# TekExpress® USB
# USB 3.0 Automated Test Solution Software

## Printable Application Help


077-0853-00

Tektronix

# TekExpress® USB
# USB 3.0 Automated Test Solution Software

## Printable Application Help

www.tektronix.com
077-0853-00

Tektronix

TekExpress USB 3.0 Application Help, 076-0321-00.

## 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.tektronix.com to find contacts in your area.

# Table of Contents

## Getting help and support



## Getting started



## Operating basics

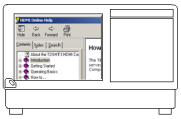
## Reference

# Index

# Related documentation

The following manuals are available as part of the TekExpress® USB 3.0 Automated Test and Compliance Solution documentation set.

**Table 1: Product documentation**

| Item | Purpose | Location |
|---|---|---|
| Help | Application operation and User Interface help |  |
| PDF of the help | Printable version of the compiled help |  www.Tektronix.com<br>PDF file that ships with USB-TX software distribution (*USB-TX-Automated-Test-Solution-Software-Printable-Help-EN-US.pdf*). |
| *DPOJET SuperSpeed (USB 3.0) Setup Library Methods of Implementation (MOI) for Verification, Debug and Characterization.* | Detailed information on test setup and execution | PDF file that ships with USB-TX software distribution |

### See also

Technical support (see page 2)

# 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 the setup files for all the instruments used and the application
- If possible, save the TekExpress setup files, log.xml, *.TekX (session files and folders), and status messages text file
- If possible, save the waveform on which you are performing the measurement as a .wfm file

# Minimum system requirements

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

**Table 3: System requirements**

| | |
|---|---|
| **Oscilloscope** | See Supported instruments (see page 5) |
| **Processor** | Same as the oscilloscope |
| **Operating System** | Same as the oscilloscope:<br>■ Windows 7 (64-bit only) SP1 Windows 7 user account settings (see page 4) |
| **Memory** | Same as the oscilloscope |
| **Hard Disk** | Same as the oscilloscope |
| **Display** | Super VGA resolution or higher video adapter (800 x 600 minimum video resolution for small fonts or 1024 x 768 minimum video resolution for large fonts). The application is best viewed at 96 dpi display settings [1] |
| **Firmware** | ■ TekScope 6.7.4.3 and later (for Windows 7) |

**Table 3: System requirements (cont.)**

| | |
|---|---|
| **Software** | ■ TekExpress Framework (version 3.0.x or later) installed.<br>■ Microsoft .NET 4.0 Framework<br>■ SigTest 3.2.x or later installed.<br>■ DPOJET Jitter and Eye Analysis Tool (version 6.0.x or later) with Advanced Jitter and Eye analysis (DJA option) installed.<br>■ (Optional- required for USB3 testing) SuperSpeed USB DPOJET Module (DPOJET option USB3)<br>■ Microsoft Internet Explorer 7.0 SP1 or later, or other Web browser for viewing reports.<br>■ Adobe Reader software 7.0 or later for viewing portable document format (PDF) files.<br>■ (Optional) Serial Data Link Analysis (SDLA) software for Channel De-Embed, for custom filter development. |
| **Other Devices** | ■ Microsoft compatible mouse or compatible pointing device.<br>■ Two USB ports (four USB ports recommended).<br>■ PCI-GPIB or equivalent interface for instrument connectivity [2]. |

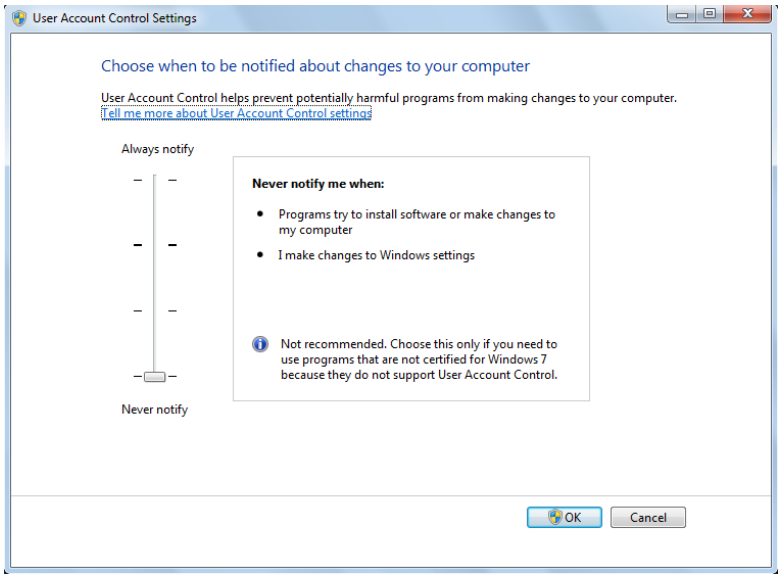
1 If TekExpress is running on an instrument that has a video resolution less than 800x600, connect and configure a second monitor to the instrument.

2 If TekExpress is installed on a Tektronix oscilloscope, TekExpress cannot use the virtual GPIB port to communicate with oscilloscope applications. If using external devices for instrument connectivity (such as USB-GPIB adapters or equivalent), enable the Talker Listener utility in the GPIB menu of the Tektronix MSO/DPO/DSA oscilloscope.

# Windows 7 user account settings

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

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




### See also

Supported oscilloscopes (see page 5)

# Supported instruments

### Table 4: Required equipment

| Resource | Model supported |
|---|---|
| Real-time oscilloscope | Compliance testing: Tektronix MSO/DPO/DSA71000 Series oscilloscopes with a bandwidth of 12 GHz or greater<br>Normative measurements: MSO/DPO/DSA70804 Series oscilloscopes |
| Probes | Two TCS-SMA cables<br>P7313SMA differential probe<br>P7500 differential probe |
| Host test fixtures | TF-USB3-A-P (for best signal quality)<br>For more mechanical flexibility use TF-USB-B-R (with included 13 cm USB 3.0 Cable - Part number 174-5772-00).<br>For precision De-embed of TF-USB3-A plug fixture, order TF-USB3-AB-KIT (includes Cal Kit).<br>TF-USB3-KIT (includes short USB 3.0 cable)<br>USB-IF fixtures [1] |

**Table 4: Required equipment (cont.)**

| Resource | Model supported |
|---|---|
| Device test fixtures | TF-USB3-A-R (includes short USB 3.0 Cable)<br>USB-IF fixtures [1] |
| Tektronix AWG/AFG instruments | Tektronix AWG7102, AWG7122 Series with options 6,8<br>Tektronix AWG5014B, AWG5014C<br>Tektronix AFG3252, AFG3252C, AFG3251, AFG3251C, AFG3102, AFG3102C, AFG3101, AFG3101C |

1 Available through USB-IF.

### See also

Minimum system requirements (see page 3)

# Install the software

Use the following steps to obtain the latest USB-TX software from the Tektronix Web site and install on any compatible instrument running Microsoft Windows 7 (64-bit). See Minimum System Requirements (see page 3) for details.

1. Close all applications (including the TekScope application).
2. Go to the www.tek.com Web site and locate the **Downloads** fields.
3. Enter **tekexpress usb** in the *Model or Keyword* field, select **Software** from the *Select Download Type* list, and click **GO**.
4. Select the latest version of software. Follow instructions to download the software file.
5. Copy or download the USB-TX installer executable file to the oscilloscope.
6. Double-click the installer .exe file to extract the installation files and launch the InstallShield Wizard. Follow the on-screen instructions.

   Software is installed at `C:\Program Files (x86)\Tektronix\TekExpress\TekExpress USB`
7. Verify application installation (see page 7)

### See also

Minimum system requirements (see page 3)
Supported instruments (see page 5)
Required \My TekExpress folder settings (see page 9)

# Verify application installation

To verify the installation was successful:

1. Open the TekScope application.
2. Click the **Analyze** menu.
3. Verify that **TekExpress USB** is listed in the Analyze menu.
4. Click **TekExpress USB** to open the USB-TX application. Verify that the application opens successfully.

### See also

Activate the license (see page 7)
Required \My TekExpress folder settings (see page 9)

# Activate the license

Activate the license using the **Option Installation** wizard in the TekScope application:

1. In the TekScope application menu bar, click **Utilities > Option Installation**.

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

### See also

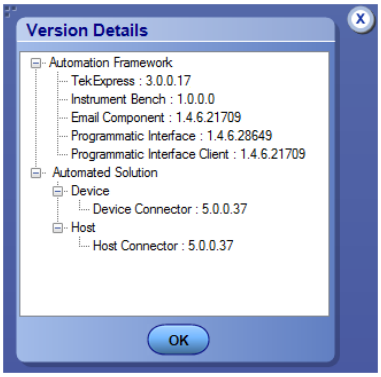
View version and license information (see page 8)
Required \My TekExpress folder settings (see page 9)

# View software version and license information

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

To view version information for USB-TX:

1. In the USB-TX application, click the **Options** button and select **About TekExpress**.
2. Click the View Version Details link to view the version numbers of the installed test suites.




***NOTE.*** *This example shows a typical Version Details dialog box, and may not reflect the actual values as shown when you open this item in the application.*

To view license and option key information:

1. In the TekScope application, select **Help > About TekScope**.
2. Scroll through the **Options** section list to locate **USB: TekExpress USB**.
3. To view the Option key, look below the **Options** list.

### See also

Activate the license (see page 7)
Options menu (see page 19)

# Required \My TekExpress folder settings

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

1. Map the \My TekExpress folder to Drive X (see page 10)
2. Set the \My TekExpress folder permissions (see page 11)

### See also

Application directories and usage (see page 12)
File name extensions (see page 13)

# Map the My TekExpress folder to drive X

The first time you run TekExpress USB, it creates the following folders on the oscilloscope:

- `\My Documents\My TekExpress\USB`
- `\My Documents\My TekExpress\USB\Untitled Session`

You need to map the shared **My TekExpress** folder as drive **X:** on the instrument running the USB-TX application. USB-TX uses this shared folder to save session waveform files and for other application file transfer operations.

