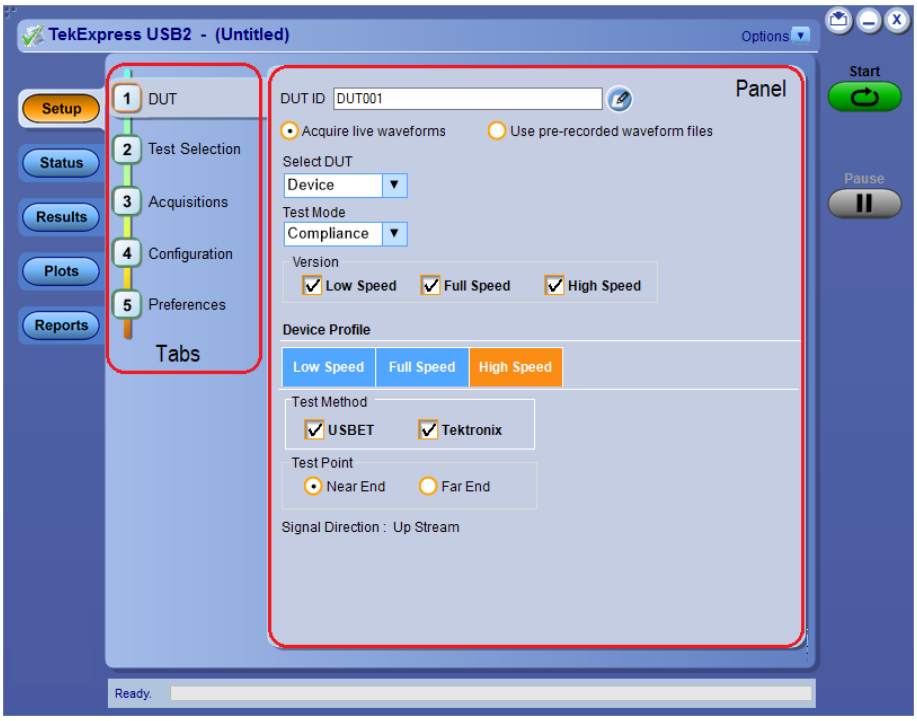


Table 6: Application panels overview

Panel Name	Purpose
Setup	The Setup panel shows the test setup controls. Click the **Setup** button to open this panel. Use this panel to: ■ *Select DUT parameters*. ■ *Select the test(s)*. ■ *Set acquisition parameters* for selected tests. ■ *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.
Plots	View a summary of plot generated during run.

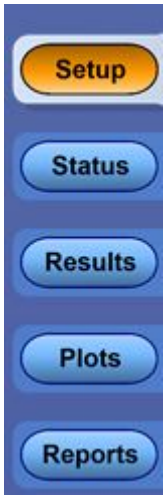
See also

Application controls

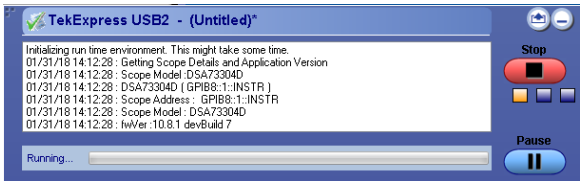
Global application controls

Application controls

Table 7: Application controls descriptions

Item	Description
Options menu 	Menu to display global application controls.
Test Panel buttons 	Controls that open panels for configuring test settings and options.
Start / Stop button  	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. The button toggles to the Stop mode while tests are running. Use the **Stop** button to abort the test.

Item	Description
Pause / Continue button  	Use the **Pause** button to temporarily interrupt the current acquisition. When a test is paused, the button name changes to “Continue.”
Clear button 	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 panel*.
Application window move icon 	Place the cursor over the three-dot pattern in the upper left corner of the application window. When the cursor changes to a hand, drag the window to the desired location.
Minimize button 	Minimizes the application.

Item	Description
Close button 	Closes the application.
Mini view / Normal view  	Toggles the application between mini view and normal view. Mini view displays the run messages with the time stamp, progress bar, Start / Stop button, and Pause / Continue button. The application automatically moves to the mini view when you click the Start button. 

See also.

Application panel overview

Options menu overview

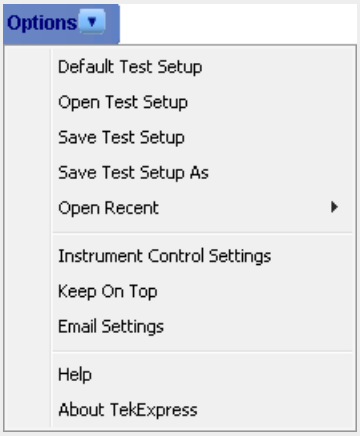
The **Options** menu is located in the upper right corner of the application.

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
Save Test Setup As	Creates a new test setup based on an existing one
Open Recent	Displays a menu of recently opened test setups to select from
Instrument Control Settings	Detects, lists, and refreshes the connected instruments found on specified connections (LAN, GPIB, USB, and so on)
Keep On Top	Keeps the TekExpress USB2 application on top of other open windows on the desktop
Email Settings	Use to configure email options for test run and results notifications

Menu	Function
Help	Displays the TekExpress USB2 help
About TekExpress	■ Displays application details such as software name, version number, and copyright ■ Provides access to *License information* for your TekExpress USB2 installation ■ Provides a link to the Tektronix Web site

Options menu



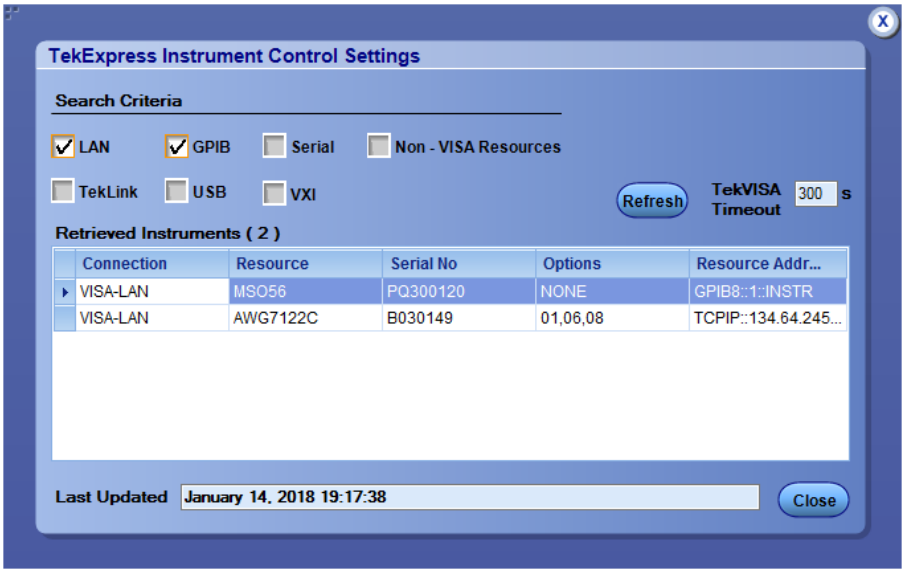
See also.

Application controls

Instrument control settings

Use the **TekExpress Instrument Control Settings** dialog box to search for and list the connected resources (instruments) detected on selected connections (LAN, GPIB, USB), and each instruments connection information.

To access, click **Options > Instrument Control Settings**.



Use the Instrument Control Settings feature to *search for connected instruments* and view instrument connection details. You can select listed connected instruments for use in the **Global Settings** tab in the test **Configuration** pane.

See also. *Options menu overview*

View connected instruments

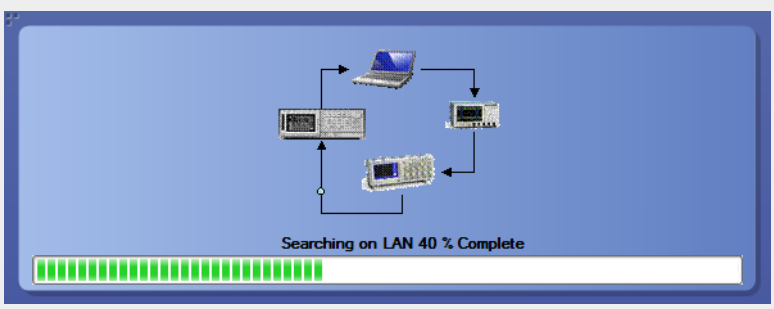
Use the Instrument Control Settings dialog box to view or search for connected instruments required for the tests. The application uses TekVISA to discover the connected instruments on all selected connection types.

NOTE. *The correct instruments for the current test setup must be connected and recognized by the application before running tests.*

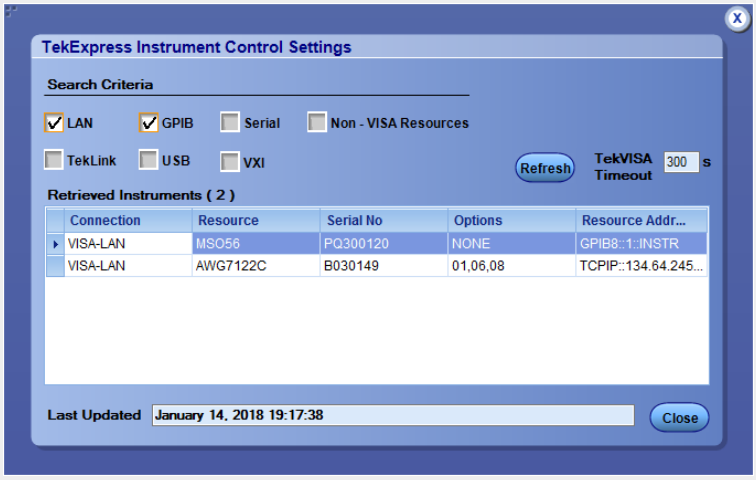
To refresh the list of connected instruments:

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.
3. Click **Refresh**. The application searches for connected instruments.



4. After searching, 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, and then lists detected instruments on those connection types.



The Retrieved Instruments table lists instrument details. The time and date of the last time this table was updated is displayed in the Last Updated field.

See also. *Equipment connection setup*

Configure email settings

Use the **Email Settings** dialog box to be notified by email when a test completes, fails, or produces an error:

1. Select **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 the test notification has to be sent. 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: MSO58_B130099@yourcompany.com.

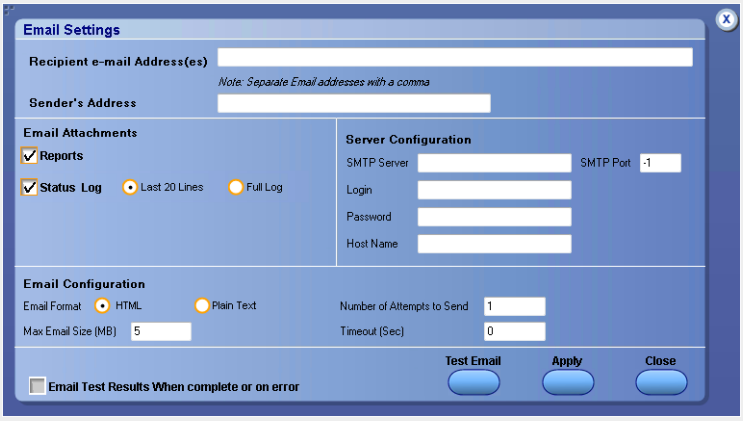
4. (Required) In the **Server Configuration** section, type the SMTP Server address of the Mail server configured at the client location, and the SMTP Port number, in the corresponding fields.

   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, select whether you want to receive the full log or just the last 20 lines.
6. In the **Email Configuration** section:
   - **Email Format**: Select the message file format to send: HTML (the default) or plain text.
   - **Max Email Size (MB)**: 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.
   - **Number of Attempts to Send**: Enter the number 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

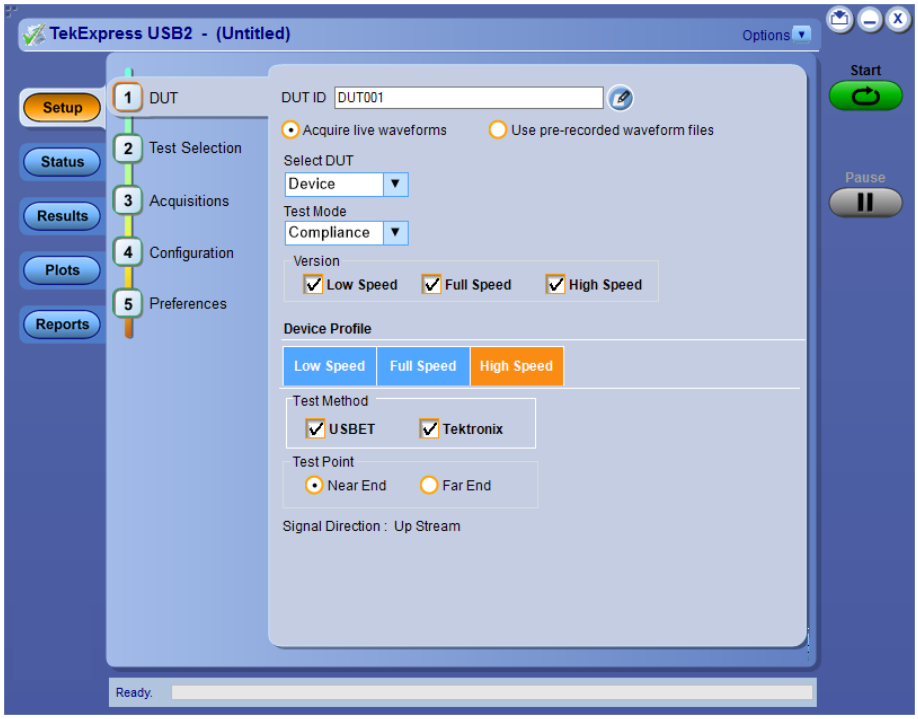


Application Panels

Setup panel

Setup controls overview

The Setup panel contains sequentially ordered tabs that guide you through a typical test setup and execution process. Click a tab to open the associated panel and controls.



The tabs on this panel are:

DUT: *Set the DUT parameters*

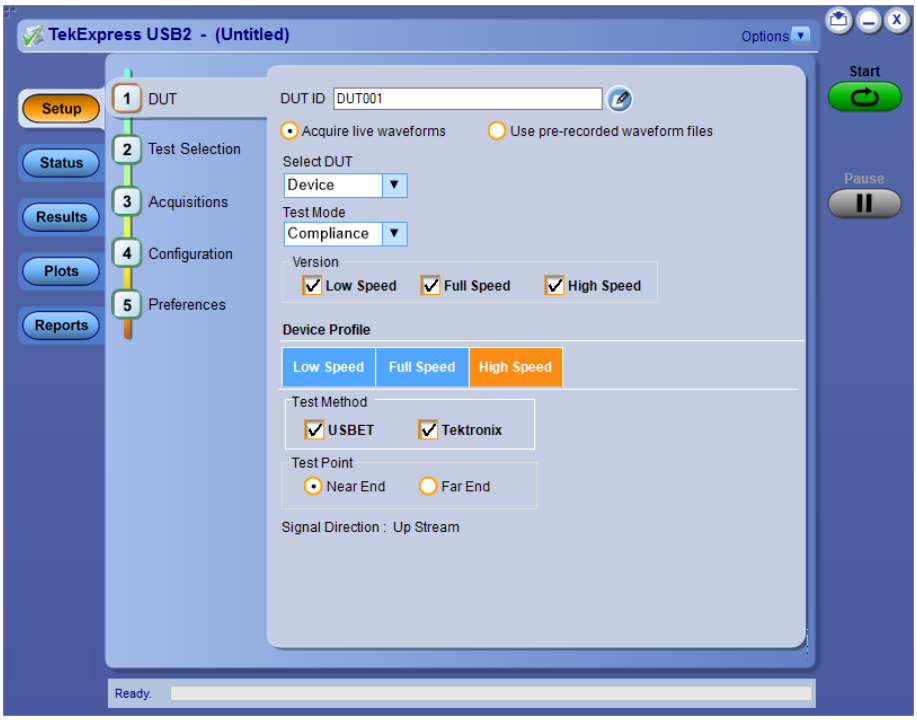
Test Selection: *Select test(s)*

Acquisitions: *Select acquisition parameters*

Preferences: *Select test fail 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 8: DUT tab settings

Setting	Description
DUT ID	Adds an optional text label for the DUT to reports. The default value is DUT001. The maximum number of characters is 32. You cannot use the following characters in an ID name: (.,..,...,\,/:?"<>|*)
 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. 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 measurement and analysis.
Use prerecorded waveform files	Run tests on a saved waveform. *Open (load) a saved test setup*

Setting	Description
Select DUT	Sets the DUT device type to test (Device, Host or Hub). Selecting the DUT device type preloads other fileds on the DUT panel.
Test Mode	■ Compliance: Preselects tests and parameters needed to meet compliance specifications for the selected device type. Disables the compliance filter controls. ■ User Defined: Enables the user to select specific tests and set custom parameters for tests (using the Configuration button).
Version	Lists the supported USB 2.0 generations.
Device Profile	
Low Speed, Full Speed and High Speed	Sets the available test parameters for the Device Profile area. Device Profile parameter availability depends on the Test Mode setting. The default test mode setting is High Speed.
Test Method	Sets the algorithms used to measure and analyze the signal. ■ USBET: Select to perform measurements implemented in USBET. ■ Tektronix: Select to perform measurements implemented by Tektronix.
Test point	Sets the Near End or Far End test point location. ■ Near End: Select if you connect the device with a captive cable. ■ Far End: Select if you connect the device with other than captive cable.
Signal Direction	Sets the Up or Down stream data signal direction. ■ Up Stream: Direction of data is towards the host. ■ Down Stream: Direction of data is away from the host.
Tier	Sets the Tier based on the position in the hub where the device is connected.

Setting	Description
Power Condition	Sets the energy source as Self Powered or Bus Powered. ■ Self Powered: Select if the power is drawn from an external energy source. ■ Bus Powered: Select if the power is drawn from the USB port.
Port	Enter the number of port used to connect to the host.

See also.

Select a test

Select tests

Use the Test Selection tab to select USB 2.0 tests. Listed tests depend on settings in the DUT panel.

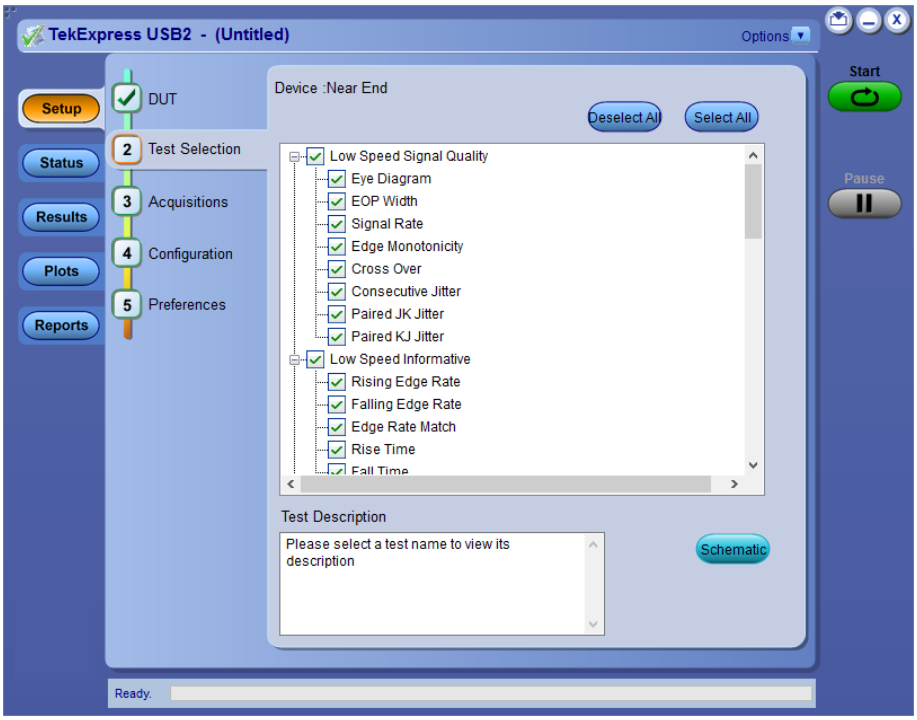


Table 9: Test Selection tab settings

Setting	Description
Deselect All, **Select All** buttons	Deselect or select all tests in the list.
Tests	Click a test to select or deselect. Selecting a test also shows details about that test in the Test Description pane. The application automatically selects all required tests when in Compliance mode.
Schematic button	Shows an equipment and test fixture setup schematic (connection diagram) for the selected test. Use to set up the equipment and fixtures or to verify the setup before running the test.

NOTE. *All tests are selected by default.*

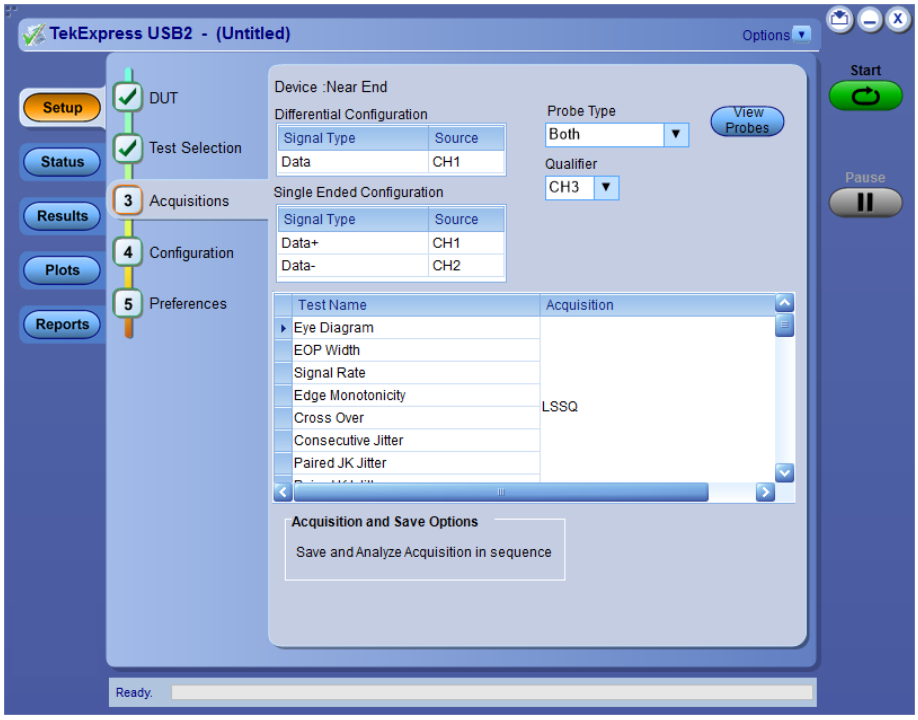
See also.

Set acquisition parameters

Set acquisition parameters

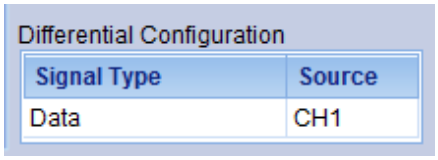
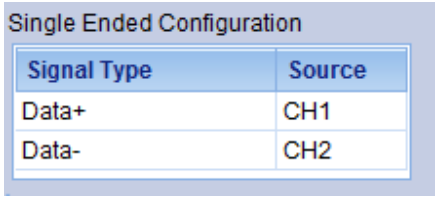
Select the Acquisition tab in the Setup panel to view test acquisition parameters. You can also use this tab to load pre-recorded (saved) test session waveform files on which you can run tests.

Contents displayed on this tab depend on the DUT type and selected tests.



NOTE. *TekExpress USB2 acquires all waveforms required by each test group and generation being tested before performing analysis.*

Table 10: Acquisitions tab settings

Setting	Description
Differential Configuration	Lists the signal type and input channel assigned to that type. 
Single Ended Configuration	Click in a Source fields to assign a channel source to a signal type. 
Qualifier	Acquires the signals from the reference device. Select the appropriate channel.
Probe Type	Sets the probe type based on the measurements.
View Probes button	Shows the detected probe configurations, and enable or disable probe signal source access in the application. Only available for live waveforms.

TekExpress USB2 saves all acquisition waveforms to files by default. 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:\TekExpress USB2\Untitled Session\<dutid>\<date>_<time>. Images created for each analysis, test session reports, and other information specific to that session are also saved in this folder.

When the session is saved, content is moved to that session folder and the "Untitled Session" name is replaced by the session name.

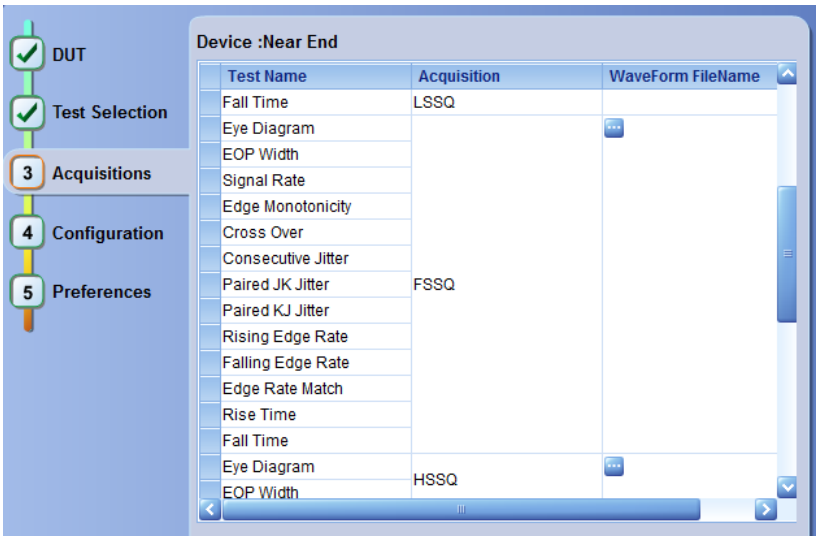
See also.

Running tests on prerecorded saved waveforms

Running tests on prerecorded (saved) waveforms

To load a saved waveform file on which to run tests:

1. Click **DUT**.
2. Click **Use pre-recorded waveform files**.
3. Click **Acquisitions**. The Waveform Filename column now shows browse buttons.



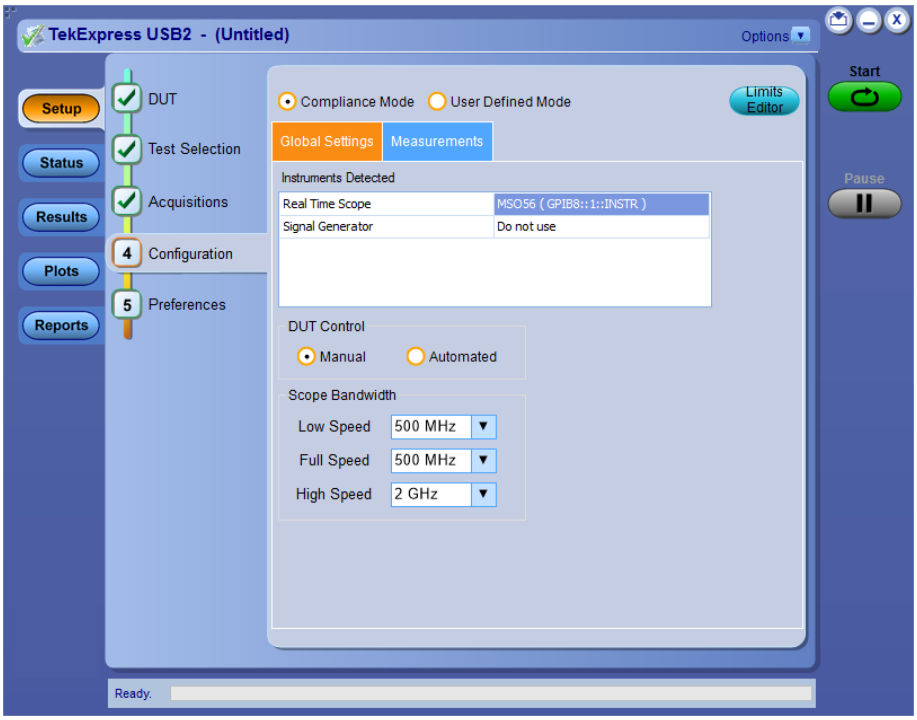
4. Click the browse button ( ) for each test acquisition type (For example, LSSQ, FSSQ, HSSQ).

   The **Select a Waveform file** dialog box appears.
5. Select the required waveform file(s). You must select all waveforms required for the acquisition type.
6. To change, remove, or add a file to the list, click the browse button next to the file name to change, and use the menu items to replace, remove (delete) or add a file in the list.
7. Click **Start**.

Configuration tab parameters

Use the Configuration tab to set and view global instrument parameters for the selected tests. Available fields for editing depend on the selected test mode (Compliance or User Defined) as set in this tab or the DUT tab.

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



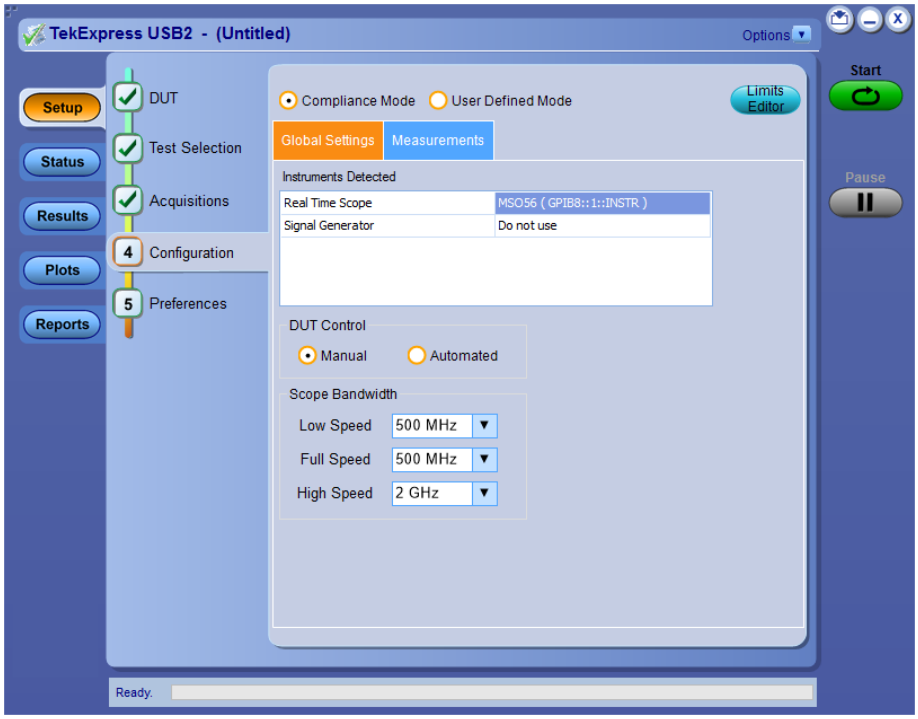
See also.

Configuration tab: Global settings parameters

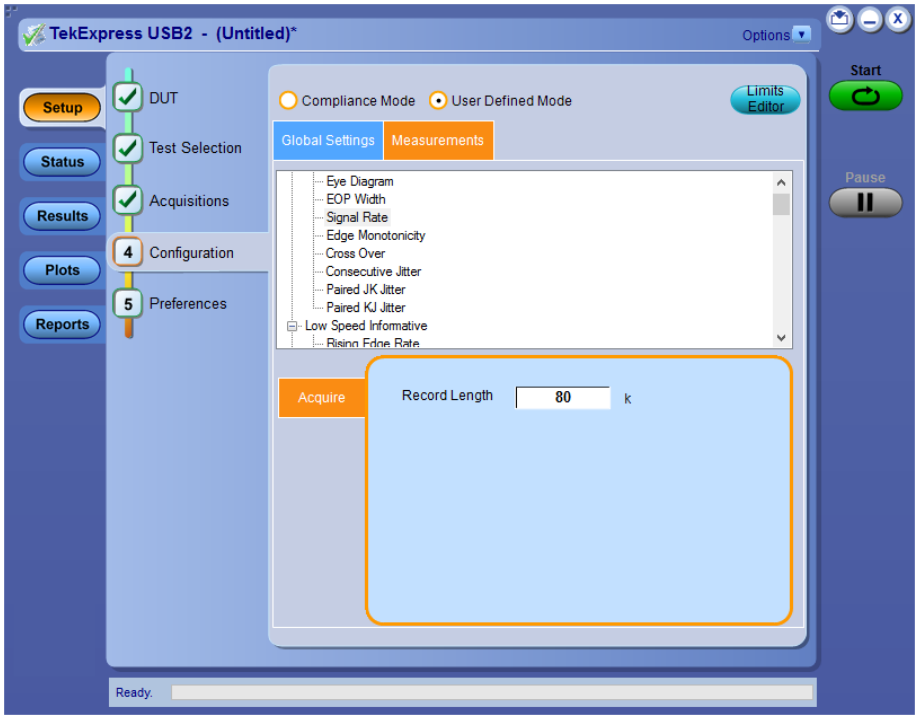
Configuration tab: global settings and measurement parameters

The following table lists the Configuration tab settings and measurement parameters. Fields shown on this tab can change depending on selected items.

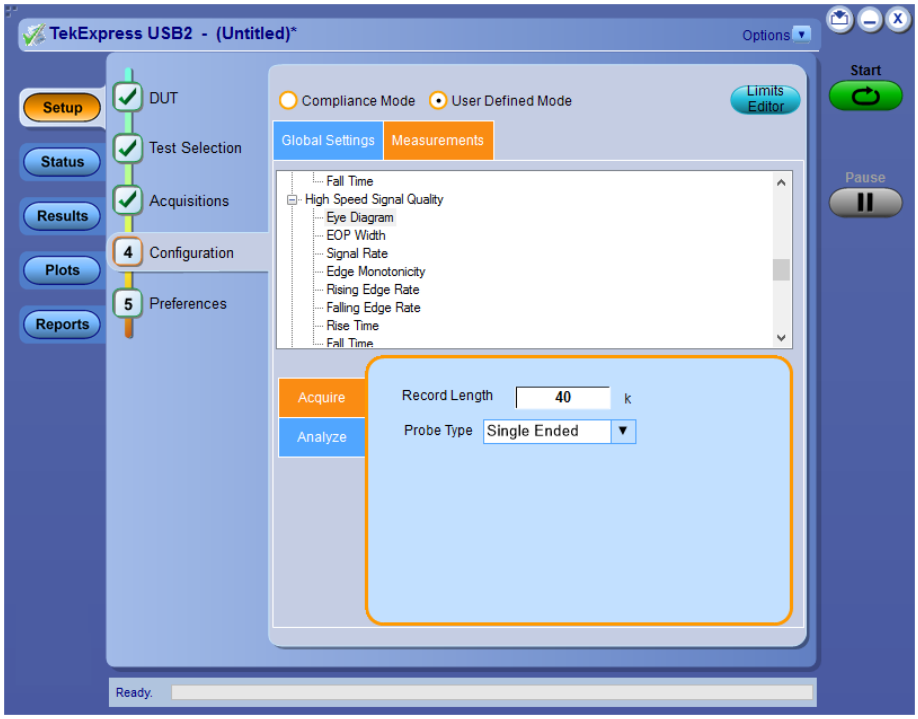
Configuration > Global Settings



Configuration > Measurements > Low Speed



Configuration > Measurements > High Speed



Configuration > Measurements > Power Measurements

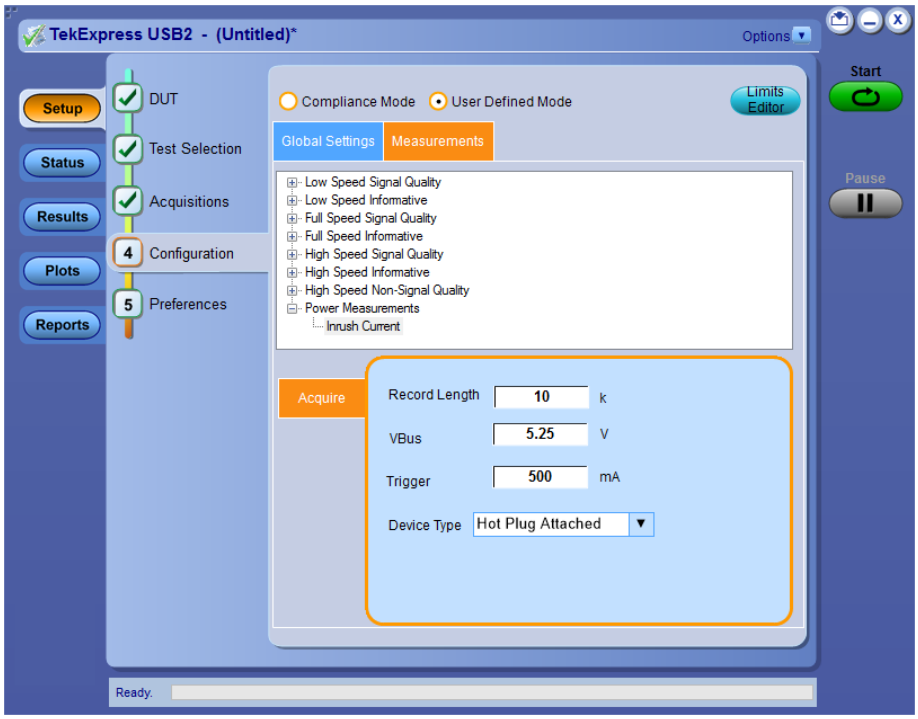
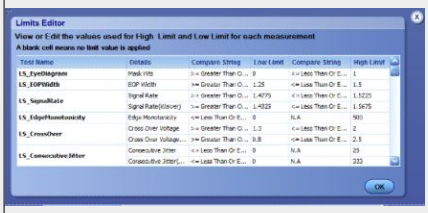


Table 11: Configuration tab Global and Measurement Settings

Control	Description
Test Mode	Determines whether test parameters are in compliance or are able to be edited. ■ Compliance: Most test parameter values cannot be edited. Select this for compliance mode testing. ■ User Defined: Enables editing of measurement configuration.
Limits Editor button	Opens the **Limits Editor** dialog box. In User Defined Mode, use the Limits Editor to edit individual test limit settings.  In Compliance Mode, use the Limits Editor to view the measurement high and low limits used for selected tests. You cannot edit values while in Compliance mode.
Global Settings Tab	Displays a list of the connected instruments.
Instruments Detected	Displays a list of the connected instruments found during the instrument discovery. Instrument types include equipment such as oscilloscopes and signal sources (AWG). Select **Options > Instrument Control Settings** to *refresh the connected instrument list*.
DUT Control	Display options for controlling DUT test mode.
Manual:	Select this option to control DUT test mode manually.
Automated:	Select this option to let TekExpress USB2 control the DUT test mode. Controller PC IP: Enter the IP address of Controller PC or Laptop.
Scope Bandwidth	Sets the oscilloscope bandwidth to use for Low Speed, Full Speed and High Speed tests. Select the bandwidth from the drop-down menu.