To map the My TekExpress folder on the instrument to be drive X:

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

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

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

***NOTE.*** *The My TekExpress folder has the share name format* *<domain><user ID>My TekExpress*.

*If the instrument is not connected to a domain, the share name format is* *<instrument name><user ID>My TekExpress*.

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

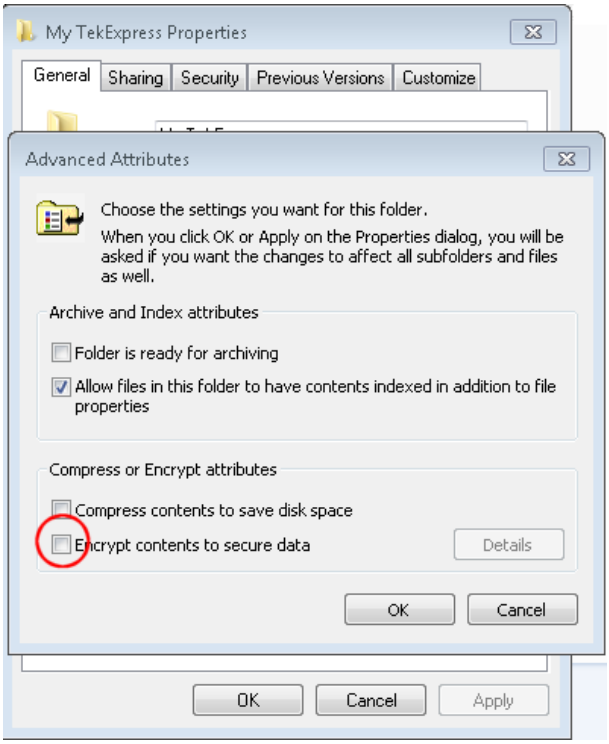
### See also

Set the \My TekExpress folder permissions (see page 11)
Application directories and usage (see page 12)
File name extensions (see page 13)

# Set the \My TekExpress folder permissions

Make sure that the My TekExpress folder has read and write access. Also verify that the folder is not set to be encrypted:

1. Right-click the folder and select **Properties**.
2. Select the **General** tab and then click **Advanced**.
3. In the Advanced Attributes dialog box, make sure that the option **Encrypt contents to secure data** is NOT selected.




4. Click the **Security** tab and verify that the correct read and write permissions are set.

## See also

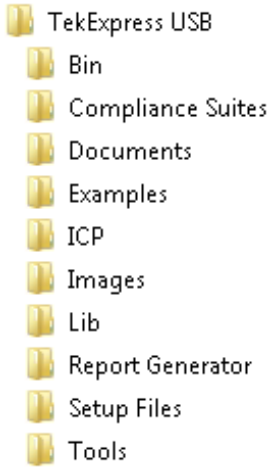
Map the \My TekExpress folder to Drive X (see page 10)
Application directories and usage (see page 12)
File name extensions (see page 13)

# Application directories and their contents

## TekExpress USB application

The TekExpress USB application files are installed at the following location:

```
C:\Program Files (x86)\Tektronix\TekExpress\TekExpress USB
```




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

**Table 5: Application directories and usage**

| Directory names | Usage |
|---|---|
| Bin | Contains USB-TX application libraries |
| Compliance Suites | Contains compliance-specific files |
| Documents | Contains the technical documentation for the USB-TX application |
| Examples | Contains various support files |
| ICP | Contains instrument and USB-TX application-specific interface libraries |
| Lib | Contains utility files specific to the USB-TX application |
| Report Generator | Contains style sheets for report generation |
| Tools | Contains instrument and USB-TX application-specific files |

## See also

View test-related files (see page 40)
File name extensions (see page 13)

# File name extensions

The TekExpress USB application uses the following file name extensions:

| File name extension | Description |
|---|---|
| .TekX | Application session files (the extensions may not be displayed) |
| .py | Python sequence file |
| .xml | Test-specific configuration information (encrypted) files<br>Application log files |
| .wfm | Test waveform files |
| .mht | Test result reports (default)<br>Test reports can also be saved in HTML format (see page 41) |
| .flt | Filter files |
| .xslt | Style sheet used to generate reports |

## See also

View test-related files (see page 40)
Application directories and their contents (see page 12)

# Where test files are stored

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

- `\My Documents\My TekExpress\USB`
- `\My Documents\My TekExpress\USB\Untitled Session`

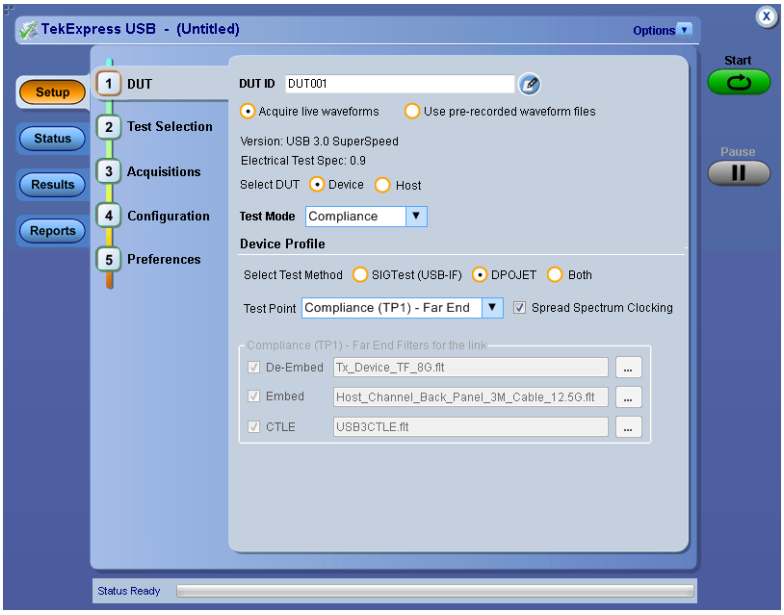
Every time you launch TekExpress USB, an `Untitled Session` folder is created in the `USB` folder. The `Untitled Session` folder is automatically deleted when you exit the `USB` application. To preserve your test session files, save the test setup before exiting the TekExpress application.

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

### See also

Map the \My TekExpress folder to Drive X (see page 10)
Set the \My TekExpress folder permissions (see page 11)
Application directories and usage (see page 12)
File name extensions (see page 13)

# TekExpress® USB key features




Welcome to the TekExpress® USB Automated Test Solution Software application (referred to as TekExpress USB or USB in the rest of the document). TekExpress USB provides an automated, simple, and efficient way to test USB transmitter interfaces and devices consistent to the requirements of the USB 3.0 specifications.

Key features and benefits of TekExpress USB include:

- Comprehensive test coverage; select or deselect individual tests
- Precise debugging and troubleshooting
- Accurate and reliable results
- Intel SigTest is integrated into the TekExpress framework
- Reduced time required to run tests
- Minimizes user intervention when performing time-consuming testing
- SSC Measurements controlled through a single UI control
- Able to select individual filters for de-embedding, embedding, and equalization
- Automatically detect Tektronix TriMode™ P7500 series probes
- De-emphasis measurement uses CP7 pattern
- Automatic compliance pattern toggle from CP1-CP7
- Consolidated report with both SigTest and DPOJET measurement results
- LFPS Vcm-AC measurement support using a Tektronix TriMode Probe

- Configuration panel features:
  - DUT specific AFG/AWG configuration for pattern toggle
  - Automatically save channel waveforms.
  - Support common mode measurements with TriMode P7500 series probes
  - Automatically recover oscilloscope settings when ‘CP0 CP1 CP7 Toggle using’ is set to ‘Do not use’
  - Extended probing configurations (Single Ended (Ch1-Ch3), Single Ended (Ch2-Ch4))
- Complete programmatic interface enables automation scripts to call USB-TX functions

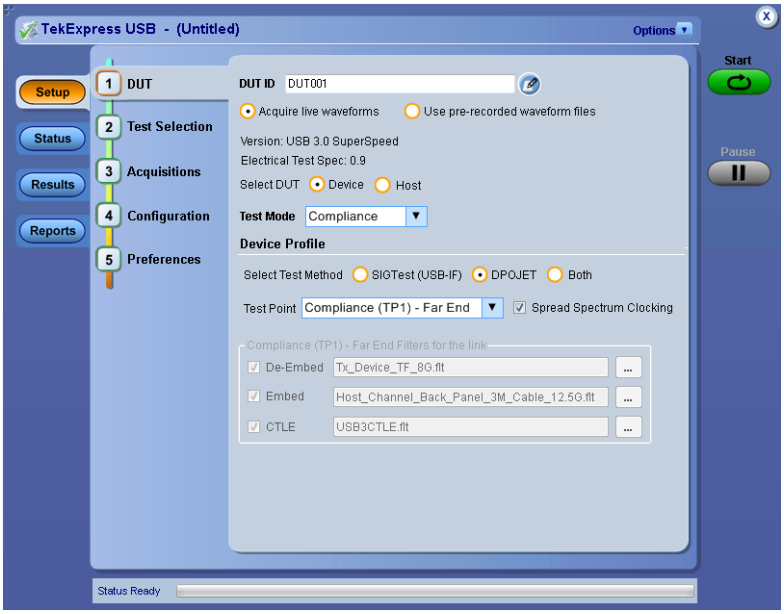
***NOTE.*** *The USB 3.0 near end measurement ‘Vtx-de-ratio’ analyzes using the CP7 pattern.*

# Run the application

To launch the TekExpress USB application, do either of the following:

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

The oscilloscope opens the TekExpress USB application:




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

***NOTE.*** *Do the steps in the Required \My TekExpress folder settings (see page 9) topic before running tests with the USB application for the first time.*

To keep the USB application window on top, select **Keep On Top** from the USB Options menu (see page 19). If the application goes behind the oscilloscope application, click **Analyze > TekExpress USB** to move the application to be in front.