Control	Description
Measurements Setting Tab	Displays a list of selected measurements and tests.
Acquire/Analyze Tab	Sets additional parameters for specified measurements. For example, if you select Eye Diagram measurement under High Speed Signal Quality, the Acquire and Analyze tabs are populated with additional parameters to set. Probe Type: Select the appropriate probe. Custom Eye Mask: Browse to select a mask file. Record Length: Enter the record length value in the text box or use the TekExpress keypad. VBus: Select the appropriate source for bus voltage from the drop-down. Trigger: Select the appropriate source for trigger from the drop-down. Device Type: Select the appropriate device type from the drop-down.

See also.

About acquisitions

Preferences tab

Use the Preferences tab to set the application action when a test measurement fails.

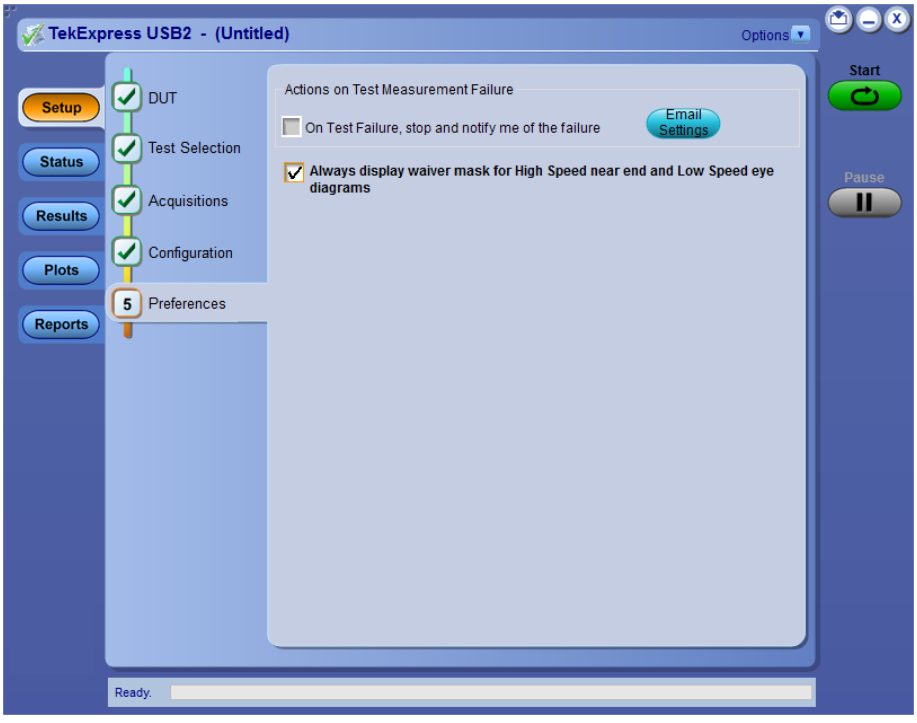


Table 12: Preferences tab settings

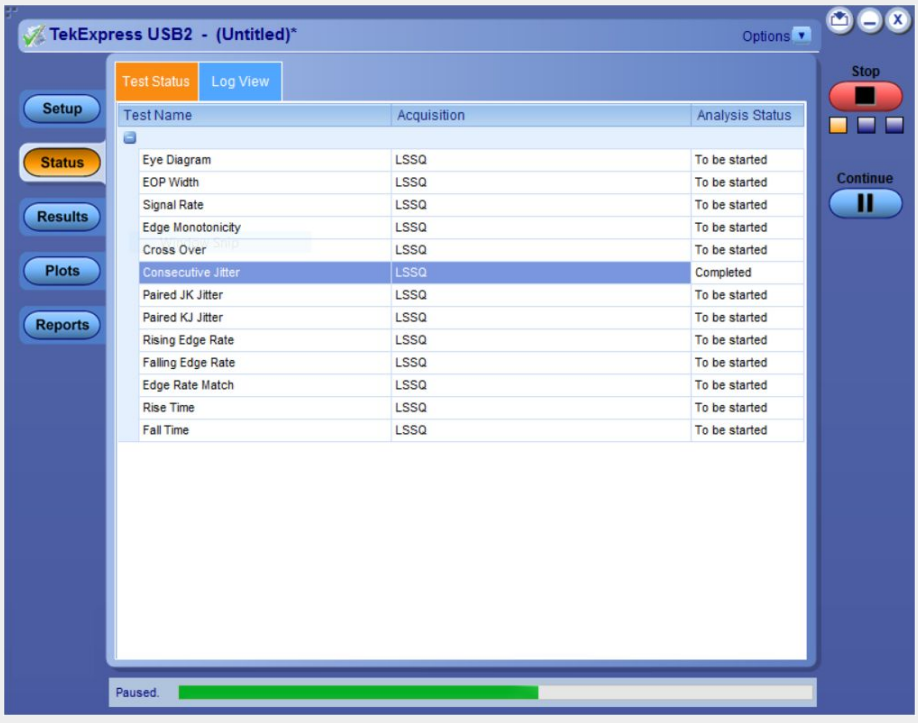
Setting	Description
On Test Failure, stop and notify me of the failure	Stops the test and sends an email when a test fails. Click **Email Settings** button and verify that "Email Test Results when complete or on error" is selected, and to verify the address to which the email is sent.
Always display waiver mask for High Speed near end and Low Speed eye diagrams	Sets the application to add the waiver mask for High Speed near end and Low Speed eye diagrams to the reports.

Status panel

Status panel overview

The **Status** button accesses the Test Status and Log View tabs, which provide 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.

Test status view



Log view

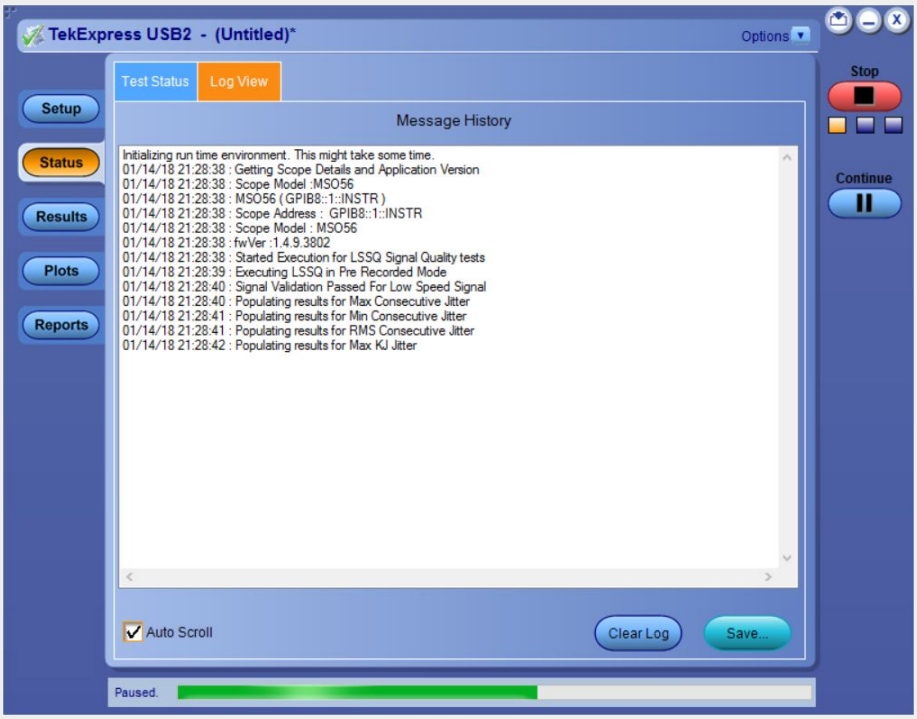


Table 13: Status panel : Log View pane settings

Control	Description
Message History	Window that lists all executed test operations and timestamp information.
Auto Scroll	Enables automatic scrolling of the log view as information is added to the log during the test.
Clear Log	Clears all messages from the log view.
Save	Saves the log file to a text file. Use the standard Save File window to navigate to and specify the folder and file name to which to save the log text.

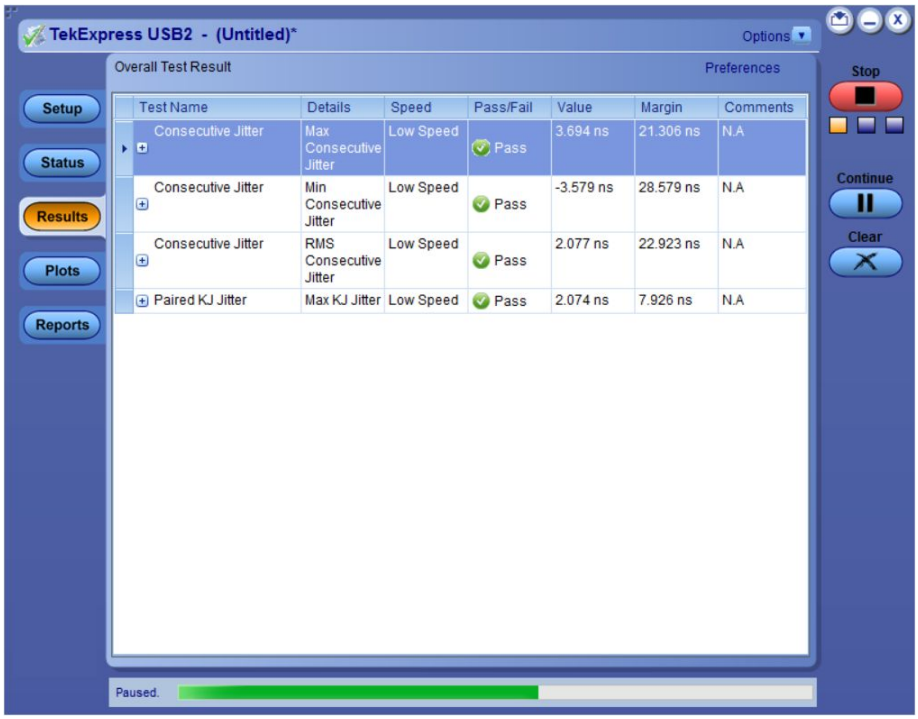
See also.

Application panel overview

Results panel

Results panel overview

When a test finishes, the application automatically opens the **Results** panel to display a summary of test results.



The Overall Test Result is displayed at the top left of the Results table. If all of the tests for the session pass, the overall test result is Pass. If one or more tests fail, the overall test result is Fail.

Set viewing preferences for this panel from the *Preferences menu* in the upper right corner. Viewing preferences include showing whether a test passed or failed, summary or detailed results, and enabling wordwrap.

Each test result occupies a row in the Results table. By default, results are displayed in summary format with the measurement details collapsed and with the Pass/Fail column visible. Change the view in the following ways:

- To expand and collapse tests to show more or less detail, click the plus and minus buttons in the table.
- 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 clear all test results displayed, click **Clear**.
- Use the *Preferences menu* to change how some items display in the Results panel.

See also.

View a report

Application panels overview

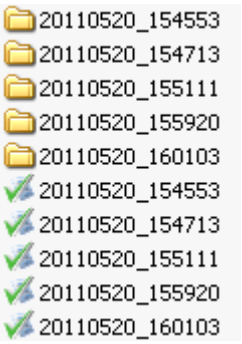
View test-related files

Files related to tests are stored in the C:\Users\<username>\Documents\My TekExpress\USB2 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 C:\Users\<username>\Documents\\My TekExpress\USB2. 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 application.

See also.

File name extensions

Preferences menu

The Preferences menu is part of the Results panel display. Use the Preferences menu to change how some items display in the Results panel.

- To stop the test and send an email when test fails, select **Preferences > On Test Failure, stop and notify me of the failure**.
- To display the weiver mask for High Speed near end and Low Speed eye diagrams in the reports, select **Preferences > Always display weiver mask for High Speed near end and Low Speed eye diagrams**.

See also.

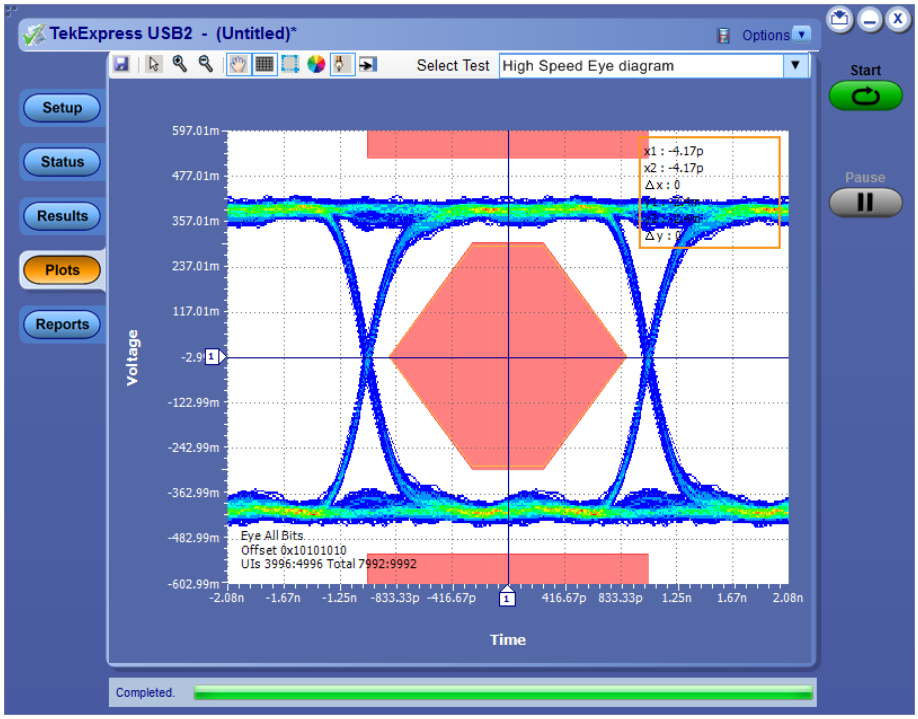
Results panel overview

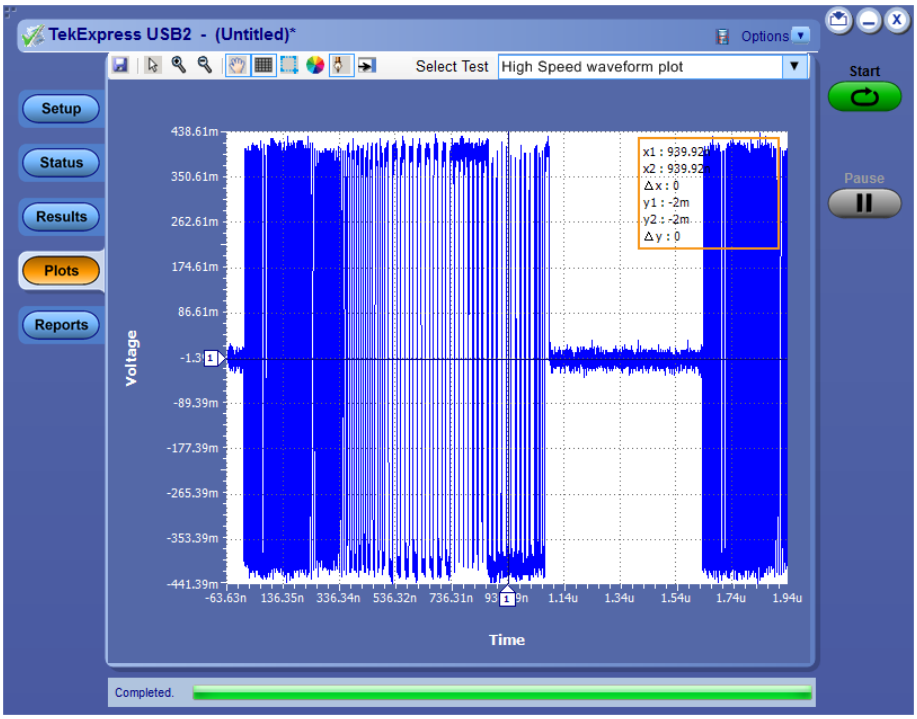
Plots panel

The **Plots** panel displays a summary of plot generated during run.

Depending on DUT speed and selected measurement(s), application will generate one/two plot(s) for each speed. These two plots are waveform plot and eye diagram plot.

These plots have zoom, cursors, save, and dock/undock features.





NOTE. *TekExpress USB2 keeps a copy of USBET generated plots in the saved test session folder.*

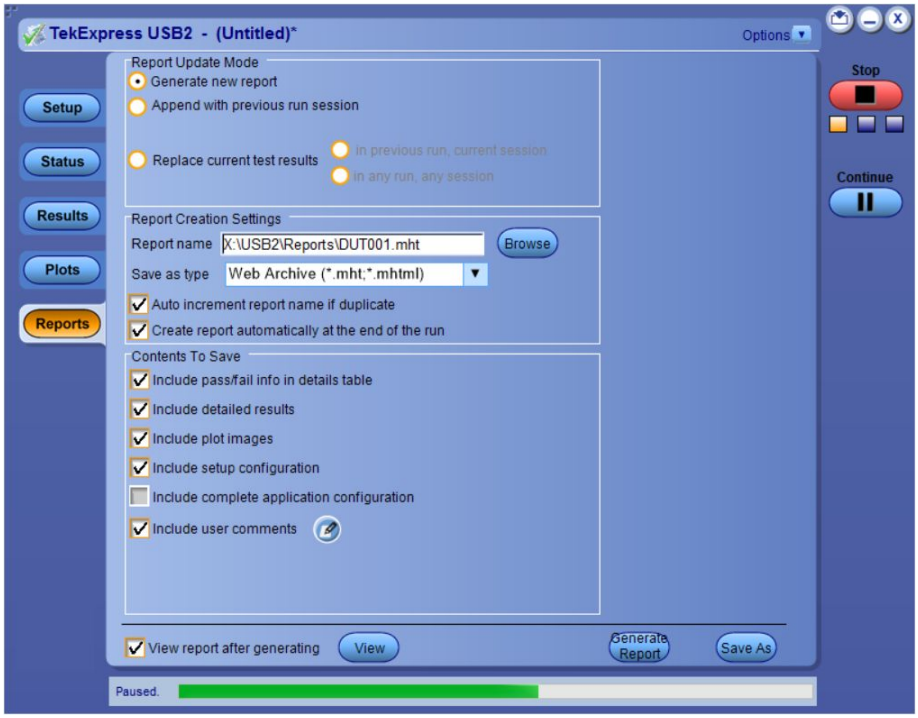
See also

Application panel overview

Reports panel

Reports panel overview

Use the Reports panel to view saved reports, name and save reports from the current session, 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 will be included in the report, and the naming conventions to use for the report.