### See also

Required My TekExpress folder settings (see page 9)
Activate the license (see page 7)
Exit the application (see page 18)
Application controls (see page 18)
Application panel overview (see page 26)

# Exit the application

To exit the application, click  on the application title bar. Follow on-screen prompts to save any unsaved session, save test setup files, or exit the application.

***NOTE.*** *Using other methods to exit the application can result in abnormal termination of the application.*

# Application controls

### Table 6: Application controls descriptions

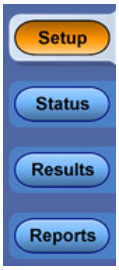
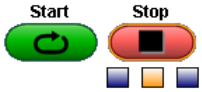
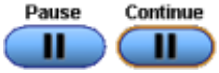
| Item | Description |
|---|---|
| Options menu (see page 19)   | Menu to display global application controls. |
|   Test Panel buttons (see page 26) | Controls that open panels for configuring test settings and options. |
| Start / Stop button   | Use the Start button to start the test run of the measurements in the selected order. If prior acquired measurements have not been cleared, the new measurements are added to the existing set.<br>The button toggles to the Stop mode while tests are running. Use the Stop button to abort the test. |
| Pause / Continue button   | Use the Pause button to temporarily interrupt the current acquisition. When a test is paused, the button name changes to “Continue.” |

**Table 6: Application controls descriptions (cont.)**

| Item | Description |
|---|---|
| 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 (see page 38). |
| 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. |

### See also

Application panel overview (see page 26)

# Options menu overview

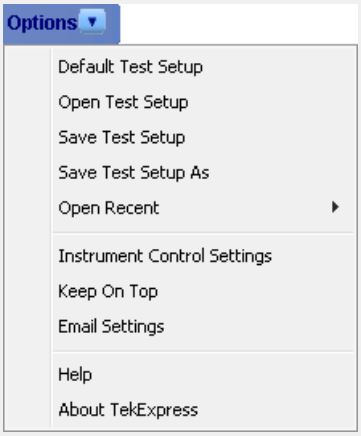
The Options menu is located in the upper right corner of the application.

The Options menu (see page 20) 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 (see page 20) | Detects, lists, and refreshes the connected instruments found on specified connections (LAN, GPIB, USB, and so on) |
| Keep On Top | Keeps the TekExpress USB application on top of other open windows on the desktop |
| Email Settings (see page 23) | Use to configure email options for test run and results notifications |

| Menu | Function |
|---|---|
| Help | Displays the TekExpress USB help |
| About TekExpress | ■ Displays application details such as software name, version number, and copyright<br>■ Provides access to license information (see page 8) for your USB-TX installation<br>■ Provides a link to the Tektronix Web site |

### Options menu



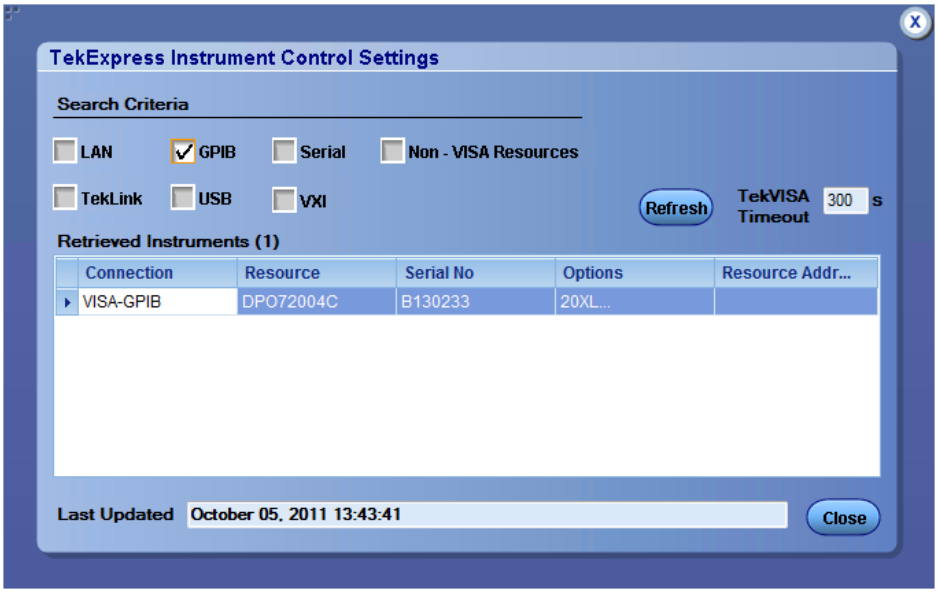

### See also

Application controls (see page 18)

# Instrument control settings

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

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




Use the Instrument Control Settings feature to search for connected instruments (see page 21) and view instrument connection details. Connected instruments displayed here can be selected for use under Global Settings in the test configuration section.

### See also

Options menu overview (see page 19)

# 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.

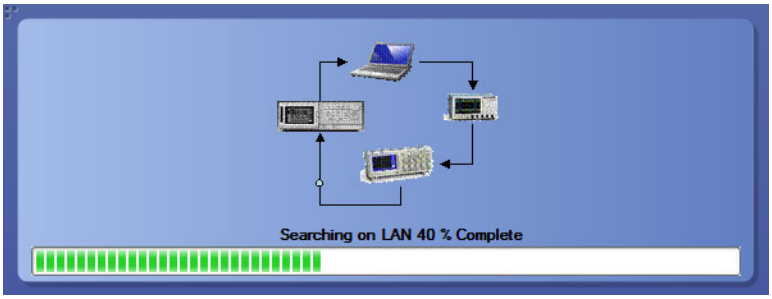
***NOTE.*** *The correct instruments for the current test setup must be connected and recognized by USB-TX before running tests.*

To refresh the list of connected instruments:

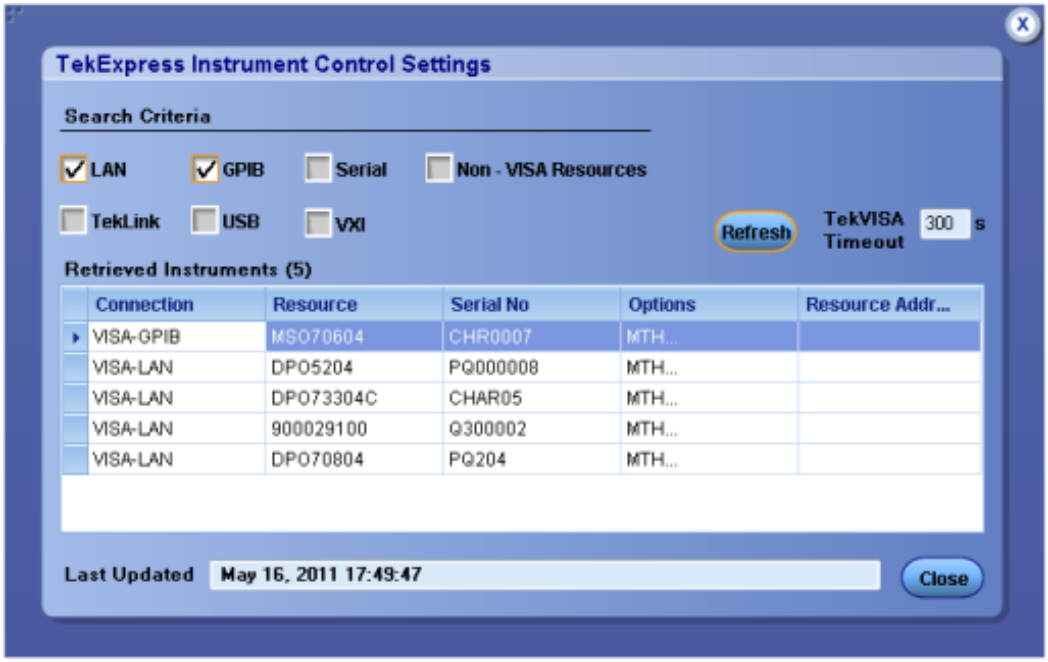
1. From the Options menu, select **Instrument Control Settings**.
2. In the **Search Criteria** section of the Instrument Control Settings dialog box, select the connection types of the instruments for which to search.

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

## See also

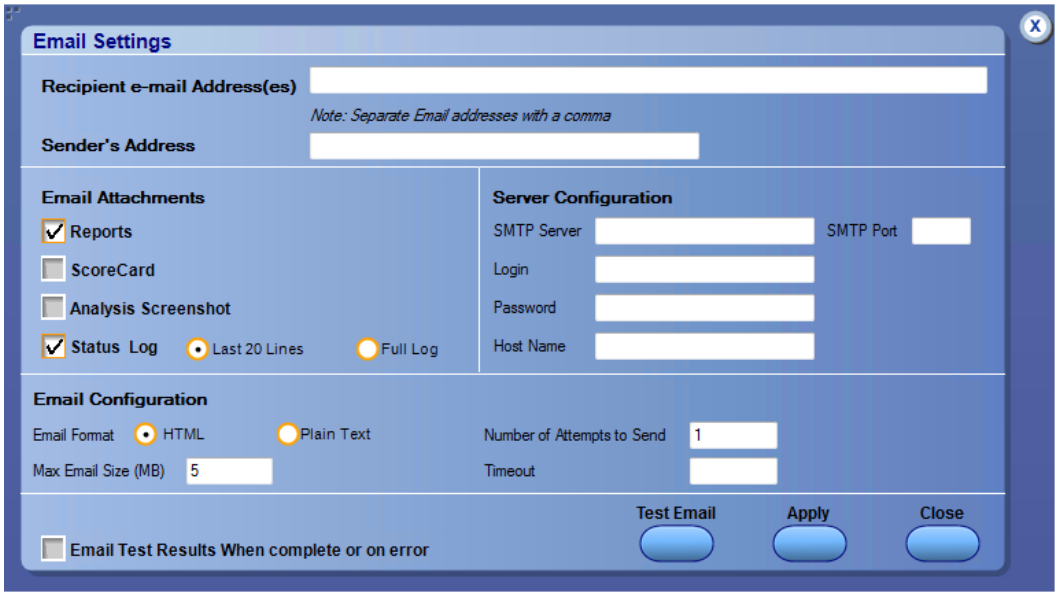
Configuration test parameters (see page 35)
Equipment connection setup (see page 47)

# Email settings

Use the Email Settings utility to configure email notifications (see page 23) to receive notifications when a test completes, produces an error, or fails. Select the type of test session information to include in the notification, such as test reports and test logs, the email message format, and the email message size limit.