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 14: Report options

Setting	Description
Report Generation	
Generate new report	Creates a new report.
Append to previous run session	Appends the latest test results to the end of the current session's 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 name	Displays the name and location from which to open a report. The default location is at *C:\Users\<username>\Documents\My TekExpress\USB2\Untitled Session*. 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. Change the report name or location. Do one of the following: ■ In the Report Path field, enter a new folder path and name. ■ Double-click in the Report Path field and then make selections from the popup keyboard and click the **Enter** button. Be sure to include the entire folder path, the file name, and the file extension. For example: C:\Users\<username>\Documents\My TekExpress\USB2 \DUT001_Test_72.7.1.3.mht. ***NOTE.*** *You cannot set the file location using the Browse button.* Open an existing report. Click **Browse**, locate and select the report file and then click **View** at the bottom of the panel.
Save as type	Saves a report in the selected output format (Web archive or PDF).

Setting	Description
Auto increment report name if duplicate	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.
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	Select to include detailed results of a test
Include plot images	Select to include plotted diagrams such as Eye diagram.
Include setup configuration	Select 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.
View Report After Generating	Select to automatically open 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	Generates a new report based on the current (most-recent) analysis results.
Save As	Specify a name for the report.

View a report

The application automatically generates a report when test analysis is completed and displays the report in your default Web browser (unless you cleared the **View Report After Generating** check box in the Reports panel before running the test). If you cleared this check box, or to view a different test report, do the following:

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

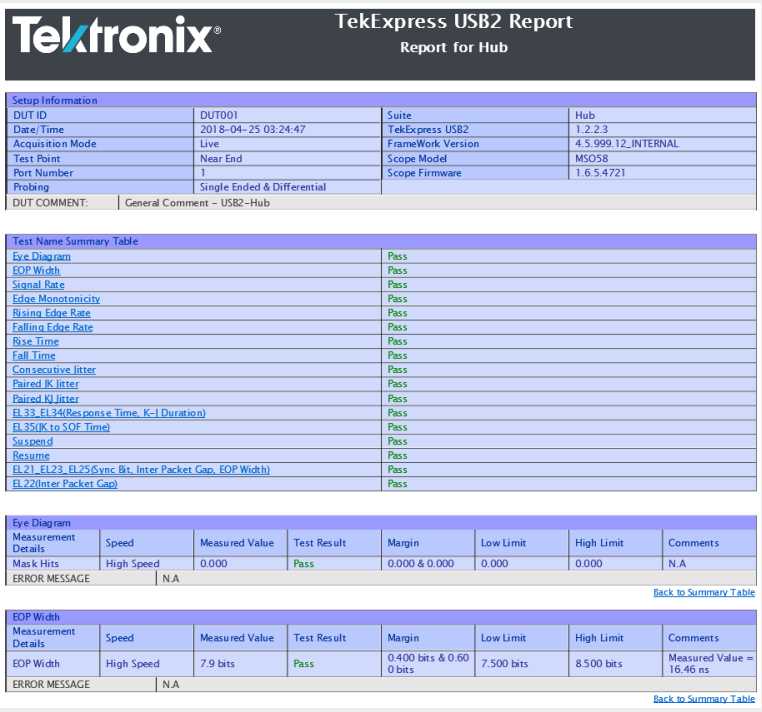
For information on changing the file type, file name, and other report options, see *Select report options*.

Report contents

A report shows specified test details, such as detailed results and plots, as set in the Reports panel.

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.



User comments

If you include comments in the test report, any comments you added in the **DUT** tab are shown at the top of the report.

Test result summary

The Test Result column indicates whether a test passed or failed. If the test passed, the cell text is green. If the test failed, the text is red. To exclude this information from a report, clear the **Include Pass/Fail Results Summary** check box in the Reports panel before running the test.

See also.

Results panel overview

View test-related files

Running tests

Test process flow

Allow test instrument to warm up for 20 minutes before running tests and use the following list to set up and perform USB2 tests.

1. *Set up test equipment*.
2. *Verify that required instruments are connected to USB2 fixtures*.
3. *Set DUT parameters*.
4. *Select tests*.
5. *View acquisition settings*.
6. *Set global signal-related parameters*.
7. *Select test notification preferences*.
8. *Select report options*.
9. *Check the prerun checklist*.
10. Click **Start** to *Run tests*.

See also

About test setups

About running tests

Instrument and DUT 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

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 USB2 application on top, select **Keep On Top** from the **Options** menu.

See also

Configuration tab parameters

Prerun checklist

Make sure that all the required instruments are properly warmed up for 20 minutes before running tests and do the following before you click the **Start** button to run a test:

1. Perform Signal Path Compensation (SPC)
   a. On the oscilloscope main menu, select the **Utilities** menu.
   b. Select **Instrument Calibration**.
   c. Follow the on-screen instructions.
2. Verify that the correct instruments are connected (oscilloscope and signal sources):
   a. In TekExpress USB2, click **Setup > Configuration**.
   b. Click **Global Settings**.
   c. In the Instrument Detected list, verify that the test setup instruments are shown. If they are not, click the arrow button to list and select from all detected instruments. If the required instrument is still not listed, use the **TekExpress Instrument Control Settings** dialog box to scan for and detect instruments (see *View connected instruments*).

See also

Instrument and DUT connection setup

Saving and recalling test setup files

Test setup files overview

Saved test setup 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:\USB2**.

Use test setups to:

- Run a new session, acquiring live waveforms, using a saved test configuration.
- Create a new test setup based on an existing one.
- 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.
- Run a saved test using saved waveforms.

See also

Save a test setup

Recall a saved test setup

Save a test setup file

Save a test setup before or after running a test to save the test settings. 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 immediately save the current setup session to the same setup name, select **Options > Save Test Setup**.

To immediately 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** to return the application to default test settings.
2. Click the application **Setup** button and use the setup tabs to set required options and parameters (DUT, Test Selection).
3. Click the application **Reports** button and set *report options*.

4. Optional: Click **Start** to run the test and verify that it runs correctly and captures the specified test information and reports. 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 new setup file. The application saves the file to X:\USB2\<*session_name*>.

See also

Test process flow

View test-related files

Configuration tab parameters

Open (load) a saved test setup file

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:\USB2**.

See also

About test setups

Create a new test setup based on an existing one

Test setups overview

Run a saved test in prerecorded mode

Run a saved test in prerecorded mode

Use this option to rerun a complete test using just the oscilloscope and the saved test setup files including the captured waveforms from the saved test.

NOTE. *When you run a saved test in prerecorded mode and then save it under the same name, the test results are saved in a new session folder named for the date and time of the session. Any test settings that you changed for the session are saved as a new test session file and are paired with a folder of the same name. For example, when you open a test setup that has multiple sessions and you select a session from the Run session list in the DUT tab, the settings associated with that test session are restored.*

Each test session folder has a matching test session file that stores the individual test settings for that session.

1. Use the Options menu to *Open a saved test setup file*
2. Select **Setup > DUT** and then select **Use pre-recorded waveform files**. A Run session drop-down list appears that displays the previous saved sessions for this test.
3. Select the session to run.

   NOTE. *If you select a session for which no waveform files were saved, you will receive an error message. Either select another test session or select waveform files to use.*

4. Click **Start**.
5. To save the test results, session settings, and related files, save the test setup before selecting another test setup or exiting the application.

See also

About test setups

Create a new test setup based on an existing one

Test setups overview

Create a new test setup file 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**.

   The **File Open** dialog box appears.

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**.

See also

About test setups

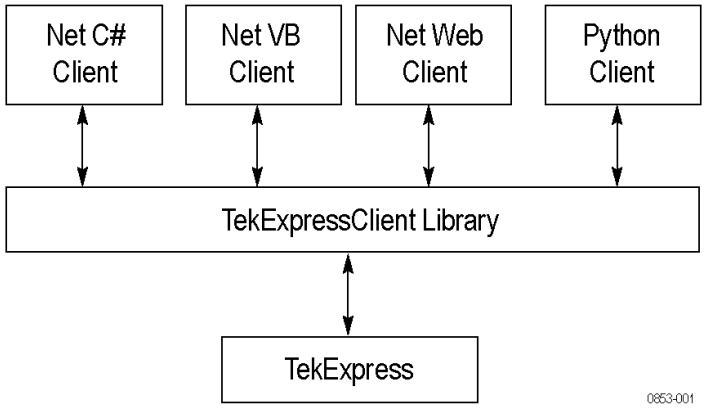
Set DUT parameters

Select acquisitions

TekExpress USB2 programmatic interface

About the programmatic interface

The Programmatic interface seamlessly integrates the TekExpress USB2 application with the high-level automation layer. This also lets you control the state of the TekExpress USB2 application running on a local or a remote computer.



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

- **TekExpress USB2 Client:** A high-level automation application that communicates with TekExpress USB2 using TekExpress USB2 Programmatic Interface.
- **TekExpress USB2 Server:** The TekExpress USB2 application when being controlled by TekExpress USB2 Client.

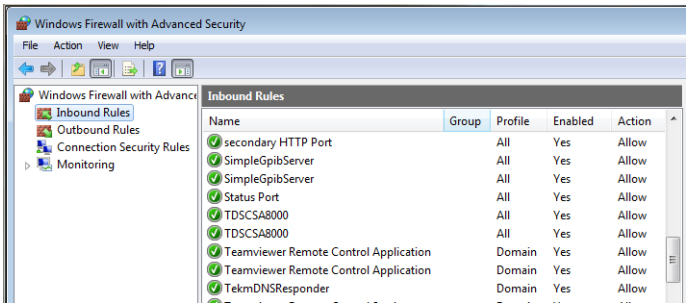
TekExpress USB2 leverages .Net Marshalling to enable the Programmatic Interface for TekExpress USB2 Client. TekExpress USB2 provides a client library for TekExpress USB2 clients to use the programmatic interface. The TekExpress USB2 client library is inherited from .Net MarshalByRef class to provide the proxy object for the clients. The TekExpress USB2 client library maintains a reference to the TekExpress USB2 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 USB2 programmatic interface, you need to change specific firewall settings as follows:

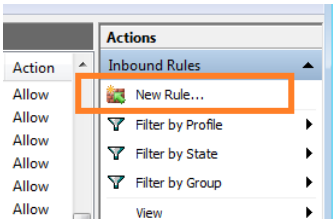
1. Access the Windows Control Panel and open the Windows Firewall tool (**Start > Control Panel > All Control Panel Items > Windows Firewall**).
2. Click **Advance Settings > Inbound Rules**.
3. Scroll through the **Inbound Rules** list to see if the following items (or with a similar name) are shown:
   - TekExpress USB2
   - 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 USB2 application.
6. On the client side, include controller.exe through which TekExpress USB2 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**.

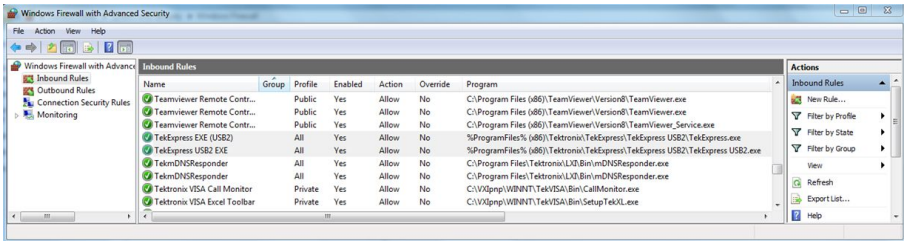


2. Verify that **Program** is selected in the Rule Type panel and click **Next**.
3. Click **Browse** in the Program panel and navigate to and select one of the following TekExpress USB2 applications (depending on the one for which you need to create a rule):
   - TekExpress USB2.exe
   - TekExpress.exe

NOTE. *See Application directories and content for the path to the application files.*

---

4. Click **Next**.
5. Verify that **Allow the connection** is selected in the Action panel and click **Next**.
6. Verify that all fields are selected (**Domain**, **Private**, and **Public**) in the Profile panel and click **Next**.
7. Use the fields in the Name panel to enter a name and optional description for the rule. For example, a name for the TekExpress USB2 application could be **TekExpress USB2 Application**. Add description text to further identify the rule.
8. Click **Finish** to return to the main Windows Firewall screen.
9. Scroll through the Inbound Rules list and verify that the list shows the rule that you just entered.



10. Repeat steps 1 through 9 to enter the other TekExpress USB2 executable if it is missing from the list. Enter **TekExpress USB2 PI** as the name.

11. Scroll through the Inbound Rules list and verify that the list shows the rule that you just entered.
12. Exit the Windows Firewall tool.

To use the remote access:

1. Obtain the IP address of the instrument on which you are running TekExpress USB2.
2. On the PC from which you are accessing the remote instrument, use the instrument IP address as part of the TekExpress USB2 PI code to access that instrument. For example:

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

Requirements for developing TekExpress USB2 client

While developing a TekExpress USB2 Client, use the TekExpressClient.dll. The client can be a VB .Net, C# .Net, Python, or Web application. Examples for interfaces in each of these applications are in the Samples folder.

Required steps for a client

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

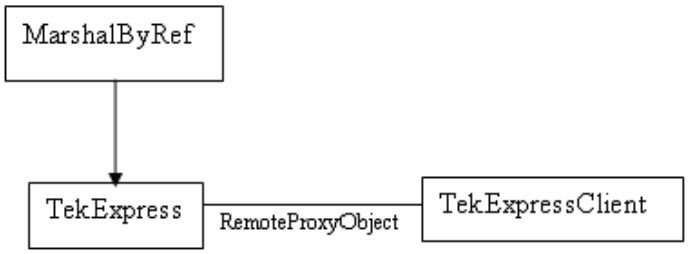
Develop a client UI to access the interfaces exposed through the server. This client loads TekExpressClient.dll to access the interfaces. After TekExpressClient.dll is loaded, the client UI can call the specific functions to run the operations requested by the client. When the client is up and running,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

Remote proxy object

The server exposes a remote object to let the remote client access and perform the server-side operations remotely. The proxy object is instantiated and exposed at the server-end through marshalling.



The following is an example:

RemotingConfiguration.RegisterWellKnownServiceType (typeof (TekExpressRemoteInterface), “TekExpress Remote interface”, WellKnownObjectMode.Singleton);

This object lets the remote client access the interfaces exposed at the server side. The client gets the reference to this object when the client gets connected to the server.

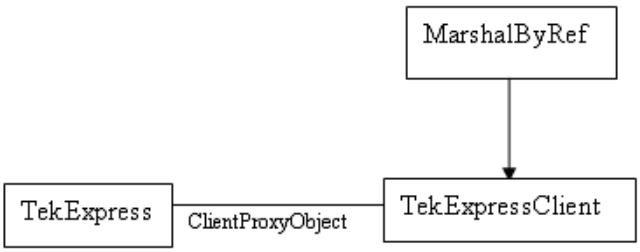
For example,

//Get a reference to the remote object

remoteObject = (IRemoteInterface)Activator.GetObject(typeof(IRemoteInterface), URL.ToString());

Client proxy object

The client exposes a proxy object to receive certain information.



For example,

//Register the client proxy object

WellKnownServiceTypeEntry[] e = RemotingConfiguration.GetRegisteredWellKnownServiceTypes();

clientInterface = new ClientInterface();

RemotingConfiguration.RegisterWellKnownServiceType(typeof(ClientInterface), “Remote Client Interface”, WellKnownObjectMode.Singleton);

//Expose the client proxy object through marshalling

RemotingServices.Marshal(clientInterface, "Remote Client Inteface");

The client proxy object is used for the following:

- To get the secondary dialog messages from the server.
- To get the file transfer commands from the server while transferring the report.

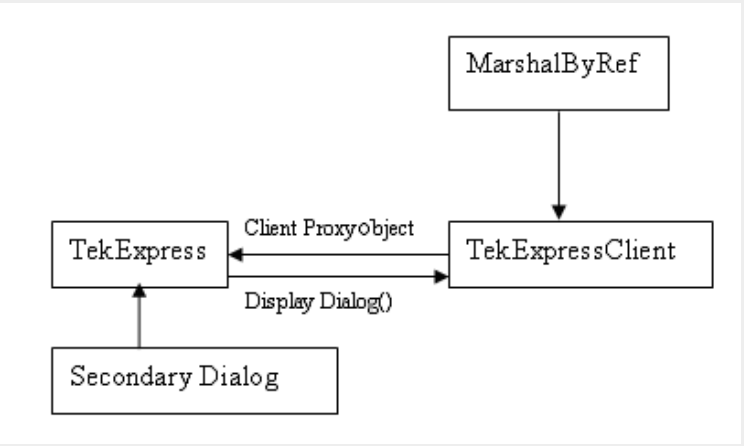
Examples

```
clientObject.clientIntf.DisplayDialog(caption, msg,iconType, btnType);

clientObject.clientIntf.TransferBytes(buffer, read, fileLength);
```

For more information, click the following links:

Secondary Dialog Message Handling



The secondary dialog messages from the Secondary Dialog library are redirected to the client-end when a client is performing the automations at the remote end.

In the secondary dialog library, the assembly that is calling for the dialog box to be displayed is checked and if a remote connection is detected, the messages are directed to the remote end.

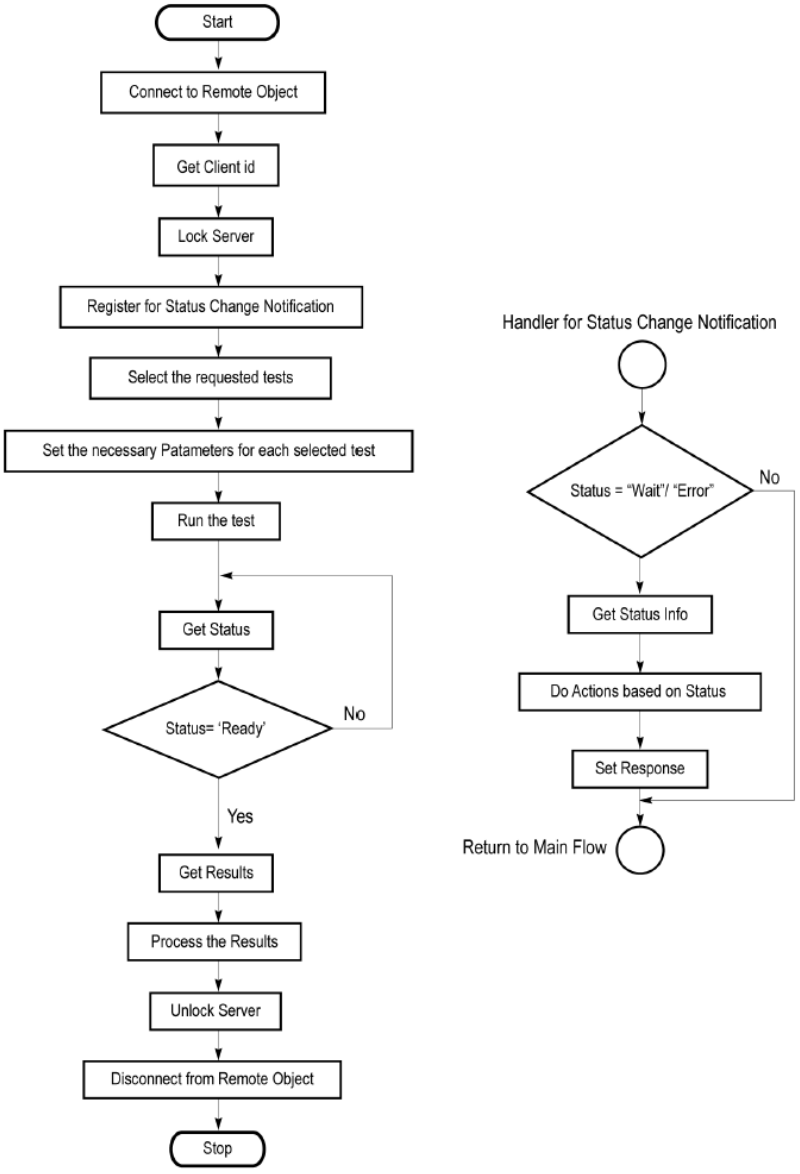
File Transfer Events

When the client requests the transfer of the report, the server reads the report and transfers the file by calling the file transfer methods at the client-end.

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

Program remote access code example

Program remote access code example

This code example shows how to communicate between a remote PC and TekExpress USB2.

Table 15: Remote access code example

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

Command list

Select Record Length ()

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.	m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL.
				returnval as string

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 USB 2.0 testing, set the device as Device, Host or Hub.

string devicesuite

Name	Type	Direction	Description
devicesuite	string	IN	Specifies the name of the USB 2.0 device suite enclosed in quotes. Valid values are **Device Connector**,**Host Connector** and **Hub Connector**.

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 is the Record Length. Valid value is **RecordLength** The second parameter sets the Record Length. See the following string examples:

Examples:

Example: with device set as Device and devicesuite set as Device Connector:

Given default value beside $ and the range next to it as given below:

- String example for ' Low Speed Tests ' : "RecordLength_LSSQ$80 " (Range: 50-500)
- String example for ' Full Speed Tests ' : "RecordLength_FSSQ$25" (Range: 20-500)
- String example for ' High Speed Tests ' : "RecordLength_HSSQ$40 " (Range: 15-1000)

String example for other ' High Speed Tests ' :

- "RecordLength_EL28_EL29_EL31$40" (Range:50-1000)
- "RecordLength_EL38$50" (Range: 5-90)
- "RecordLength_EL40$50" (Range:40-2000)
- "RecordLength_EL27$100" (Range:40-1000)
- "RecordLength_EL28$50" (Range:25-1000)
- "RecordLength_EL21_EL22_EL25$50"(Range:25-1000)
- "RecordLength_EL22$10"(Range:9-20)
- "RecordLength_Inrush$10" (Range:1-200)

Exampel: with device set as Host/Hub and devicesuite set as Host Connector/Hub Connector:

- String example for ' Low Speed Tests ' : "RecordLength_LSSQ$80 " (Range: 50-500)
- String example for ' Full Speed Tests ' : "RecordLength_FSSQ$25 " (Range: 20-500)
- String example for ' High Speed Tests ' :"RecordLength_HSSQ$40" (Range: 15-1000)

String example for other ' High Speed Tests ' :

"RecordLength_EL33_EL34$5" (Range:2-100)

"RecordLength_EL35$5" (Range:2-100)

"RecordLength_EL39$50" (Range:5-90)

"RecordLength_EL41$50" (Range:40-2000)

"RecordLength_EL21_EL23_EL25$50" (Range:25-1000)

"RecordLength_EL22$10" (Range:9-20)

"RecordLength_EL25$50" (Range:25-1000)

"RecordLength_Droop$5" (Range:2-1000)

Application status()

ApplicationStatus(clientId). This method gets the status (ready, running, paused) of the server application.

Parameters.

Name	Type	Direction	Description
clientId	string	IN	Identifier of the client that is performing the remote function. clientId example

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."

Return value. String value that gives the status of the server application.

Example. 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)

Comments. The application is in the Running, Paused, Wait, or Error state at any given time.

Related command(s).

Get Current State Info

Query status

Send response

Status

in string clientId example

clientId = <client_id_number>-<client_IP_address>.

For example, 1065–192.157.98.70

Check session saved()

CheckSavedSession(clientID, savedStatus). This command checks whether the current session is saved.

Parameters.

Parameter	Type	Direction	Description
clientId	string	IN	Identifier of the client that is performing the remote function.
savedStatus	boolean	OUT	Boolean representing whether the current session is saved

Return value. Return value is either True or False.

Example. 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)

Related command(s).

Recall session

Save session

Save session as

Connect()

Connect(hostIPAddress,clientInterface,clientID). This command connects the client to the server. The client provides the IP address of the server to connect to the server. The server provides a unique clientId when the client is connected to it.

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

Parameters.

Parameter	Type	Direction	Description
HostIPAddress	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.
clientIntf	string	IN	The handle of the remote object interface
clientId	string	OUT	Identifier of the client that is performing the remote function.

Return value. Value that indicates the connection status (connection was established or an error occurred). The return value can be a boolean value (true), or a string (returning the error message).

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."

Example. try {

IPAddress[] hostIPAddr = Dns.GetHostAddresses(Dns.GetHostName());

// Connect to the remoter Server

remoteObject.Connect(hostIPAddress, clientInterface, out clientID);

return true;

}

catch (Exception error)

{

return error;

}

Related command(s).

Disconnect

Disconnect()

Disconnect(clientId). This command disconnects the client from the server it is connected to.

Parameters.

Parameter	Type	Direction	Description
clientId	string	IN	Identifier of the client that is performing the remote function.

Return value. Integer value that indicates the status of the operation upon completion.

1: Success

–1: Failure

Example. try

{

string returnVal = UnlockServer(clientId);

remoteObject.Disconnect(clientId);

return 1;

}

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

Related command(s).

Connect

DUT automation()

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.	m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL.
				returnval as string

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 USB 2.0 testing, set the device as Device, Host or Hub.

string devicesuite

Name	Type	Direction	Description
devicesuite	string	IN	Specifies the name of the USB 2.0 device suite enclosed in quotes. Valid values are **Device Connector**,**Host Connector** and **Hub Connector**.

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 Test mode. Valid value is **DUT paramaters** The second parameter sets the test mode. Valid value is **55.577.88.89**. String example: "DUT Automation$True".

Related command(s).

Enter controller PC IP address

Enter controller PC IP address()

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.	m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL.
				returnval as string

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 USB 2.0 testing, set the device as Device, Host or Hub.

string devicesuite

Name	Type	Direction	Description
devicesuite	string	IN	Specifies the name of the USB 2.0 device suite enclosed in quotes. Valid values are **Device Connector**,**Host Connector** and **Hub Connector**.

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 Test mode. Valid value is **Controller PC IP** The second parameter sets the test mode. Valid value is **55.577.88.89**. String example: "Controller PC IP $23.55.78.99".

Related command(s).

DUT automation

Get current status info()

GetCurrentStatusInfo(clientID, WaitingMsbBxCaption, WaitingMsbBxMessage, WaitingMsbBxButtontexts). This command 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.

NOTE. *This command is used when the application is running and is in the wait or error state.*

Parameters.

Parameter	Type	Direction	Description
clientId	string	IN	Identifier of the client that is performing the remote function.
WaitingMsbBxCaption	string	OUT	The wait state or error state caption sent to you
WaitingMsbBxMessage	string	OUT	The wait state/error state message sent to you
WaitingMsbBxButtontexts	string array	OUT	An array of strings containing the possible response types that you can send

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."

Return value. This command does not return any value.

This function populates the Out parameters that are passed when invoking this function.

Example. m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL

mClient.GetCurrentStateInfo(clientID, WaitingMsbBxCaption, WaitingMsbBxMessage, WaitingMsbBxButtontexts)

Related command(s).

Application status

Query status

Send response

Get or set 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."

Get pass fail status()

GetPassFailStatus(clientId, device, deviceConnector, test). This command gets the pass or fail status of the measurement after test completion.

NOTE. *Execute this command after completing the measurement.*

Parameters.

Parameter	Type	Direction	Description
clientId	string	IN	Identifier of the client that is connected to the server
device	string	IN	Specifies the DUT type (**Host**.
deviceConnector	string	IN	string with device connection type. Valid values are **Host Connector**
test	string	IN	Specifies the name of the test for which to obtain the pass or fail status.

Return value. String value that indicates the status of the operation upon completion.

Example. GetPassFailStatus(clientId, “Host”, “Host Connector”, test);

Get report parameter()

GetReportParameter(clientId, device, suite, test, parameterString). This command gets the general report details such as oscilloscope model,TekExpress Firm Ware version and Application version.

Parameters.

Parameter	Type	Direction	Description
clientId	string	IN	Identifier of the client that is connected to the server
device	string	IN	Specifies the DUT type (**Host**.
deviceConnector	string	IN	string with device connection type. Valid values are **Host Connector**.
test	string	IN	Specifies the name of the test for which to obtain the pass or fail status or a test result value.
parameterString	string	IN	Specifies to return the measured value for the indicated test. Enter **"Scope Model"**, **"TekExpress Version"**, or **"Application Version"** for this argument

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."

Return value. The return value is the connected oscilloscope model, TekExpress base software version, or TekExpress UHS2 application version.

Example. GetReportParameter(clientId, "Host", "Host Connector", test, "Application Version")

Get results value()

GetResultsValue (clientId, device, deviceConnector, test, parameterString). This command gets the result values of the specified measurement after the run.

Parameters.

Parameter	Type	Direction	Description
clientId	string	IN	Identifier of the client that is connected to the server
device	string	IN	Specifies the DUT type (**Host**.
deviceConnector	string	IN	string with device connection type. Valid value is **Host Connector**
test	string	IN	Specifies the name of the test for which to obtain the test result value.
parameterString	string	IN	Specifies to return the measured value for the indicated test. Enter **"Value"** for this argument

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."

Return value. String value that indicates the status of the operation upon completion. Returns the result value in the form of a string.

Example. GetResultsValue(clientId, "Host", "Host Connector", test, "Value");

Lock server()

LockServer(clientID). This command locks the server to which it is connected.

Parameters.

Parameter	Type	Direction	Description
clientId	string	IN	Identifier of the client that is performing the remote function

Return value. Integer value that indicates the status of the operation upon completion.

Example. try

{

string returnVal = remoteObject.lockServer(clientId);

remoteObject.connect(clientId);

return 1;

}

Related command(s).

Unlock server

Lock session()

LockSession(clientId). This command locks the server. The client has to call this command before running any of the remote automations. The server is locked by only one client.

Parameters.

Parameter	Type	Direction	Description
clientId	string	IN	Identifier of the client that is performing the remote function

Return value. Returns the status of the operation upon completion.

Example. if (locked)

return “Session has already been locked!”;

returnVal = remoteObject.LockSession(clientId);

if ((OP_STATUS)returnVal == OP_STATUS.SUCCESS)

{

locked = true;

return “Session Locked...”;

}

Comments. When the client tries to lock a server that is locked by another client, the client gets a message that the server is already locked and it has to 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 automatically unlocked from that client.

Related command(s).

Unlock session

Query status()

QueryStatus (clientID, out status): This command transfers Analyze panel status messages from the server to the client.

Table 16: Parameters

Parameter	Type	Direction	Description
clientID	string	IN	Identifier of the client that is connected to the server
status	string array	OUT	The list of status messages generated during the run

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."

Return value

String value that indicates the status of the operation upon completion. On success the return value is “Transferred...”.

Example

returnVal=m_Client.QueryStatus(clientID, out statusMessages)

if ((OP_STATUS)returnVal == OP_STATUS.SUCCESS)

return “Status updated...”

else

return CommandFailed(returnVal)

Related command(s).

Application status

Get current state Info

Send response

Register status change notification()

RegisterStatusChangeNotification(clientID, statusChangeHandler). There are two ways to poll the application when it comes out of the Busy state. This command registers when there is an event, which indicates that activity is complete.

This command is used to select the particular version for a specific suite.

Parameters.

Parameter	Type	Direction	Description
clientId	string	IN	Identifier of the client that is performing the remote function
statusChangeHandler	Delegate of type TekExpressClient.StatusChangeHandler	IN	Handler

Return value. Returns an empty string when the operation is successful; otherwise it returns an error description.

Example. m_Client.RegisterStatusChangeNotification(clientId, new TekExpressClient.StatusChangeHandler (OnStatusChange));

```
public void OnStatusChange(string _status)
 {
 _status = m_Client.Application_Status(clientId);
 if (_status.CompareTo("Wait") == 0 || _status.CompareTo("Error") == 0)
   {
   string caption = "", message = "";
   string[] buttonTexts = null;
   m_Client.GetCurrentStateInfo(clientId, out caption, out message, out buttonTexts);
   Console.WriteLine("Caption:" + caption);
   Console.WriteLine("Message:" + message);
   Console.WriteLine("Message Type:" + FormatStringArray(buttonTexts));
   Console.WriteLine("Press Enter to send response . Waiting for Response...");
   string response = Console.ReadLine();
   m_Client.SendResponse(clientId, caption, message, response);
   Console.WriteLine("Message Response " + response + " Sent");
   }
 }
```

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."

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."

Save session as()

SaveSessionAs(clientId,sessionName). Saves the current session in a different name every time this command is called. The name of the session is provided by the client.

Parameters.

Parameter	Type	Direction	Description
clientId	string	IN	Identifier of the client that is connected to the server
sessionName	string	IN	The name of the session being saved.

Return value. String that indicates the status of the operation upon completion.

Example. returnVal = remoteObject.SaveSessionAs(clientId,sessionName);

if ((OP_STATUS)returnVal == OP_STATUS.SUCCESS)

return “Session Saved...”;

else

return CommandFailed(returnVal);

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

Related command(s). '

Recall session

Save session

Select device()

Selectdevice(clientId, device, true). This command selects the DUT type (Host).

Parameters.

Parameter	Type	Direction	Description
clientId	string	IN	Identifier of the client that is connected to the server
device	string	IN	String with the device DUT type. Valid value is **Host** .

Return value. String value that indicates the status of the operation upon completion.

Example. SelectDevice(clientId, "Host", true);

Select pre-recorded waveform files

Set pre-recorded waveform files (clientID, bset, ERRORString). This command selects the "Use pre-recorded waveform files" control in the DUT panel of the application UI.

Parameters.

Parameter	Type	Direction	Description
clientId	string	IN	Identifier of the client that is performing the remote function
bset	string	IN	This should be "True" or "False" based on the condition
ERRORString	string	IN	Error message to print if the command did not execute

Return value. 1 if pass, -1 if fail.

Example. m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL.

returnval as Integer = m_Client.SetPrerecorded(clientId, True, " ")

Where:

clientId = clientId

bset= True

Error= ""

Comments.

Use *Recall session()* before using this command.

Select port()

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.	m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL.
				returnval as string

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 USB 2.0 testing, set the device as**Hub**

string devicesuite

Name	Type	Direction	Description
devicesuite	string	IN	Specifies the name of the USB 2.0 device suite enclosed in quotes. Valid value is **Hub Connector**.

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 Ports. Valid value is **Ports** The second parameter sets the Test Method. Valid values are **Tier 1**, **Tire 2**,**Tire 3**,**Tire 4**,**Tire 5**,**Tire 6** String example: "Port $1".

Select probe type()

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.	m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL.
				returnval as string

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 USB 2.0 testing, set the device as Device, Host or Hub

string devicesuite

Name	Type	Direction	Description
devicesuite	string	IN	Specifies the name of the USB 2.0 device suite enclosed in quotes. Valid values are **Device Connector**,**Host Connector** and **Hub Connector**.

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 Probe Type: DUT Signal Probing. Valid value is **Probe Type: DUT Signal Probing** The second parameter sets the Probe Type: DUT Signal Probing. Valid values are **Differential**, and **Single-ended**. String example: "Probe Type:DUT Signal Probing$Differential".

Get results value for sub measurement()

GetResultsValueforsubmeasurements (clientId, device, deviceConnector, test, parameterString, rownumber). This command gets the result values of the sub measurement after the run.

Parameters.

Parameter	Type	Direction	Description
clientId	string	IN	Identifier of the client that is connected to the server
device	string	IN	Specifies the DUT type (**Host** or **Device**).
deviceConnector	string	IN	string with device connection type. Valid values are **Host Connector** and **Device Connector**
test	string	IN	Specifies the name of the test for which to obtain the test result value.
parameterString	string	IN	Specifies to return the measured value for the indicated test. Enter **"Value"** for this argument
rownumber	string	IN	Specifies the row number of the results panel.

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."

Return value. String value that indicates the status of the operation upon completion. Returns the result value in the form of a string.

Example. GetResultsValueForSubMeasurements(clientId, "Device", "Device Connector", "EL21_EL22_EL25", "Value", 0)

Related command(s).

Get results value

Select power condition()

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.	m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL.
				returnval as string

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 USB 2.0 testing, set this to Hub.

string devicesuite

Name	Type	Direction	Description
devicesuite	string	IN	Specifies the name of the USB 2.0 device suite enclosed in quotes. Valid value is ,**Hub Connector**.

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 Power Condition. Valid value is **Power Condition** The second parameter sets the Power Condition. Valid values are **Self Powered**, **Bus Powered**. String example: " Power Condition$Self Powered".

Select qualifier()

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.	m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL.
				returnval as string

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 USB 2.0 testing, set the device as Device, Host, and Hub.

string devicesuite

Name	Type	Direction	Description
devicesuite	string	IN	Specifies the name of the USB 2.0 device suite enclosed in quotes. Valid values are **Device Connector**,**Host Connector** and **Hub Connector**.