Select **Options > Email Settings** to open this dialog box.

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




### See also

Configure email settings (see page 23)
Options menu (see page 19)
Select test notification preferences (see page 36)

# 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 (see page 25) dialog box.
2. (Required) For Recipient email Address(es), enter one or more email addresses to which to send the test notification. To include multiple addresses, separate the addresses with commas.

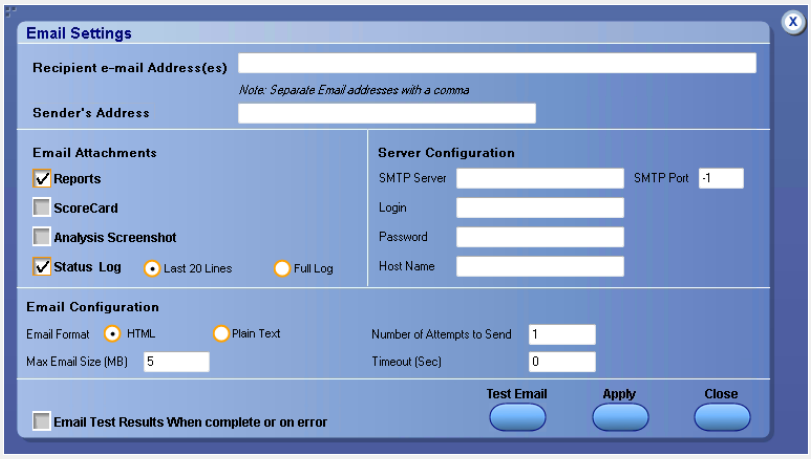
3. (Required) For Sender's Address, enter the email address used by the instrument. This address consists of the instrument name followed by an underscore followed by the instrument serial number, then the @ symbol and the email server used. For example: DPO72016C_B130099@yourcompany.com.
4. (Required) In the Server Configuration section, type the SMTP Server address of the Mail server configured at the client location, and the SMTP Port number, in the corresponding fields.

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

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

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

## Email settings



# Application panels overview

TekExpress USB uses panels to group related configuration, test, and results settings. Click on 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.

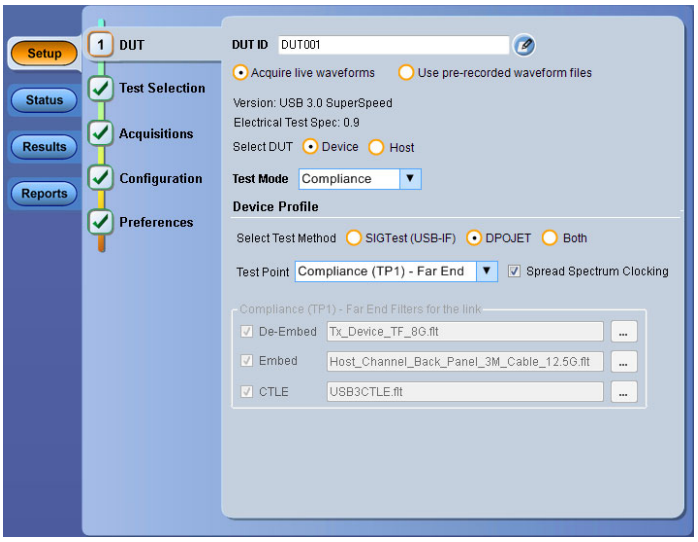



**Table 7: Application panels overview**

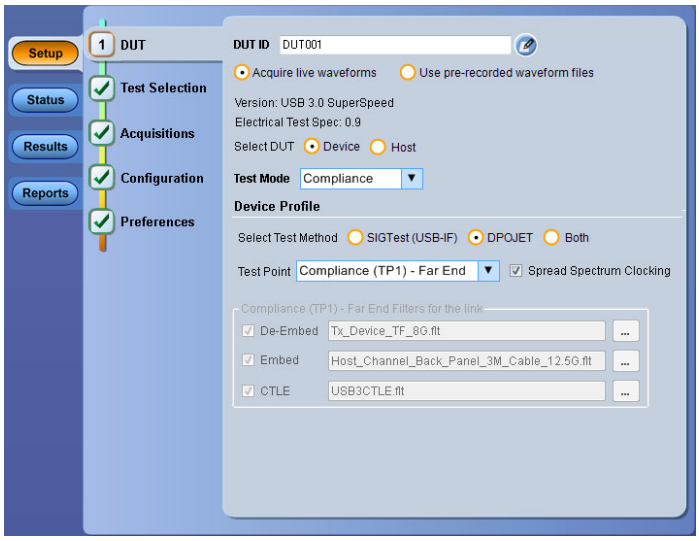
| Panel Name | Purpose |
|---|---|
| Setup (see page 27) | The Setup panel shows the test setup controls. Click the **Setup** button to open this panel.<br>Use this panel to:<br>■ Select DUTparameters (see page 27).<br>■ Select the test(s) (see page 28).<br>■ Set acquisitions parameters (see page 30) for selected tests.<br>■ Configuration test parameters (see page 35)<br>■ Select test notification preferences (see page 36). |
| Status (see page 37) | View the progress and analysis status of the selected tests, and view test logs. |
| Results (see page 38) | View a summary of test results and select result viewing preferences. |
| Reports (see page 40) | Browse for reports, save reports as specific file types, specify report naming conventions, select report content to include (such as summary information, detailed information, user comments, setup configuration, application configuration), and select report viewing options. |

### See also

Application controls (see page 18)

## Setup control overview

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




The tabs on this panel are:

DUT: Set the DUT parameters (see page 27)

Test Selection: Select test(s) (see page 28)

Acquisitions: Select acquisition parameters (see page 30)

Configuration: Set configuration parameters (see page 35)

Preferences: Select test fail notification preferences (see page 36)

## 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**

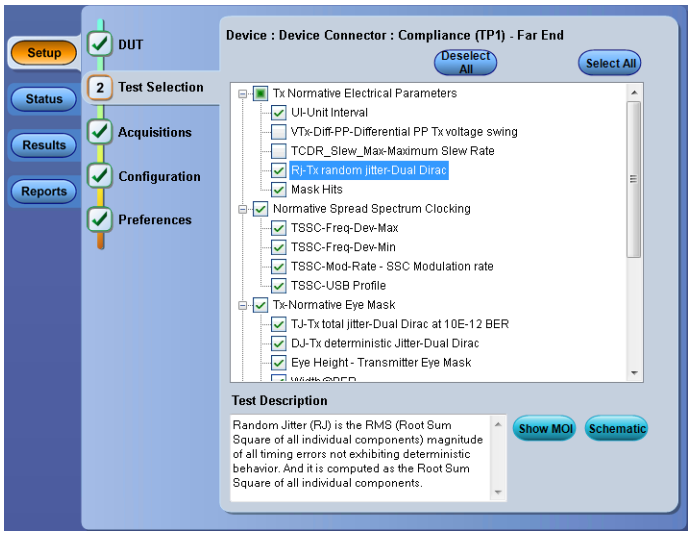
| Setting | Description |
|---|---|
| DUT ID | Adds an optional text label for the DUT to reports. The default value is DUT001. The maximum number of characters is 32.<br>You cannot use the following characters in a ID name: (.,..,...,\,/:?"<>\|*) |
|  Comments icon (to the right of the DUT ID field) | Opens a Comments dialog box in which to enter optional text to add to a report. Maximum size is 256 characters. To enable or disable comments appearing on the test report, see Select report options (see page 41).) |
| 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 (see page 50) |
| Version | Lists the supported USB test specification(s). |
| Electrical Test Spec | Lists the supported electrical test specification(s). |
| Select DUT | Sets the DUT device type to test (Device or Host). |
| Test Mode | ■ Compliance: Preselects tests and parameters needed to meet compliance specifications for the selected device type. Disables the compliance filter controls.<br>■ User Defined: Enables the user to select specific tests and set custom parameters for tests (using the Configuration button). |
| **Device Profile** | |
| Select Test Method | Select the algorithms used to measure and analyze the signal. DPOJET is the default test method. |
| Test point | Select the appropriate test point location from those listed.<br>Only Compliance (TP1) - Far End test point is available when Sigtest or Both are selected. |
| Spread Spectrum Clocking | Select this check box if your DUT supports Spread Spectrum Clocking (SSC). Selects SSC tests based on your DUT configuration. |
| Filters for the link | Lists de-embed, embed, and CTLE filter files used for the current DUT near end and far end test points. Use filters to take cable and fixture signal path length and characteristics into account.<br>Click the ... button to view and select other filter files.<br>These controls are not available when Test Mode is set to Compliance.<br>DeEmbeding and Channel Embedding Overview (see page 112) |

### See also

Select a test (see page 28)

# Select tests

Use the **Test Selection** tab to select USB-TX tests.




### Table 9: Test Selection tab settings

| Setting | Description |
|---|---|
| **Deselect All**, **Select All** buttons | Deselect or select all tests in the list. |
| Tests | Click on a test to select or unselect. Highlight a test to show details in the Test Description pane. All required tests are selected when in Compliance test mode.<br>Measurements are grouped according to standard specifications such as Tx Informative Electrical Parameters (see page 29), Normative Spread Spectrum Clocking (see page 29), Tx Normative Eye Mask (see page 30), and LFPS measurement (see page 30). |
| **Show MOI** button | Opens the USB-TX Method of Implementation (MOI) PDF file. |
| **Schematic** button | Shows an equipment and test fixture setup schematic 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.*

#### Tx Informative Electrical Parameters

Includes UI-Unit Interval, VTx-Diff-PP-Differential PP Tx voltage swing, TCDR_Slew_Max-Maximum Slew Rate, Rj-Tx-random jitter-Dual Dirac, and Mask Hits measurements.