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 Qualifier. Valid value is **Qualifier** The second parameter sets the Qualifier. Valid values are **CH1**, and **CH2**,**CH3**,**CH4** String example: "Qualifier$CH1".

Select single test()

SelectSingleTest(clientID, device, suite, version, test). This command is to select a single test from a group of tests.

Parameters.

Parameter	Type	Direction	Description
clientId	string	IN	Identifier of the client that is performing the remote function
device	string	IN	Device DUT type
suite	string	IN	Host type
version	string	IN	Enter a null value for this field
test	string	IN	Name of the test

Return value. Returns an empty string if the command is executed properly, otherwise returns the string “Failed.”

Example. m_Client = new Client()

Note: m_Client is a reference to the Client class in the Client DLL.

To return a string:

returnval=m_Client.SelectSingleTest(clientId, device, suite, Version, test)

Where:

clientId = clientId

device = "Device" or "Host"

suite = "Device Connector" or "Host Connector"

Version= "" (null)

To select High Speed Eye Diagram Test:

test = "HS_EyeDiagram"

To select Full Speed Eye Diagram Test:

test = "FS_EyeDiagram"

To select Low Speed Eye Diagram Test:

test = "LS_EyeDiagram"

NOTE.

To configure 'probe type' for Packet Parameter test and High Speed SQ tests, use 'SetGeneralParameter' command in 'User Defined Mode'.

'

Example for Packet Parameter test.

Set the test mode to 'User Defined Mode' followed by setting probe type:

mClient.SetGeneralParameter(clientId, "Device", "Device Connector", "", "Test Mode$User Defined")

mClient.SetGeneralParameter(clientId, "Device", "Device Connector", "EL21_EL22_EL25", "Packet parameter selected probing$Single Ended")

'

Example for High Speed SQ tests.

Set the test mode to 'User Defined Mode' followed by setting probe type:

mClient.SetGeneralParameter(clientId, "Host", "Host Connector", "", "Test Mode$User Defined")

mClient.SetGeneralParameter(clientId, "Host", "Host Connector", "HS_SignalRate", "High speed probing$Differential")

Select suite()

Selectsuite (clientId, device, device Connector, true). This command selects one of the three suites: "Device Connector" or "Host Connector" or "Hub Connector".

Parameters.

Parameter	Type	Direction	Description
clientId	string	IN	Identifier of the client that is connected to the server
device	string	IN	String with the device DUT type. Valid values are **Host** and **DeviceHub**.
device Suite	string	IN	string with device connection type. Valid values are **Host Connector** and **Device ConnectorHub Connector**

Return value. String value that indicates the status of the operation upon completion.

Example. SelectSuite(clientId,"Device","Device Connector",true);

SelectSuite(clientId,"Device","Host Connector",true);

SelectSuite(clientId,"Host","Device Connector",true);

SelectSuite(clientId,"Host","Host Connector",true);

Select test method()

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.	m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL. returnval as string

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 USB 2.0 testing, set this as Device, Host or Hub .

string devicesuite

Name	Type	Direction	Description
devicesuite	string	IN	Specifies the name of the USB 2.0 device suite enclosed in quotes. Valid values are **Device Connector**,**Host Connector** and **Hub Connector**.

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 Test Method. Valid value is **Test Method** The second parameter sets the Test Method. Valid values are **USBET**, and **Tektronix**. String example: "Test mode$USBET". "Test mode$Tektronix"

Select test mode()

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.	m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL.
				returnval as string

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 USB 2.0 testing, set the device as Device,Host or Hub.

string devicesuite

Name	Type	Direction	Description
devicesuite	string	IN	Specifies the name of the USB 2.0 device suite enclosed in quotes. Valid values are **Device Connector**,**Host Connector** and **Hub Connector**.

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 Test mode. Valid value is **Test Mode** The second parameter sets the test mode. Valid values are **Compliance** and **User Defined**. String example: "Test mode$Compliance".

Select test point:Near End()

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.	m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL.
				returnval as string

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 USB 2.0 testing, set device as Device, Host or Hub.

string devicesuite

Name	Type	Direction	Description
devicesuite	string	IN	Specifies the name of the USB 2.0 device suite enclosed in quotes. Valid values are **Device Connector**,**Host Connector** and **Hub Connector**.

string parameterString (for test point)

Name	Type	Direction	Description
parameterString	string	IN	A string containing two parameters separated by a $ symbol, enclosed in quotes. To select Test Point, the first parameter is Near End . Valid value is **Near End** ,and To select Test Point, the second parameter is Near End Valid values are **True**,**False** String example: "Test point$Near End".

Select test point:Far End

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.	m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL.
				returnval as string

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 USB 2.0 testing, set device as Host or Hub.

string devicesuite

Name	Type	Direction	Description
devicesuite	string	IN	Specifies the name of the USB 2.0 device suite enclosed in quotes. Valid values are **Host Connector** and **Hub Connector**.

string parameterString (for test point)

Name	Type	Direction	Description
parameterString	string	IN	A string containing two parameters separated by a $ symbol, enclosed in quotes. To select Test Point, the first parameter is Far End . Valid value is **Far End** ,and To select Test Point, the second parameter is Far End Valid values are **True**,**False** String example: "Far End$True".

Select Tier()

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.	m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL.
				returnval as string

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 USB 2.0 testing, set the device as Device, Host and Hub

string devicesuite

Name	Type	Direction	Description
devicesuite	string	IN	Specifies the name of the USB 2.0 device suite enclosed in quotes. Valid values are **Device Connector**,**Host Connector** and **Hub Connector**.

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 Tier. Valid value is **Tier** The second parameter sets the Tier. Valid values are **Tier 1**,**Tier 2Tier 3Tier 4Tier 5Tier 6** String example: "Tier $Tier1".

Send response()

SendResponse(clientID, WaitingMsbBxCaption, WaitingMsbBxMessage, WaitingMsbBxButtontexts). After receiving the additional information using the command 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 GetCurrentState Info function.

NOTE. *This command is used when the application is running and is in the wait or error state.*

Parameters.

Parameter	Type	Direction	Description
clientId	string	IN	Identifier of the client that is connected to the server
WaitingMsbBxCaption	string	OUT	The wait state or error state message sent to you
WaitingMsbBxMessage	string	OUT	The wait state/error state message sent to you
WaitingMsbBxButtontexts	string array	OUT	An array of strings containing the possible response types that you can send

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."

Return value. This command does not return any value.

Example. m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL

mClient.SendResponse(clientID, out WaitingMsbBxCaption, out WaitingMsbBxMessage, out WaitingMsbBxButtontexts)

Related command(s).

Application status

Get current state Info

Query status

Select versions()

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.	m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL.
				returnval as string

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 USB 2.0 testing, set the device as Device, Host and Hub.

string devicesuite

Name	Type	Direction	Description
devicesuite	string	IN	Specifies the name of the USB 2.0 device suite enclosed in quotes. Valid values are **Device Connector**,**Host Connector** and **Hub Connector**.

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 Version. Valid value is **Version** The second parameter sets the Version. Valid values are **Low Speed**,**Full Speed** and**High Speed** String example: "Low Speed$True". "Full Speed $True"."High Speed $True"

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."

Set instrument()

SetInstrument(clientID, device, suite, test, paramString). Sets the specified instrument as a general configuration parameter to the selected test.

Parameters.

Parameter	Type	Direction	Description
clientId	string	IN	Identifier of the client that is performing the remote function
device	string	IN	Device DUT type
suite	string	IN	Host type
test	string	IN	Name of the test
paramString	string	IN	Specifies the control to set

Return value. Returns the string value of the instrument specified for setting in configuration parameter.

Example. mClient = new Client()

Dim clientId As String

Dim DUTType As String = “Device” or “Host”

Dim TekExpress_Suite As String = “Device Connector” or “Host Connector”

Dim str As String

Str= mClient.SetInstrument(clientId, DUTType, TekExpress_Suite, "UI-Unit Interval", " AnalyzeInstrument$Real Time Scope$ MSO58 (GPIB0::1::INSTR)")

Set verbose mode()

SetVerboseMode(clientId, verboseMode). This command sets the verbose mode to either true or false. When the value is set to true, event messages are routed to the client machine that is controlling TekExpress USB2.

When the value is set to false, event messages are shown only on the server machine.

Parameters.

Parameter	Type	Direction	Description
clientId	string	IN	Identifier of the client that is connected to the server
verboseMode	boolean	IN	Sets the verbose mode to be turned ON (true) or OFF (false).

Return value. String that gives the status of the operation after it was performed. Returnval as string.

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”.

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

If the server is LOCKED then the message shown is "Server is locked by another client".

If the session is UNLOCKED then the message shown is "Lock Session to execute the command".

If the server is NOTFOUND then the message shown is "Server not found...Disconnect!".

When none of these fail conditions occur, then the message shown is "Failed...".

Example. m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL.

Turn on verbose mode:

return=m_Client.SetVerboseMode(clientId, true);

Turn off verbose mode:

returnval=m_Client.SetVerboseMode(clientId, false);

Status()

Status(clientId, out statusMessages). This command gives the status of the run as messages. The status messages are generated once the run is started.

Parameters.

Parameter	Type	Direction	Description
clientId	string	IN	Identifier of the client that is connected to the server
statusMessage	string array	OUT	The list of status messages generated during run.

Return value. String that indicates the status of the operation upon completion.

Example. returnVal = remoteObject.QueryStatus(clientId, out statusMessages);

if ((OP_STATUS)returnVal == OP_STATUS.SUCCESS)

return “Status updated...”;

else

return CommandFailed(returnVal);

Comments. The status messages are updated periodically after the run begins. The status is an out parameter which is set when the server processes the request.

Related command(s).

Application status

Transfer images()

TransferImages(clientId, filePath). This command transfers all the images (screen shots) to the specified client and folder (directory) from the current run.

NOTE. *Every time you click Start, a folder is created in the X: drive. Transfer the waveforms before clicking Start.*

Parameters.

Parameter	Type	Direction	Description
clientId	string	IN	Identifier of the client that is connected to the server
filePath	string	IN	The location where the screen shots 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. *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."

Return value. String value that indicates the status of the operation upon completion. Transfers all the images in the form of a string.

Example. TransferImages(clientId, "C:\Images")

Transfer result()

TransferResult(clientID, Filepath). Transfers (saves) the result from the results panel information to the specified path.

Parameters.

Parameter	Type	Direction	Description
clientId	string	IN	Identifier of the client that is performing the remote function
Filepath	string	IN	Specifies the destination path of the file to be saved

Return value. Return a string as "Transferred…"

Example. m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL

TransferResult as string = m_Client.TransferResult(clientId, "C:\abc\Results\");

Transfer waveforms()

TransferWaveforms(clientID, path). This command transfers all the acquired waveforms to the specified location.

Parameters.

Parameter	Type	Direction	Description
clientId	string	IN	Identifier of the client that is performing the remote function
path	string	IN	Path to location at which to store waveforms

Return value. Returns a string as "Transferred…"

Example. m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL

TransferWaveforms as string = m_Client.TransferWaveforms(clientId, "C:\abc \Waveforms\");

Unlock server()

UnlockServer(clientID, path). This command unlocks the server to which it is connected.

Parameters.

Parameter	Type	Direction	Description
clientId	string	IN	Identifier of the client that is performing the remote function

Return value. Returns and integer value that indicates the status of the operation upon completion. Session UnLocked…

Example. try

{

string returnVal = remoteObject.UnlockServer (clientId);

remoteObject.disconnect (clientId);

return 1;

}

Comments. When the client is disconnected, it is unlocked from the server and then disconnected. The ID is reused.

Unlock session()

UnlockSession(clientId). This command unlocks the server from the client. The client id of the client to be unlocked has to be provided.

Parameters.

Parameter	Type	Direction	Description
clientId	string	IN	Identifier of the client that is connected to the server

Return value. String that indicates the status of the operation upon completion.

Example. returnVal = remoteObject.UnlockSession(clientId);

if ((OP_STATUS)returnVal == OP_STATUS.SUCCESS)

{

locked = false;

return “Session UnLocked...”;

}

Comments. When the client is disconnected, it is automatically unlocked.

Related commands.

Lock session

SCPI commands

About SCPI command

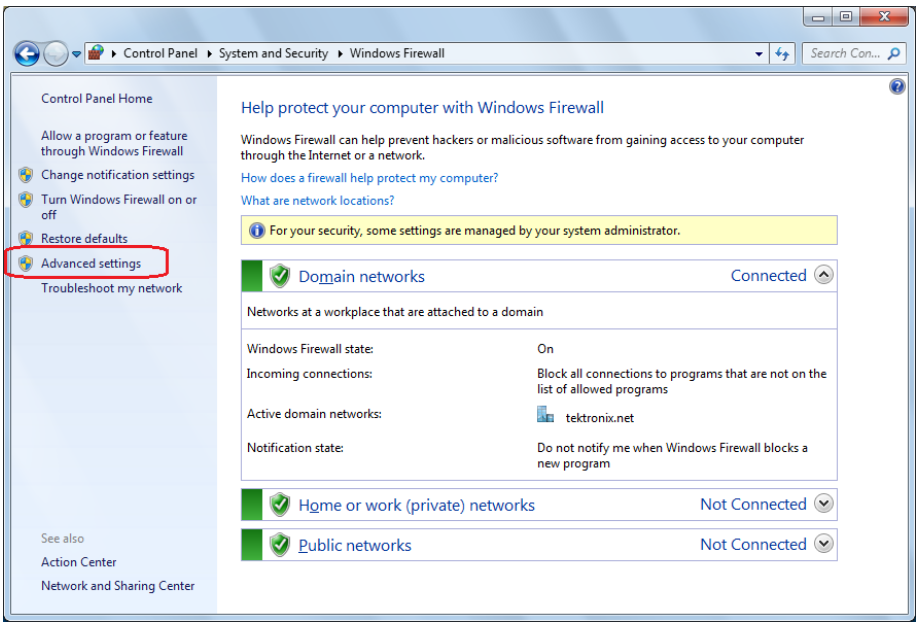
You can use Standard Commands for Programmable Instruments (SCPI) to communicate with the TekExpress application.

Socket configuration for SCPI commands

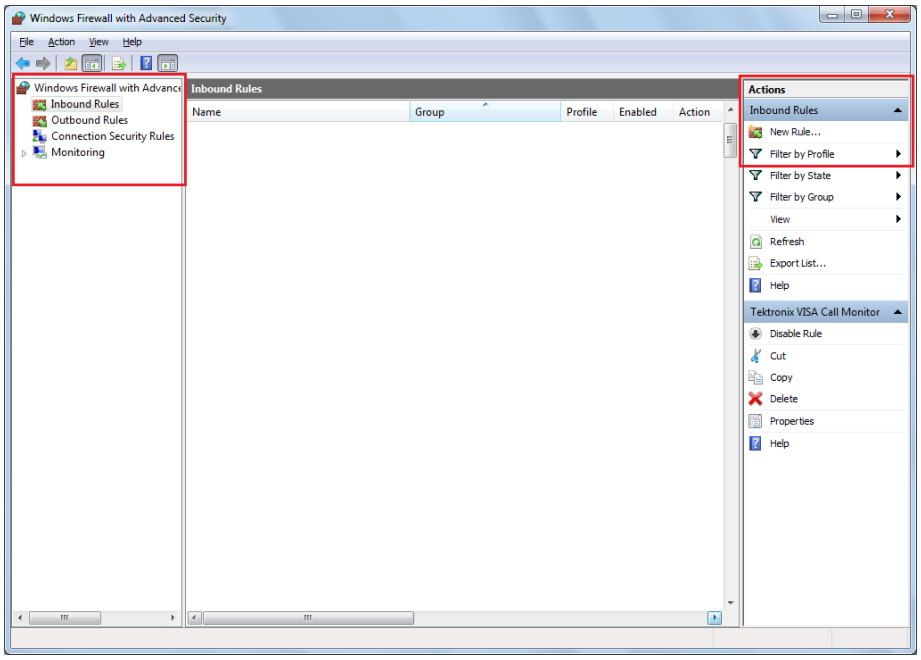
This section describes the steps for TCPIP socket configuration and TekVISA configuration to execute the SCPI commands.

TCPIP socket configuration

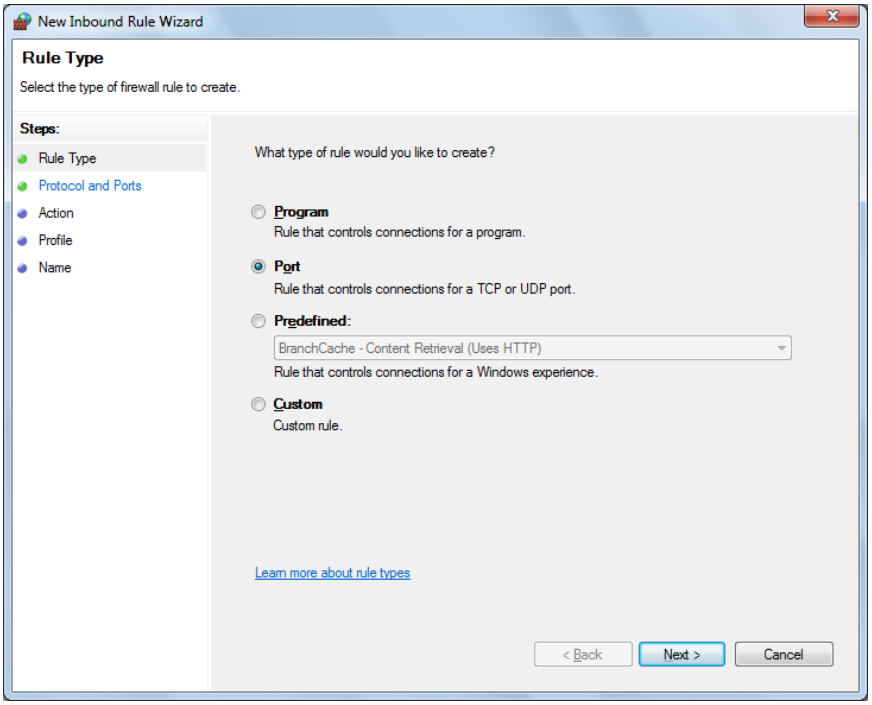
1. Click **Start** > **Control Panel** > **System and Security** > **Windows Firewall** > **Advanced settings**



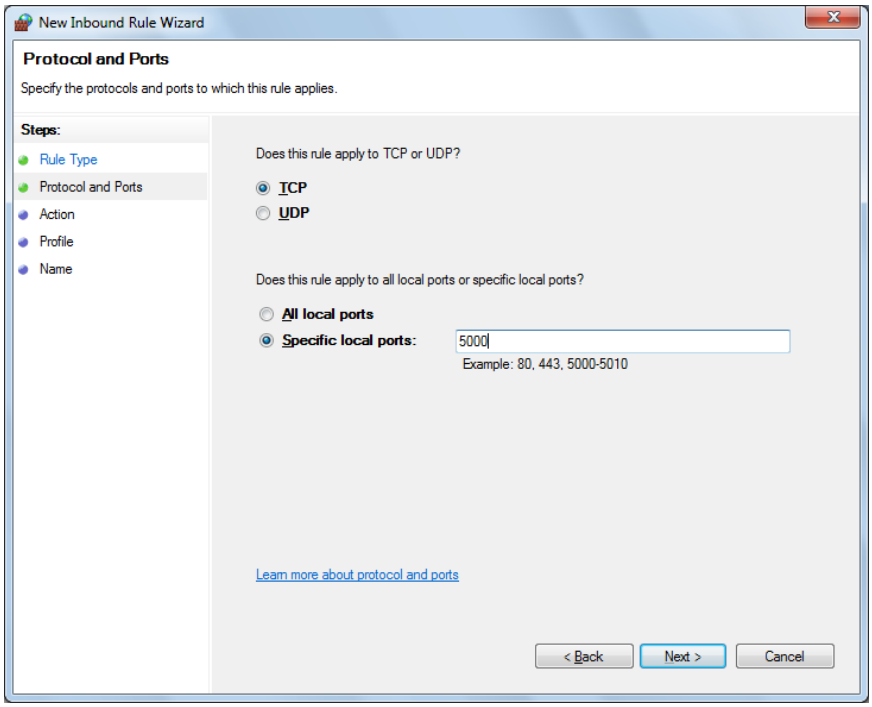
2. In Windows Firewall with Advanced Security menu, select **Windows Firewall with Advanced Security on Local Computer** > **Inbound Rules** and click New Rule…



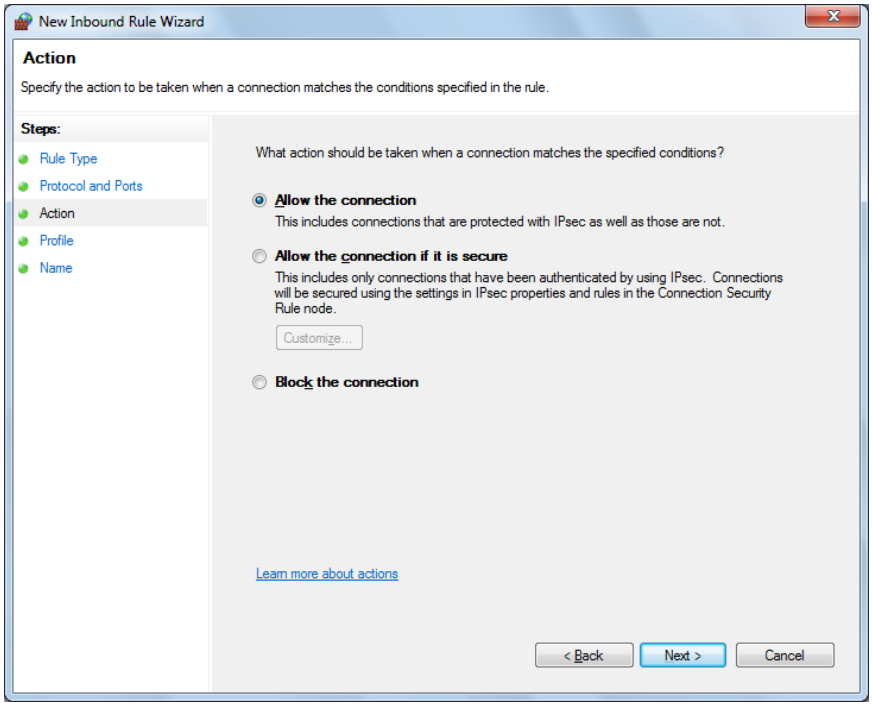
3. In New Inbound Rule Wizard menu
   a. Select **Port** and click **Next**



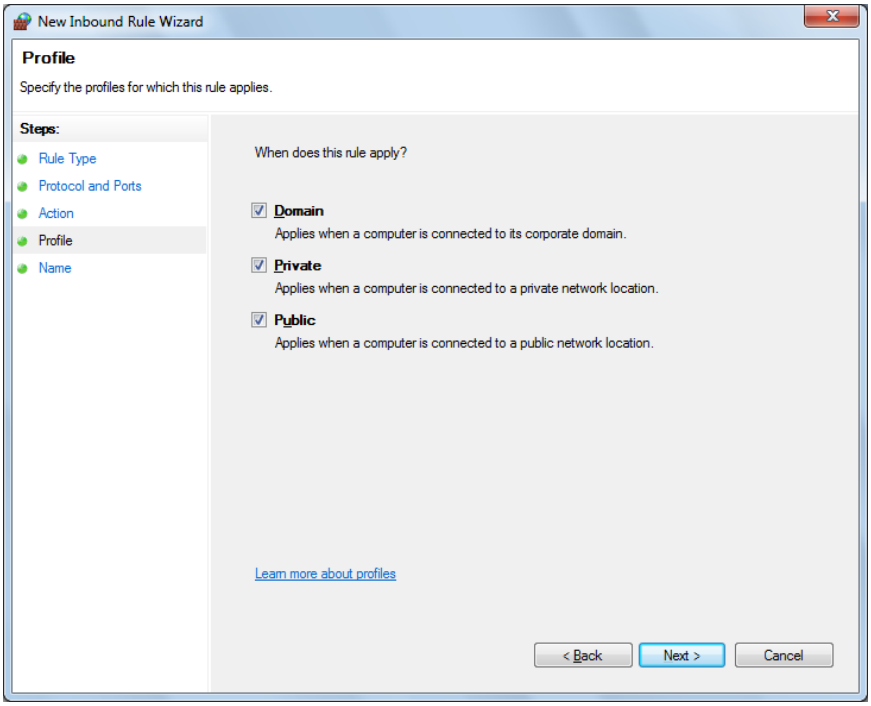
b. Select **TCP** as rule apply and enter 5000 for **Specific local ports** and click **Next**



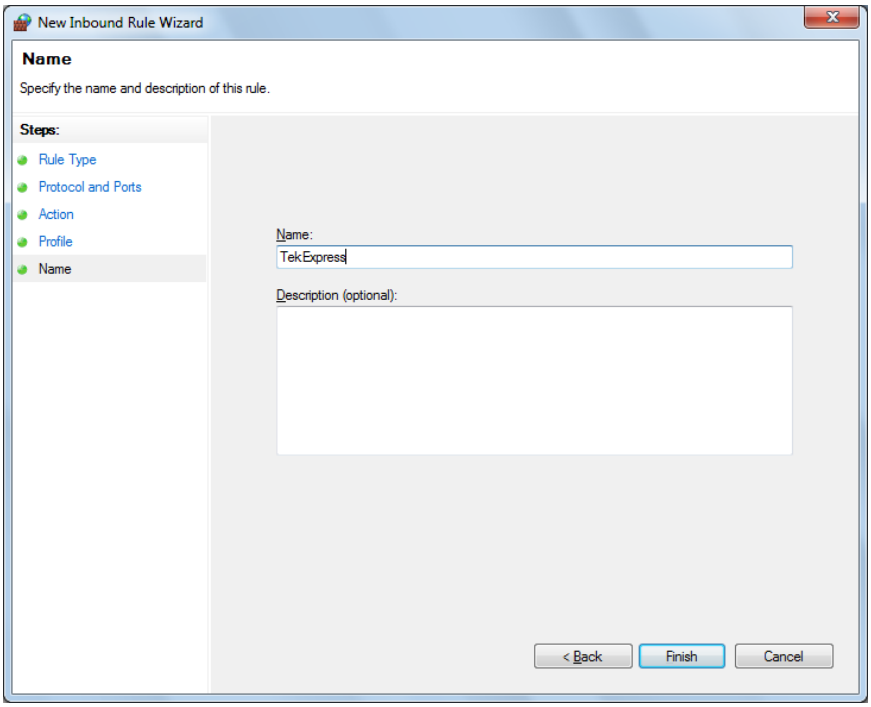
c. Select **Allow the connection** and click **Next**



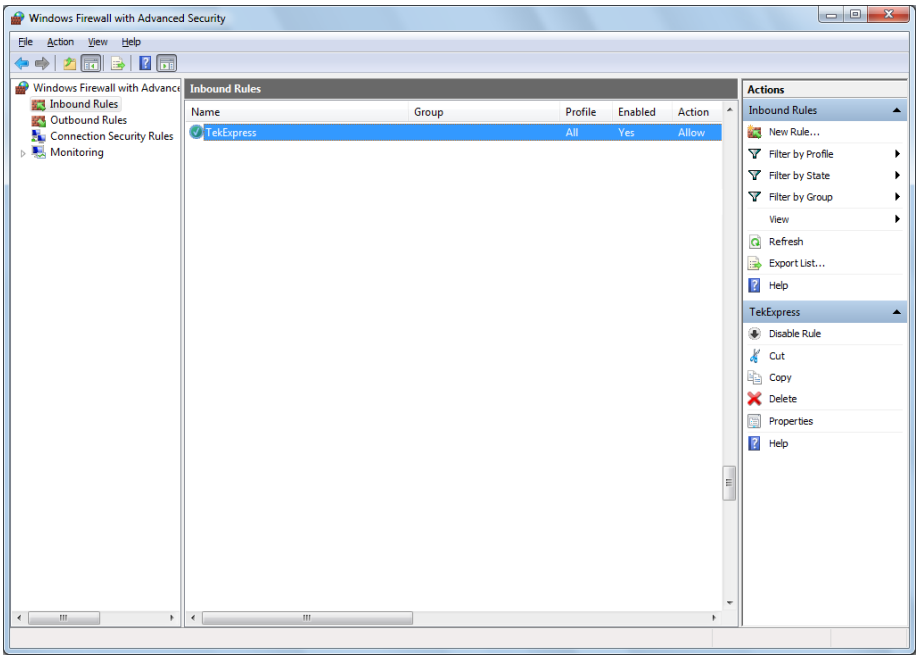
d. Select **Domain, Private, Public** and click **Next**



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

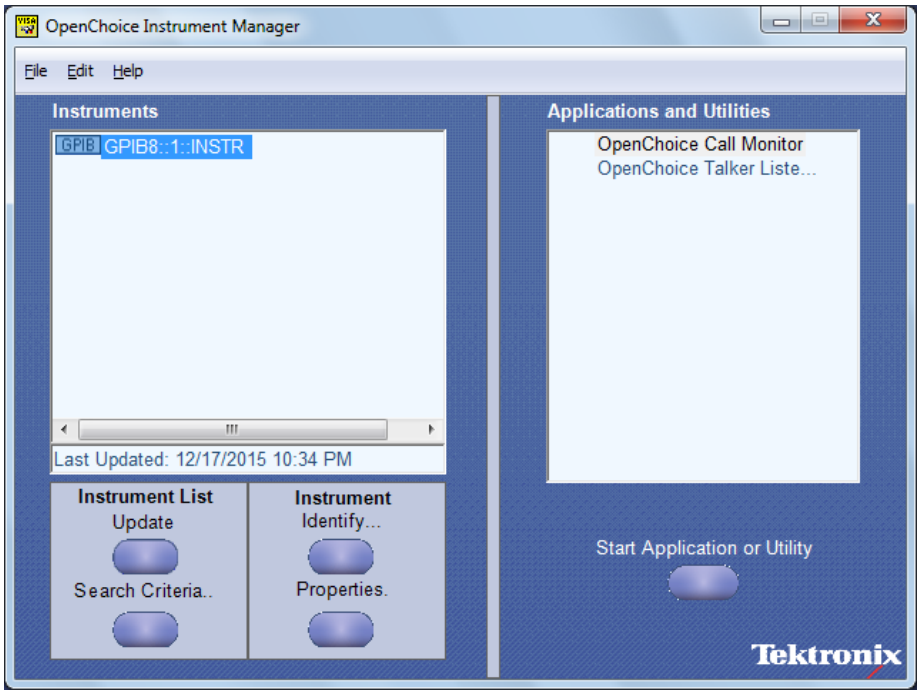


4. Check whether the Rule name is displayed in **Windows Firewall with Advanced Security menu** > **Inbound Rules**



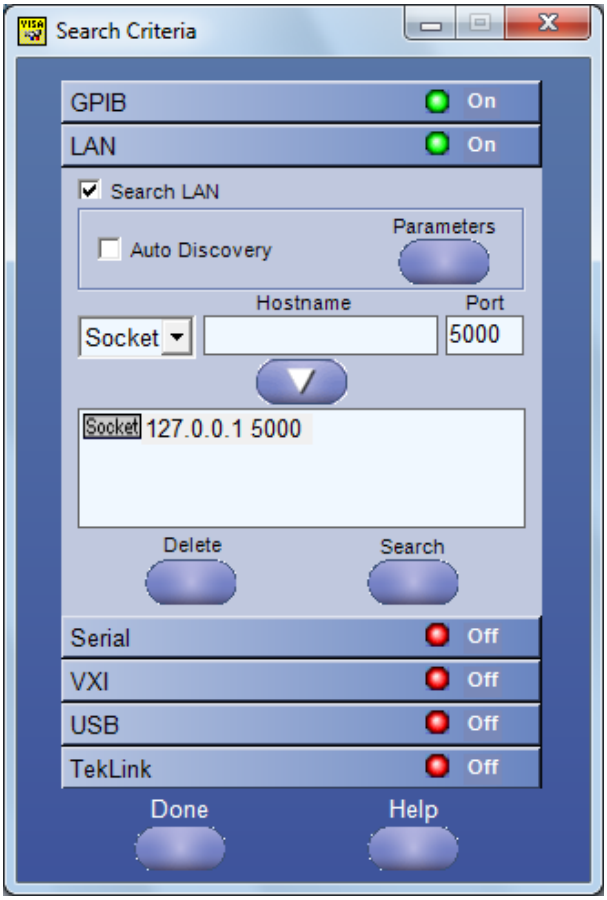
TekVISA configuration

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

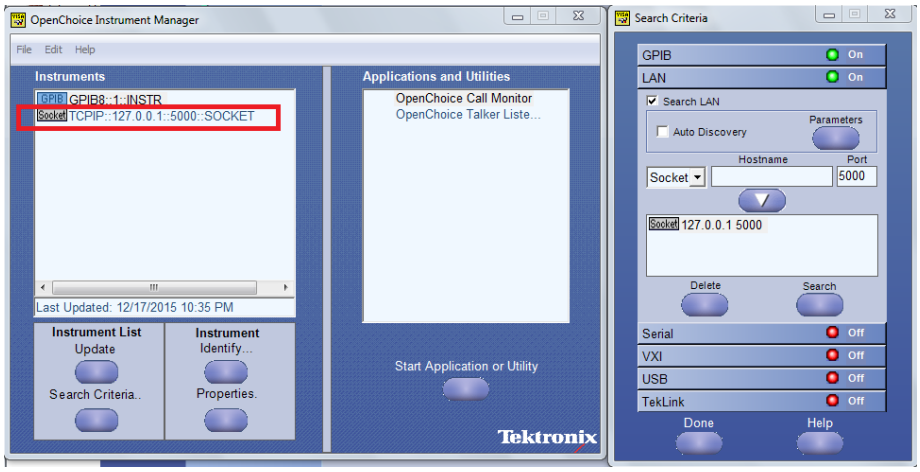


2. Click **Search Criteria**. In Search Criteria menu, click **LAN** to Turn-on. Select **Socket** from the drop-down list, enter the IP address of the TekExpress device in **Hostname** and type **Port** as 5000. Click  to configure the IP address with Port.

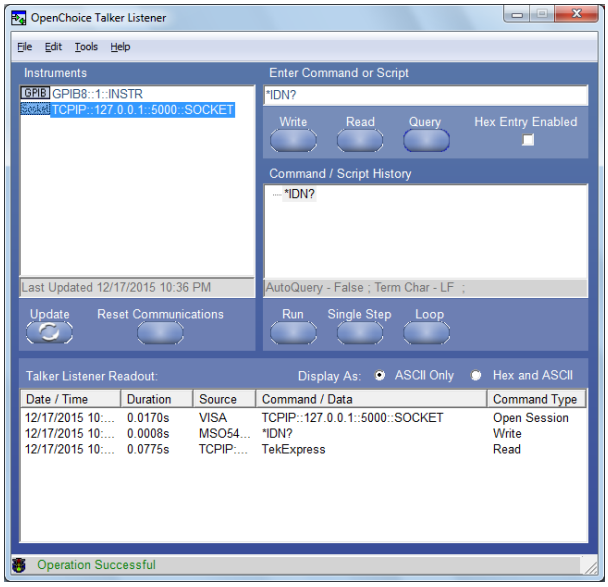
   Enter the Hostname as 127.0.0.1 if the TekVISA and TekExpress application are in the same system, else enter the IP address of the TekExpress application system.



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



4. Double-click **OpenChoice Talker 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 oscilloscope.

Syntax TEKEXP:*IDN?\n

Inputs NA

Outputs Returns active TekExpress application name running on the oscilloscope.

TIP. *Click here for examples.*

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

TIP. *Click here for examples.*

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

TIP. *Click here for examples.*

TEKEXP:ACQUIRE_MODE?

This command queries the acquire mode type.

Syntax TEKEXP:ACQUIRE_MODE?\n

Inputs NA

Outputs {LIVE | PRE-RECORDED}

TIP. *Click here for examples.*

TEKEXP:EXPORT

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

Syntax	Outputs
TEKEXP:EXPORT REPORT\n	Returns the report file in bytes
TEKEXP:EXPORT WFM,"<FileName>"\n	Returns the specified waveform file in bytes
TEKEXP:EXPORT IMAGE,"<FileName>"\n	Returns the specified image file in bytes

Inputs FileName - Specifies the file name

TIP. *Click here for examples.*

TEKEXP:INFO?

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

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

TIP. *Click here for examples.*

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

TIP. *Click here for examples.*

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

TIP. *Click here for examples.*

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>

TIP. *Click here for examples.*

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.

NOTE. *This command returns the list of items within double quotes (""). Iterate the receive procedure until the list ends with double quotes otherwise the next query commands won't work as expected.*

Inputs InstrumentType

TIP. *Check Command parameters list for InstrumentType parameters.*

TIP. *Click here for examples.*

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

TIP. *Click here for examples.*

TEKEXP:MODE?

This command queries the execution mode type.

Syntax TEKEXP:MODE?\n

Inputs NA

Outputs {COMPLIANCE | USER-DEFINED}

TIP. *Click here for examples.*

TEKEXP:POPUP

This command sets the response to the active popup shown in the application.

Syntax TEKEXP:POPUP "<PopupResponse>"\n

Inputs PopupResponse

Outputs NA

TIP. *Click here for examples.*

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.

TIP. *Click here for examples.*

TEKEXP:REPORT

This command generates the report for the current session.