#### Normative Spread Spectrum Clocking

Includes TSSC-Freq-Dev-Max, TSSC-Freq-Dev-Min, TSSC-Mod-Rate-SSC Modulation rate, and TSSC USB Profile measurements.

### Tx-Normative Eye Mask

Includes TJ-Tx total jitter-Dual Dirac at 10E-12 BER, DJ-Tx-deterministic Jitter-Dual Dirac, Eye Height- Transmitter Eye Mask, and Width@BER measurements.

### Low Frequency Periodic Signaling (LFPS) Measurement

Includes LFPS Duty Cycle, LFPS Fall Time, LFPS Rise Time, LFPS TPeriod, LFPS TBurst, LFPS TRepeat, LFPS Vcm-AC, and LFPS VTx-DIFF-PP mesurements. Refer to the USB MOI document, LFPS measurement section, for more information.

***NOTE.*** *The application does not show the oscilloscope cursor1 and 2 for each burst. The analysis is done for the entire acquisition, for all bursts, and then displays the result statistics.*

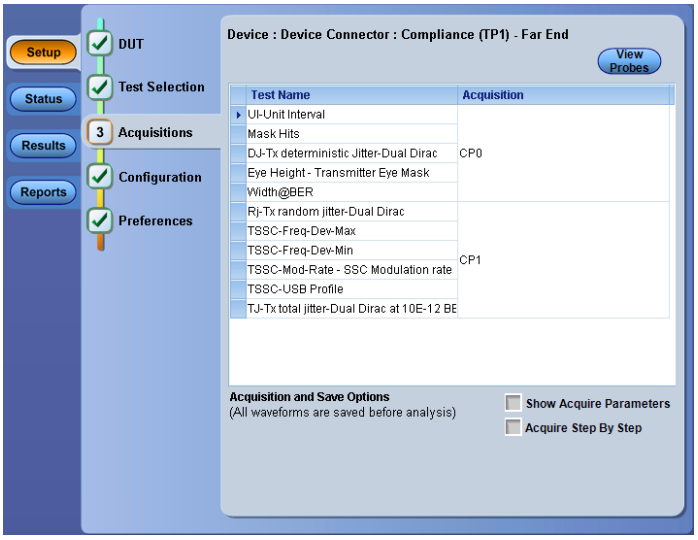
### See also

Set acquisition parameters (see page 30)

# Set acquisition parameters

Use the **Acquisition** tab in the Setup panel to view test acquisition parameters. You also use this tab to load and prerecorded (saved) test session waveform files on which to run tests.

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




Acquisitions tab: using active waveforms

***NOTE.*** *USB-TX acquires all waveforms required by each test group before performing analysis.*

**Table 10: Acquisitions tab settings**

| Setting | Description |
|---|---|
| **View Probes** button | Shows the detected probe configuration. Use the View Probes dialog box to enable or disable probe signal source access in the application. Only available for live waveforms. |
| Show Acquire Parameters | Shows the various parameters related to each acquisition of a test. The parameters listed depend on the selected tests. |
| Acquire Step By Step | Shows a prompt at the end of each acquisition and pauses the application. Click **Continue** to proceed to the next acquisition. |

USB-TX 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:\USB\Untitled Session\<dutid>\<date>_<time>`. Images created for each analysis, CSV files with result values, reports, and other information specific to that particular execution are also saved in this folder.

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

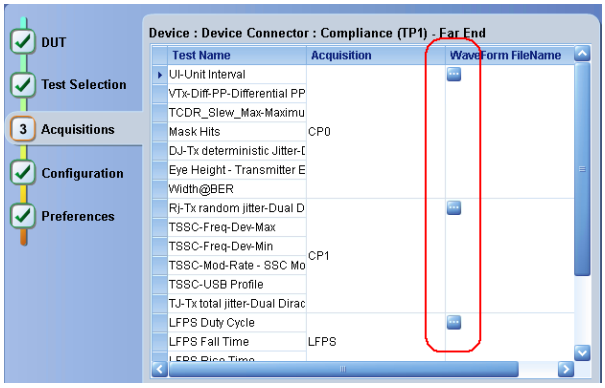
### See also

Running tests on prerecorded saved waveforms (see page 31)

# Running tests on prerecorded (saved) waveforms

To load a saved waveform file:

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



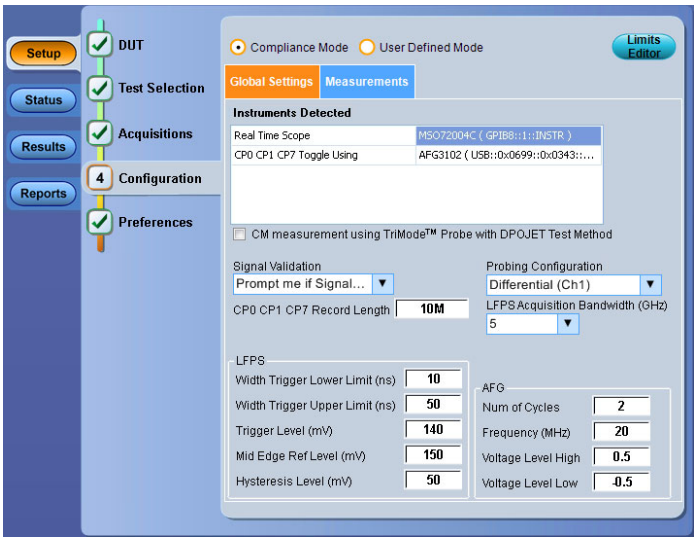

Acquisitions tab: using prerecorded (saved) waveforms

4. Click the browse button ( ) for each test acquisition type (CP0, CP1, LFPS).
5. Navigate to and select the appropriate waveform file(s). You will need to 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**.

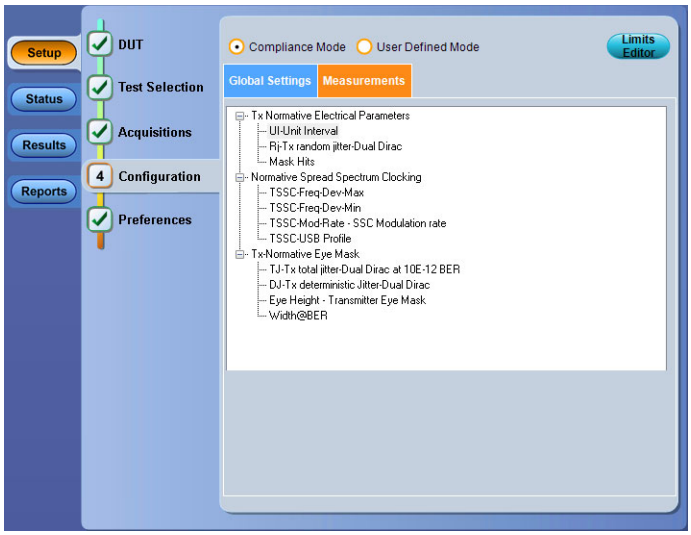
# Configuration tab parameters

Use the **Configuration** tab to set and view global instrument parameters for the selected tests. Which fields are available to edit depends 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.*




Configuration tab: Global Settings




Configuration tab: Measurements

### See also

Configuration tab: Global Settings parameters (see page 33)
Configuration tab: Measurements parameters (see page 35)

# Configuration tab: Global Settings parameters

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

### Table 11: Configuration tab Global Settings

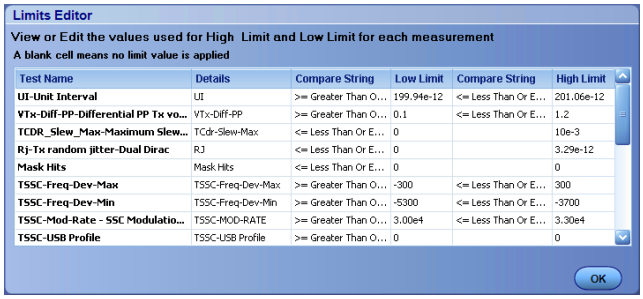
| Control | Description |
|---|---|
| Test Mode | Determines whether test parameters are in compliance or can be edited.<br>■ Compliance: Most test parameter values cannot be edited. Select this for compliance mode testing.<br>■ User Defined: Enables editing of filters for the link. |
| **Limits Editor** button | Opens the Limits Editor dialog box.<br>In User Defined Mode, use the Limits Editor to edit individual test limit settings.<br>  <br>To edit a value, click that field and either select from the displayed list or enter a new value. Use scroll bars to view all available fields.<br>Limits Editor: compare string definitions (see page 111)<br>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. |
| Instruments Detected | Displays a list of the connected instruments found during the instrument discovery. Instrument types include equipment such as oscilloscopes and signal sources (AFG, AWG). Select **Options > Instrument Control Settings** to refresh the connected instrument list (see page 20). |
| CM measurement using TriMode™ Probe with DPOJET Test Method | Set this when using a supported Tektronix TriMode probe for signal acquisition of the LFPS Vcm-AC measurement. The TriMode Probe can switch between differential, single ended and common mode (CM) measurements without changing the probe. In CM mode, it generates CM signal by taking two Single Ended inputs (D+ and D-).<br>The application applies post-processing on the entire LFPS CM acquisition, after applying compliance filters.<br>This control is only valid for the DPOJET test method. |
| CP0 CP1 CP7 Record Length | Sets the waveform record length. |
| Signal Validation | Sets the signal validation actions. Select from the available list items. |
| Probing Configuration | Sets the signal probe configuration (single-ended or differential) for all available channel selections (Ch1 – Ch 4) |
| LFPS Acquisition Bandwidth (GHz) | Sets the oscilloscope bandwidth for Low Frequency Periodic Signal (LFPS) acquisition. Valid range is 5 GHz to 12 GHz in 1 GHz steps. |

**Table 11: Configuration tab Global Settings (cont.)**

| Control | Description |
|---|---|
| LFPS fields | Sets LFPS waveform parameters. If the oscilloscope does not properly trigger on the DUT LFPS signal, adjust these trigger settings to enable the oscilloscope to detect and trigger on the LFPS signal. |
| AFG fields | Sets AFG parameters. If the oscilloscope does not properly trigger on the DUT LFPS signal, adjust these AFG settings to enable the oscilloscope to detect and trigger on the LFPS signal.<br>This is applicable if AFG is selected as the Toggle tool. |

***NOTE.*** *If AWG is selected as the Toggle tool, the application enables the* ***Verify Toggle Status*** *check box. Use this function to identify if the DUT toggled twice instead of one time.*

***NOTE.*** *If Do not use is selected as the Toggle tool, the application enables the* ***Auto Recovery Settings*** *check box. Use this function to automatically recover the oscilloscope settings if modified by the user during test execution.*

### See also

About acquisitions (see page 30)
Limits Editor: compare string definitions (see page 111)

## Configuration tab: Measurements

Lists all selected tests. There are no measurement parameter or configuration setting controls in the Measurements tab.

### See also

Set acquisition parameters (see page 30)
Limits Editor: compare string definitions (see page 111)

# Preferences tab

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

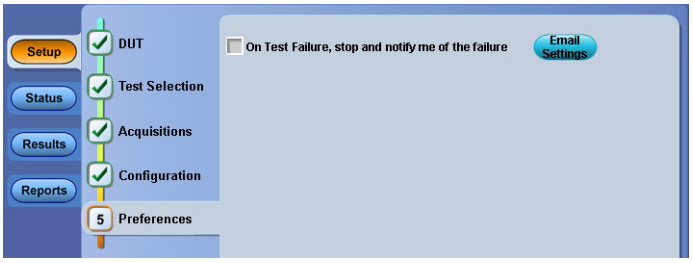


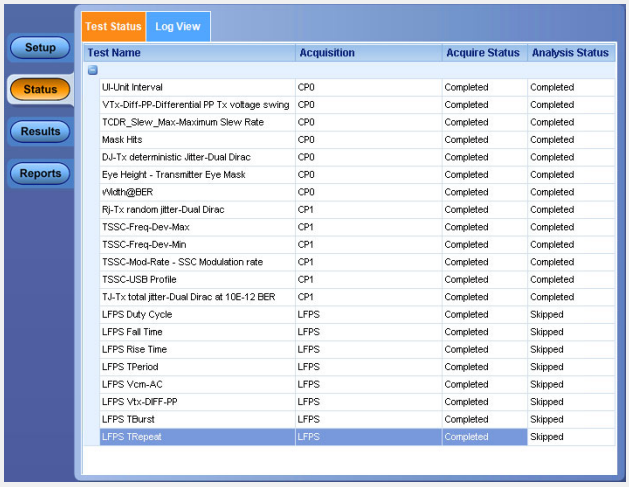

**Table 12: Preferences tab settings**

| Setting | Description |
|---|---|
| On Test Failure, stop and notify me of the failure | Stops the test and sends an email when a test fails.<br>Click Email Settings to verify that Email Test Results when complete or on error is selected, and to verify the address to which the email is sent. |

# Status panel overview

The Status button accesses the Test Status and Log View tabs, which provide status on test acquisition and analysis (Test Status (see page 37) tab) and a listing of test tasks performed (Log View (see page 37) 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

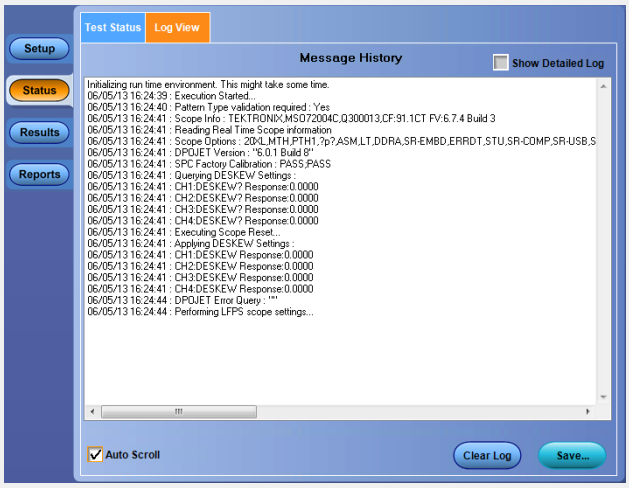


**Table 13: Status panel settings**

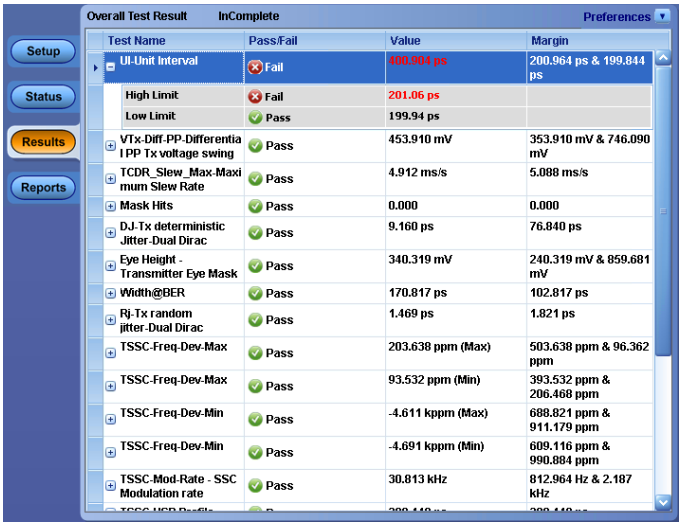
| Control | Description |
|---|---|
| Message History | Window that lists all executed test operations and timestamp information. |
| Show Detailed Log | Enables recording a more-detailed history of test execution.<br>***NOTE.*** *This must be selected before starting a measurement.* |
| Auto Scroll | 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 (see page 26)

# Results panel overview

When a test finishes, the application automatically opens the **Results** panel to display a summary of signal and preset 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 (see page 39) in the upper right corner. Viewing preferences include showing whether a test passed or failed, summary or detailed results, and enabling wordwrap.

***NOTE.*** *NAN (Not A Number) is displayed in the test results if an invalid waveform was supplied for the test.*

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 (see page 39) to change how some items display in the Results panel.

### See also

View a report (see page 43)
Application panels overview (see page 26)

## 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 show or hide the Pass/Fail column, select **Preferences > Show Pass/Fail**.
- To collapse all expanded tests, select **Preferences > View Results Summary**.
- To expand all tests listed, select **Preferences > View Results Details**.
- To enable or disable the wordwrap feature, select **Preferences > Enable Wordwrap**.

### See also

Results panel overview (see page 38)

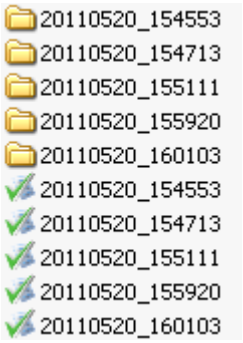
# View test-related files

Files related to tests are stored in the `My TekExpress\USB` folder. Each test setup in this folder has a test setup file and a test setup folder, both with the test setup name.

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

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

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




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

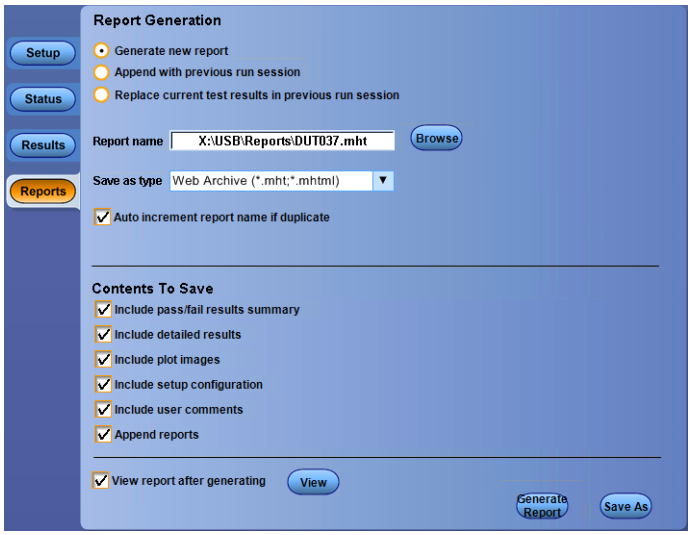
The first time you run a new, unsaved session, the session files are stored in the `Untitled Session` folder located at `..\My TekExpress\USB`. When you name and save the session, the files are placed in a folder with the name that you specify. A copy of the test files stay in the Untitled Session folder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

### See also

File name extensions (see page 13)
Required \My TekExpress folder settings (see page 9)

# Reports panel overview

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




For information on setting up reports, see Select report options (see page 41). For information on viewing reports, see View a Report (see page 43).

### See also

About panels (see page 26)

# Select report options

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

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

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

### Table 14: Report options

| Setting | Description |
|---|---|
| **Report Generation** | |
| Generate new report | Creates a new report. |
| Append with previous run session | Appends the latest test results to the end of the current test results report. |
| Replace current test in previous run session | Replaces the previous test results with the latest test results. Results from newly added tests are appended to the end of the report. |

**Table 14: Report options (cont.)**

| Setting | Description |
|---|---|
| Report name | Displays the name and location from which to open a report. The default location is at *\My TekExpress\USB\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.<br>Change the report name or location.<br>Do one of the following:<br>■ In the Report Path field, type over the current folder path and name.<br>■ Double-click in the Report Path field and then make selections from the popup keyboard and click the **Enter** button.<br>Be sure to include the entire folder path, the file name, and the file extension. For example: C:\Documents and Settings\your user name\My Documents\My TekExpress\USB\DUT001_Test_72.7.1.3.mht.<br>***NOTE.*** *You cannot set the file location using the Browse button.*<br>Open an existing report.<br>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 specified file type. Lists supported file types to choose from.<br>***NOTE.*** *If you select a file type different from the default, be sure to change the report file name extension in the Report Name field to match.* |
| 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 results summary | Sets the application to include the color block labeled Test Result (indicating whether the test passed or failed) in the report. For details, see Report Contents in View a report (see page 43). |
| Include detailed results | Sets the application to include parameter limits, execution time, and test-specific comments generated during the test. |
| Include plot images | Sets the application to include plotted diagrams such as Eye diagrams. |
| Include setup configuration | Sets the application to include hardware and software information in the summary box at the top of the report. Information includes: the oscilloscope model and serial number, probe model and serial number, the oscilloscope firmware version, SPC and factory calibration status, and software versions for applications used in the measurements. |
| Include user comments | Select to include any comments about the test that you or another user added in the DUT tab of the Setup panel. Comments appear in the Comments section, under the summary box at the beginning of each report. |
| Append reports | Append with previous run session |

**Table 14: Report options (cont.)**

| Setting | Description |
|---|---|
| View Report After Generating | Automatically opens the report in a Web browser when the test completes. This option is selected by default. |
| View | Click to view the most current report. |
| Generate Report | Generates a new report based on the current 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 (see page 41).