Syntax TEKEXP:REPORT GENERATE\n

Inputs GENERATE

Outputs NA

TIP. *Click here for examples.*

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

TIP. *Click here for examples.*

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 [1]

[1] Row number starts from zero.

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* ***Results*** *panel for TestName, ColumnName, and RowNumber parameters.*

TIP. *Click here for examples.*

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.*

TIP. *Click here for examples.*

Outputs NA

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.

TIP. *Click here for examples.*

TEKEXP:SETUP

This command sets the value of the current setup.

Syntax	Outputs
TEKEXP:SETUP DEFAULT\n	Restore to default Setup
TEKEXP: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

TIP. *Click here for examples.*

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

TIP. *Click here for examples.*

TEKEXP:STATE?

This command queries the current setup state.

Syntax	Outputs
TEKEXP:STATE?	RUNNING | PAUSED | WAIT | ERROR | READY | STOPPED
TEKEXP:STATE? SETUP	SAVED | NOT_SAVED

TIP. *Click here for examples.*

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,“<TestName>","<AcquireType>", "<ParameterName>",“<Value>”\n

TEKEXP:VALUE ANALYZE,“<TestName>","<ParameterName>".“<Value>” \n

TEKEXP:VALUE DUTID,“<Value>”\n

Inputs

ParameterName - Specifies the parameter name

TestName - Specifies the test name

AcquireType - Specifies the acquire type

Value - Specifes the value to set

TIP. *Check Command parameters list for ParameterName, AcquireType, and Value parameters.*

Outputs

NA

TIP. *Click here for examples.*

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,“<TestName>", "<AcquireType>","<ParameterName>"\n	Returns the value of Parameter for type ACQUIRE
TEKEXP:VALUE? ANALYZE, “<TestName>","<ParameterName>"\n	Returns the value of Parameter for type ANALYZE
TEKEXP:VALUE? DUTID\n	Returns the DUTID value

Inputs

ParameterName - Specifies the parameter name

TestName - Specifies the test 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.

TIP. *Click here for examples.*

Command parameters list

This section provides the parameters list for the SCPI commands.

Parameters	Description
InstrumentType	Specifies the instrument type. Valid values are: ■ Alternate Real Time Scope ■ Real Time Scope
Value	Specifies the value parameters. ■ For InstrumentType, valid values are: ■ Comment ■ For DUTID, valid values are: ■ Comment

ParameterName and Value for General

Specifies the ParameterName and Value for General. The configuration parameters available are not same for measurements.

Table 17: ParameterName and Value for General

ParameterName	Value
(For Device)	
RecordLength_LSSQ	50 to 500
RecordLength_FSSQ	10 to 500
RecordLength_HSSQ	15 to 1000
RecordLength_EL28_EL29_EL31	2 to 100
RecordLength_EL38	5 to 90
RecordLength_EL40	40 to 2000
RecordLength_EL27	40 to 1000
RecordLength_EL28	25 to 1000
RecordLength_EL21_EL22_EL25	25 to 1000
RecordLength_EL22	9 to 20
RecordLength_EL16_EL17	2 to 20
RecordLength_EL18	2 to 20
RecordLength_Inrush	1 to 200
(For Host / Hub)	
RecordLength_EL33_EL34	2 to 100
RecordLength_EL35	1000 to 1500
RecordLength_EL39	5 to 90

ParameterName	Value
RecordLength_EL41	40 to 2000
RecordLength_EL21_EL23_EL25	25 to 1000
RecordLength_EL22	25 to 1000
RecordLength_EL55	25 to 1000
RecordLength_Droop	2 to 1000
Qualifier	CH1, CH2, CH3, CH4
Data	CH1, CH2, CH3, CH4
Data+	CH1, CH2, CH3, CH4
Data-	CH1, CH2, CH3, CH4
DUT Automation	True or False
Selected Dut	Device
Packet parameter selected probing	■ Single Ended ■ Differential
High speed probing	■ Single Ended ■ Differential
Probe Type: DUT Signal Probing	■ Single Ended ■ Differential ■ Both
Speed Tab	■ Low Speed ■ High Speed ■ Full Speed
Weiver Mask	True or False
Low Speed	True or False
Full Speed	True or False
High Speed	True or False
USBET	True or False
Tektronix	True or False
Pause For Receiver Sensitivity Test	True or False
Near End	True or False
Far End	True or False
Signal Direction	■ Up Stream ■ Down Stream
Test Method	USBET

ParameterName	Value
Test Point	Near End
Tier	Tier1, Tier2, Tier3, Tier4, Tier5, Tier6
Power Condition	■ Self Powered ■ Bus Powered
Ports	1 to 8
Low Speed Bandwidth	350 MHz, 500 MHz, 1 GHz, 1.25 GHz, 2 GHz, 2.5 GHz, 3 GHz, 3.5 GHz, 4 GHz, 5 GHz, 6 GHz, 7 GHz, 8 GHz, 9 GHz, 10 GHz, 11 GHz, 12 GHz, 12.5 GHz, 13 GHz, 14 GHz, 15 GHz, 16 GHz, 17 GHz, 18 GHz, 19 GHz, 20 GHz, 21 GHz, 22 GHz, 23 GHz, 24 GHz, 25 GHz, 33 GHz
Full Speed Bandwidth	350 MHz, 500 MHz, 1 GHz, 1.25 GHz, 2 GHz, 2.5 GHz, 3 GHz, 3.5 GHz, 4 GHz, 5 GHz, 6 GHz, 7 GHz, 8 GHz, 9 GHz, 10 GHz, 11 GHz, 12 GHz, 12.5 GHz, 13 GHz, 14 GHz, 15 GHz, 16 GHz, 17 GHz, 18 GHz, 19 GHz, 20 GHz, 21 GHz, 22 GHz, 23 GHz, 24 GHz, 25 GHz, 33 GHz
High Speed Bandwidth	2 GHz, 2.5 GHz, 3 GHz, 3.5 GHz, 4 GHz, 5 GHz, 6 GHz, 7 GHz, 8 GHz, 9 GHz, 10 GHz, 11 GHz, 12 GHz, 12.5 GHz, 13 GHz, 14 GHz, 15 GHz, 16 GHz, 17 GHz, 18 GHz, 19 GHz, 20 GHz, 21 GHz, 22 GHz, 23 GHz, 24 GHz, 25 GHz, 33 GHz
Report Update Mode	■ 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	■ Web Archive (*.mht;*.mhtml) ■ PDF (*.pdf;) ■ CSV (*.csv;)
View Report After Generating	TRUE or FALSE
Create report at the end	■ Included ■ Excluded
DUTID Comment	User comment
Number of retries for instrument IO errors	0 to 5
On Failure Rerun	TRUE or FALSE
Number of Reruns On Failure	1 to 100

ParameterName	Value
Time between retries (seconds)	5 to 60
Timer Warning Info Message Popup	■ "True" ■ "FALSE"
Timer Warning Info Message Popup Duration	0 to 20
Timer Error Message Popup	■ "True" ■ "False"
Timer Error Message Popup Duration	0 to 20
On Failure Stop and Notify	TRUE or FALSE

Table 18: ParameterName and Value for Acquire

Test Name	AcquireType	ParameterName	Value
Eye Diagram	LSSQ	Record Length	50-500k
EOP Width			
Signal Rate			
Edge Monotonicity			
Cross Over			
Consecutive Jitter			
Paired JK Jitter		Sample Rate	1.25Gsps
Paired KJ Jitter			
Rising Edge Rate			
Falling Edge Rate			
Edge Rate Match			
Rise Time			
Fall Time			

Test Name	AcquireType	ParameterName	Value
Eye Diagram	FSSQ	Record Length	10-500k
EOP Width			
Signal Rate			
Edge Monotonicity			
Cross Over			
Consecutive Jitter			
Paired JK Jitter		Sample Rate	6.25Gsps
Paired KJ Jitter			
Rising Edge Rate			
Falling Edge Rate			
Edge Rate Match			
Rise Time			
Fall Time			
Eye Diagram	HSSQ	Record Length	15-1000k
EOP Width			
Signal Rate			
Edge Monotonicity			
Cross Over			
Consecutive Jitter			
Paired JK Jitter		Sample Rate	12.5Gsps
Paired KJ Jitter			
Rising Edge Rate			
Falling Edge Rate			
Edge Rate Match			
Rise Time			
Fall Time			
EL28_EL29_EL31(Reset Time, K Duration, Termination Time) **(For Device)**	EL28_EL29_EL31	Record Length	2-100k
		Sample Rate	500ksps
Suspend **(For Device)**	EL38	Record Length	5k-90k
		Sample Rate	12.5Msps
Resume **(For Device)**	EL40	Record Length	40-2000k
		Sample Rate	250Msps
Reset From HS **(For Device)**	EL27	Record Length	40-1000k
		Sample Rate	5Msps
Reset From Suspend**(For Device)**	EL28	Record Length	25-1000k
		Sample Rate	5Msps

Test Name	AcquireType	ParameterName	Value
EL21_EL22_EL25(Sync Bit, Inter Packet Gap, EOP Width) **(For Device)**	EL21_EL22_EL25	Record Length	25-1000k
		Sample Rate	12.5Gsps
EL22(Inter Packet Gap) **(For Device)**	EL22	Record Length	9-20k
		Sample Rate	12.5Gsps
EL16_EL17(Squelch, Rx Sensitivity) **(For Device)**	EL16_EL17	Record Length	NA
		Sample Rate	NA
EL18(Minimum Sync Field) **(For Device)**	EL18	Record Length	NA
		Sample Rate	NA
Inrush Current **(For Device)**	Inrush	Record Length	1-200k
		Sample Rate	1Msps
Inrush Current **(For Device)**	Inrush	Device Type	■ Hot Plug Attached ■ Low Power Configure ■ Low Power Resume ■ High Power Configure ■ High Power Unconfigure ■ High Power Resume
		VBus Channel	CH1, CH2, Ch3, CH4
		Trigger Channel	CH1, CH2, Ch3, CH4
EL33_EL34 **(For Host / Hub)**	EL33_34	Record Length	2-100k
		Sample Rate	500ksps
EL35(JK to SOF Time) **(For Host / Hub)**	EL35	Record Length	1000-1500k
		Sample Rate	12.5Msps
Suspend **(For Host / Hub)**	EL39	Record Length	5k-90k
		Sample Rate	12.5Msps
Resume **(For Host / Hub)**	EL41	Record Length	40-2000k
		Sample Rate	250Msps
EL21_EL23_EL25 **(For Host / Hub)**	EL21_EL23_EL25	Record Length	25-1000k
		Sample Rate	12.5Gsps
EL22(Inter Packet Gap) **(For Host / Hub)**	EL22	Record Length	25-1000k
		Sample Rate	12.5Gsps
EL55(Inter Packet Gap) **(For Host / Hub)**	EL55	Record Length	25-1000k
		Sample Rate	12.5Gsps
Droop **(For Host / Hub)**	Droop	Record Length	2-1000k
		Sample Rate	25Msps
		Vbus	CH1, CH2, Ch3, CH4
		Trigger	CH1, CH2, Ch3, CH4

Table 19: ParameterName and Value for Analyze

Test Name	ParameterName	Value
Eye Diagram EOP Width Signal Rate Edge Monotonicity Cross Over Consecutive Jitter Paired JK Jitter Paired KJ Jitter Rising Edge Rate Falling Edge Rate Edge Rate Match Rise Time Fall Time	Supported Probing	■ Single Ended ■ Differential
	Test Method	■ USBET ■ Tektronix

Examples

This section provides the examples for the SCPI commands.

Example	Description
TEKEXP:*IDN?\n	It returns the active TekExpress application name running on the scope.
TEKEXP:*OPC?\n	It returns the last command execution status.
TEKEXP:ACQUIRE_MODE PRE-RECORDED\n	It sets the acquire mode as pre-recorded.
TEKEXP:ACQUIRE_MODE?\n	It returns LIVE when acquire mode is set to live.
TEKEXP:EXPORT REPORT\n	It returns the report file in bytes. This can be written into another file for further analysis.
TEKEXP:INFO? REPORT\n	It returns “100,”ReportFileName.mht”", when 100 is the filesize in bytes for the filename ReportFileName.
TEKEXP:INFO? WFM\n	It returns “100,“WfmFileName1.wfm””;“200,“WfmFileName2.wfm”" when 100 is the filesize in bytes for the filename WfmFileName1.wfm and 200 is the filesize in bytes for the filename WfmFileName2.wfm.
TEKEXP:INSTRUMENT "Real Time Scope",MSO58 (GPIB8::1::INSTR)\n	It sets the instrument value as MSO58 (GPIB8::1::INSTR) for the selected instrument type Real Time Scope.
TEKEXP:INSTRUMENT? “Real Time Scope”\n	It returns "MSO56 (GPIB8::1::INSTR), when MSO56 (GPIB8::1::INSTR)" is the selected instrument for the instrument type Real Time Scope.
TEKEXP:LASTERROR?\n	It returns ERROR: INSTRUMENT_NOT_FOUND, when no instrument is found.
TEKEXP:LIST? DEVICE\n	It returns "TX-Device,RX-Device" when TX-Device, RX-Device are the available device.

Example	Description
TEKEXP:LIST? INSTRUMENT,"Real Time Scope"\n	It returns "MSO58 (GPIB8::1::INSTR),MSO56 (TCPIP::134.64.248.91::INSTR)" when MSO58 (GPIB8::1::INSTR), MSO56 (TCPIP::134.64.248.91::INSTR) are the list of available instruments.
TEKEXP:MODE COMPLIANCE\n	It sets the execution mode as compliance.
TEKEXP:MODE?\n	It returns COMPLIANCE when the execution mode is compliance.
TEKEXP:POPUP “OK”\n	It sets OK as the response to active popup in the application.
TEKEXP:POPUP?\n	It returns "OK", when OK is the active popup information shown in the application.
TEKEXP:REPORT GENERATE\n	It generates report for the current session.
TEKEXP:REPORT? “Scope Model”\n	It returns "MSO54" when MSO54 is the scope model.
TEKEXP:REPORT? “DUT ID”\n	It returns "DUT001" when DNI_DUT001 is the DUT ID.
TEKEXP:RESULT? "Period using SCOPE (Acquire-Analyze Combined)"\n	It returns Pass when the test result is Pass.
TEKEXP:RESULT? "Period using SCOPE (Acquire-Analyze Combined)","Margin",1\n	It returns "L:-50.000ps H:2000.000ps" when L:-50.000ps H:2000.000ps is the value.
TEKEXP:SELECT DEVICE, TX_Device, TRUE\n	It selects TX_Device
TEKEXP:SELECT? DEVICE\n	It returns "TX-Device" when TX-Device is the selected device type.
TEKEXP:SETUP DEFAULT\n	It restores the application to default setup.
TEKEXP:STATE STOP\n	It stops the test execution.
TEKEXP:STATE?\n	It returns as READY when the application is ready to run next measurement.
TEKEXP:STATE? SETUP\n	It returns as NOT_SAVED when the current setup is not saved.
TEKEXP:VALUE GENERAL,"Signal Type", "N1N0"\n	It sets the signal type parameter value to N1N0.
TEKEXP:VALUE? GENERAL,"Signal Type"\n	It returns "N1N0" when N1N0 is the Signal Type value.

Reference

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 is 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
4	UNLOCK	The server is not locked; lock the server before performing the operation
0	NULL	Nothing

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".

HSETT controller

Setting up controller PC for automated DUT test mode

Pre-requisites for the controller PC/Laptop:

- Windows 7 64-bit OS with at least one USB3.0 port
- USB-IF supplied xHCI HSETT v1.2.0.0 or later installed

Installation:

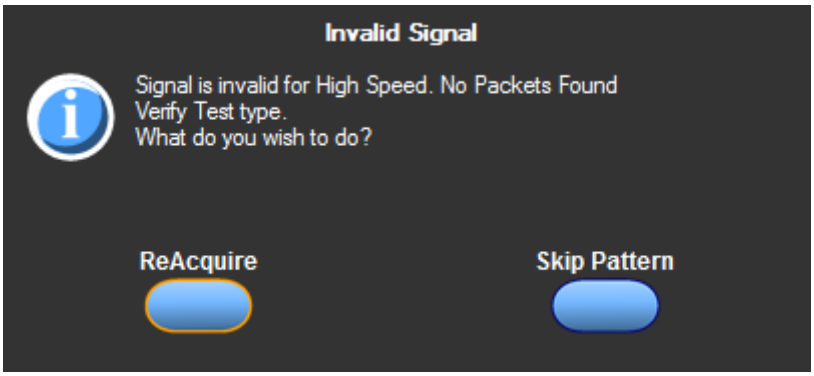
1. Use the Tektronix HSETT Controller Installer to install HSETT Controller on the controller PC/Laptop.
2. Follow the installation instructions.
3. Note the IP address of the controller PC to be provide to TekExpress USB2.

Verifying HSETT Controller installation:

1. Ensure that Tektronix HSETT Controller Service appears in Windows Service Manager (using services.msc) or Windows Task Manager.
2. Ensure path C:\Program Files\Tektronix\TekApplication\Tektronix HSETT Controller\ is created.

Signal validation

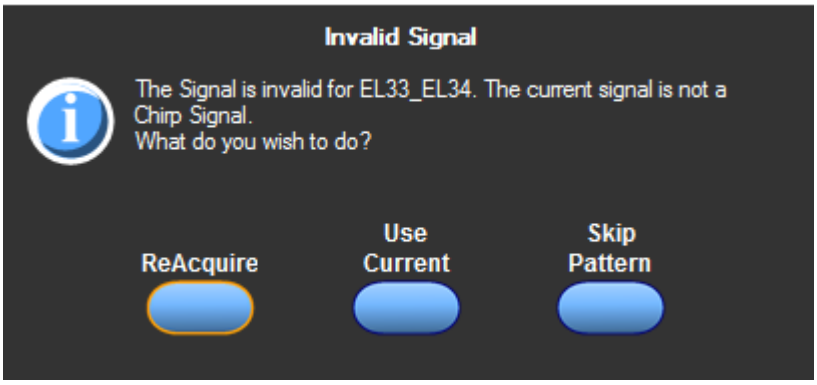
TekExpress USB2 automatically performs signal validation before analyzing and is based on the test method selected. For Test Method selected as USBET, signal validation is done by USBET algorithms. If an invalid signal is detected, application shows the following message dialog box:



■ Click **ReAcquire** to start the acquisition again.

■ Click **Skip Pattern** to skip all the tests associated with that test pattern. The rest of the selected measurements continue.

For Test Method selected as Tektronix, signal validation is done by Tektronix algorithms. If an invalid signal is detected, application shows the following message dialog box:



■ Click **ReAcquire** to start the acquisition again.

■ Click **Use Current** to continue with the currently acquired waveform.

■ Click **Skip Pattern** to skip all the tests associated with that test pattern. The rest of the selected measurements continue.

Index

M

N

O

P

R

S

T

U

W