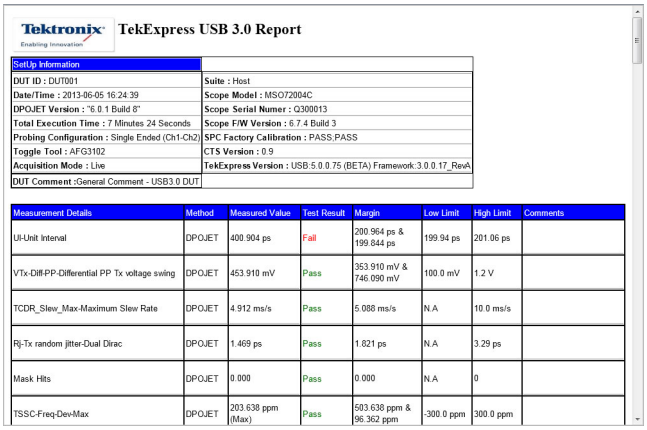
# Report contents

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

***NOTE.*** *NAN (Not A Number) is displayed in the report contents if an invalid waveform was supplied for the test.*

Setup configuration information

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



Tektronix Enabling Innovation

**TekExpress USB 3.0 Report**

| SetUp Information | |
|---|---|
| **DUT ID :** DUT001 | **Suite :** Host |
| **Date/Time :** 2013-06-05 16:24:39 | **Scope Model :** MSO72004C |
| **DPOJET Version :** "6.0.1 Build 8" | **Scope Serial Numer :** Q300013 |
| **Total Execution Time :** 7 Minutes 24 Seconds | **Scope F/W Version :** 6.7.4 Build 3 |
| **Probing Configuration :** Single Ended (Ch1-Ch2) | **SPC Factory Calibration :** PASS;PASS |
| **Toggle Tool :** AFG3102 | **CTS Version :** 0.9 |
| **Acquisition Mode :** Live | **TekExpress Version :** USB:5.0.0.75 (BETA) Framework:3.0.0.17_RevA |
| **DUT Comment :**General Comment - USB3.0 DUT | |

| Measurement Details | Method | Measured Value | Test Result | Margin | Low Limit | High Limit | Comments |
|---|---|---|---|---|---|---|---|
| UI-Unit Interval | DPOJET | 400.904 ps | Fail | 200.964 ps & 199.844 ps | 199.94 ps | 201.06 ps | |
| VTx-Diff-PP-Differential PP Tx voltage swing | DPOJET | 453.910 mV | Pass | 353.910 mV & 746.090 mV | 100.0 mV | 1.2 V | |
| TCDR_Slew_Max-Maximum Slew Rate | DPOJET | 4.912 ms/s | Pass | 5.088 ms/s | N.A | 10.0 ms/s | |
| Rj-Tx random jitter-Dual Dirac | DPOJET | 1.469 ps | Pass | 1.821 ps | N.A | 3.29 ps | |
| Mask Hits | DPOJET | 0.000 | Pass | 0.000 | N.A | 0 | |
| TSSC-Freq-Dev-Max | DPOJET | 203.638 ppm (Max) | Pass | 503.638 ppm & 96.362 ppm | -300.0 ppm | 300.0 ppm | |

User comments

If you selected to 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 (see page 38)
View test-related files (see page 40)

# Test process flow

Use the following list to set up and performing USB-TX tests.

1. Allow test instruments to warm up (~20 minutes).
2. Deskew the real-time oscilloscope (see page 45).
3. Set up test equipment (see page 47).
4. Verify that required instruments are connected to USB-TX (see page 21).
5. Set DUT parameters (see page 27).
6. Select tests (see page 28).
7. View acquisition settings (see page 30).
8. Set global signal-related parameters (see page 33).
9. Select test notification preferences (see page 36).
10. Select report options (see page 41).
11. Check the prerun checklist (see page 48)
12. Click **Start** to Run tests (see page 47).

### See also

About test setups (see page 49)
About running tests (see page 47)

# Deskew real-time oscilloscopes

Use the following procedure to deskew direct input SMA channels on a real time oscilloscope.

***NOTE.*** *DPOJET has an automatic deskew option under* ***Analyze > Jitter And Eye Analysis > Deskew****. Refer to your DPOJET online help for information on how to deskew the channels.*

1. Run Signal Path Compensation (SPC) on the oscilloscope.
2. Connect a SMA Power Splitter (preferred) or SMA 50 Ω coaxial "T" connector to the Fast Edge output of the oscilloscope.
3. Connect SMA cables from each of the two channels to be deskewed to the power splitter (or SMA coaxial "T" connector). It is best to use matched cables when making high speed serial measurements. **It is important to use the same cables that will be used for subsequent measurements**.

4. Select **Default Setup**, and then select **Autoset** on the oscilloscope front panel.
5. Set the oscilloscope for 70% to 90% full screen amplitude on both channels. Center both traces so that they overlap.
6. Make sure that volts/div, position, and offset are identical for the two channels being deskewed.
7. Set the time/div to approximately 100 ps/div or less, with sample rate at 1 ps/pt. These settings are not critical, but should be close.
8. Set the horizontal acquisition mode to average, which provides a more stable display.
9. Select **Deskew** from the **Vertical** menu.
10. Verify that the reference channel (typically CH1 or CH2) is set to 0 ps deskew.
11. In the deskew control window, select the channel to deskew (typically CH3 or CH4). Adjust the deskew to overlay the rising edge as best as possible.

***NOTE.*** *Typical values are in the 10's of ps or less with cables connected directly from Fast Edge to SMA inputs. If you are using a switch box (for example, Keithley), deskew the complete path from where the test fixture connects, through the switch, and into the oscilloscope. Deskew values in these cases may be as much as 30 ps or more.*

***NOTE.*** *There can be significant differences in the skew between two TCA-SMA adapters. If you find that a system requires a very large correction, obtain a pair of TCA-SMA adapters that closely match each other to reduce the amount of correction.*

***NOTE.*** *TekExpress retains the user configured Deskew values, and does not override the values during test runs.*

# 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 (see page 3)
View connected instruments (see page 21)

# Running tests

After selecting and configuring tests, review the prerun checklist (see page 48) 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 USB application on top, select **Keep On Top** from the TekExpress Options menu.

### See also

Configuration tab parameters (see page 32)

# Prerun checklist

Do the following before you click Start to run a test:

***NOTE.*** *If this is the first time you are running a test on the application, make sure that you have done the steps in Required \My TekExpress folder settings (see page 9) before continuing.*

1. Make sure that all the required instruments are properly warmed up (approximately 20 minutes).
2. Perform Signal Path Compensation (SPC)
   a. On the oscilloscope main menu, select the **Utilities** menu.
   b. Select **Instrument Calibration**.
   c. Follow the on-screen instructions.
3. Verify that the correct instruments are connected (oscilloscope and signal sources):
   a. In USB-TX, click **Setup > Configuration**.
   b. Click **Global Settings**.
   c. In the **Instruments 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 page 21)).

## See also

Instrument and DUT connection setup (see page 47)

# 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:\USB**.

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 (see page 49)
Recall a saved test setup (see page 50)

# 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, and so on).
3. Click the application **Reports** button and set your report options (see page 41).
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:\USB\<*session_name*.

### See also

Test process flow (see page 45)
View test-related files (see page 40)
Configuration tab parameters (see page 32)

## 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:\USB**.

### See also

About test setups (see page 49)
Create a new test setup based on an existing one (see page 50)
Test setups overview (see page 45)

## 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.

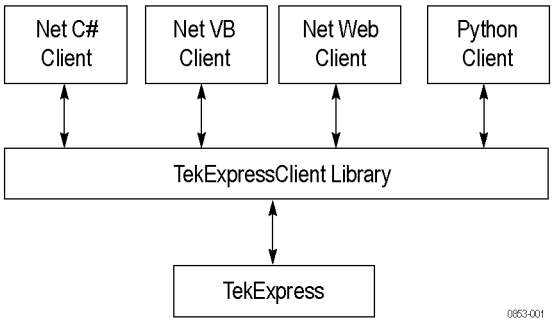
1. Select **Options > Open Test Setup**.
2. Select a setup from the list and then click **Open**.
3. Use the **Setup** and **Reports** panels to modify the parameters to meet your testing requirements.
4. Select **Options > Save Test Setup As**.
5. Enter a test setup name and click **Save**.

### See also

About test setups (see page 49)
Set DUT parameters (see page 27)
Configuration parameters (see page 35)
Select acquisitions (see page 30)

# About the programmatic interface

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




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

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

TekExpress leverages .Net Marshalling to enable the Programmatic Interface for TekExpress Client. TekExpress provides a client library for TekExpress clients to use the programmatic interface. The TekExpress client library is inherited from .Net MarshalByRef class to provide the proxy object for the clients. The TekExpress client library maintains a reference to the TekExpress Server and this reference allows the client to control the server state.

### See also

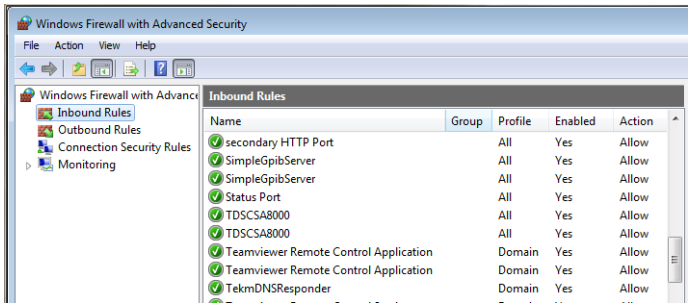
Requirements for Developing TekExpress Client (see page 54)

# To enable remote access

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

1. Access the Windows Control Panel and open the Windows Firewall tool (**Start > Control Panel > All Control Panel Items > Windows Firewall**).
2. Click **Advance Settings > Inbound Rules**.

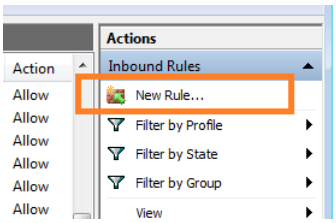
3. Scroll through the **Inbound Rules** list to see if the following items (or with a similar name) are shown:
   - TekExpress USB
   - TekExpress




4. If both items are shown, you do not need to set up any rules. Exit the Windows Firewall tool.
5. If one or both are missing, use the following procedure to run the **New Inbound Rule Wizard** and add these executables to the rules to enable remote access to the TekExpress application.

## Run the New Inbound Rule Wizard

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

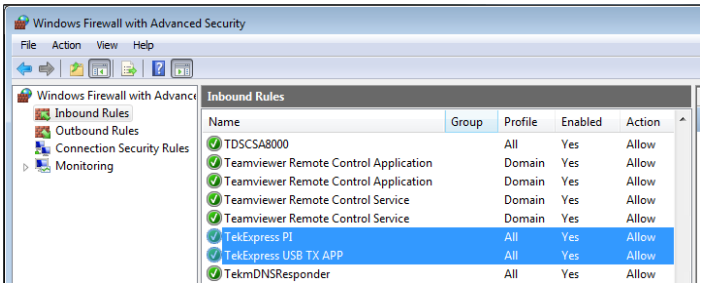



2. Verify that **Program** is selected in the Rule Type panel and click **Next**.
3. Click **Browse** in the Program panel and navigate to and select one of the following TekExpress applications (depending on the one for which you need to create a rule):
4. TekExpress USB.exe
5. TekExpress.exe

***NOTE.*** *See Application directories and content (see page 12) for the path to the application files.*

6. Click **Next**.
7. Verify that **Allow the connection** is selected in the Action panel and click **Next**.
8. Verify that all fields are selected (**Domain**, **Private**, and **Public**) in the Profile panel and click **Next**.
9. Use the fields in the Name panel to enter a name and optional description for the rule. For example, a name for the TekExpress USB application could be **TekExpress USB RX Application**. Add description text to further identify the rule.

10. Click **Finish** to return to the main Windows Firewall screen.
11. Scroll through the Inbound Rules list and verify that the list shows the rule that you just entered.




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

### To use the remote access:

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

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

# Requirements for developing TekExpress client

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

## References required

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

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

## Required steps for a client

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

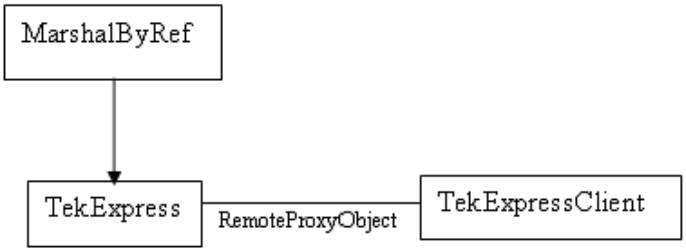
Develop a client UI to access the interfaces exposed through the server. This client loads `TekExpressClient.dll` to access the interfaces. After `TekExpressClient.dll` is loaded, the client UI can call the specific functions to run the operations requested by the client. When the client is up and running,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

1. To connect to the server, the client provides the IP address of the PC where the server is running.
2. The client locks the server application to avoid conflict with any other Client that may try to control the server simultaneously. “Lock” would also disable all user controls on the server so that server state cannot be changed by manual operation.

   If any other client tries to access a server that is locked, it will receive a notification that the server is locked by another client.
3. When the client has connected to and locked the server, the client can access any of the programmatic controls needed to run the remote automations.
4. After the client operations finish, the client unlocks the server.

# 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 (TekExpressRe-
moteInterface), “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.

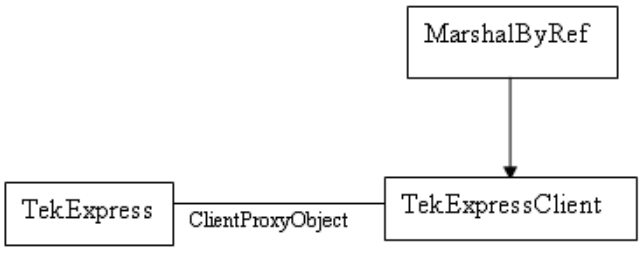
For example,

//Get a reference to the remote object

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

# Client proxy object

Client exposes a proxy object to receive certain information.



For example,

//Register the client proxy object

```
WellKnownServiceTypeEntry[] e = RemotingConfiguration.GetRegisteredWell-
KnownServiceTypes();

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.

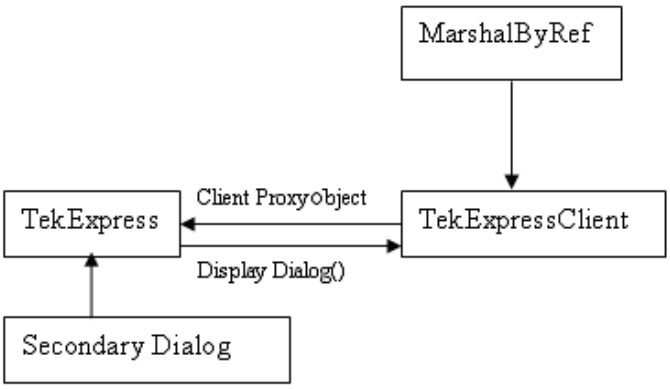
Examples

```
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.

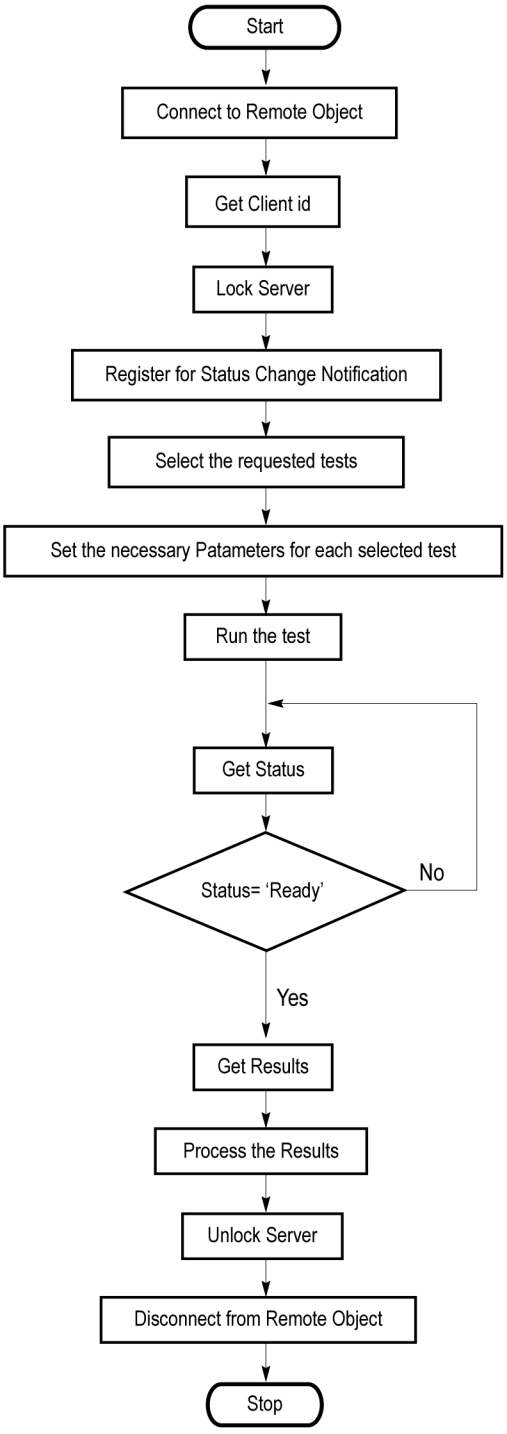
File Transfer Events

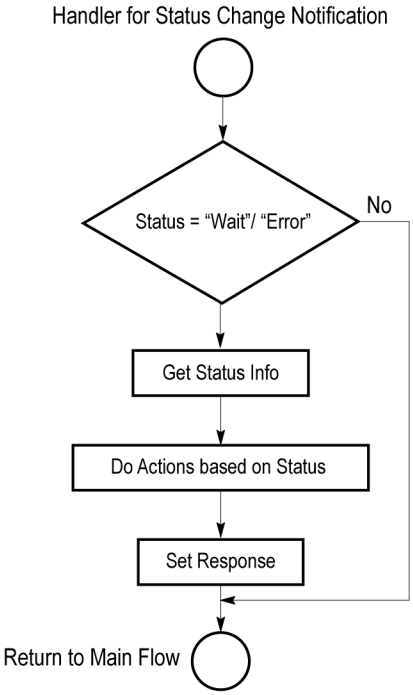
When the client requests the transfer of the report, the server reads the report and transfers the file by calling the file transfer methods at the client-end.

## Client programmatic interface example

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

Process flowchart


Start
Connect to Remote Object
Get Client id
Lock Server
Register for Status Change Notification
Select the requested tests
Set the necessary Patameters for each selected test
Run the test
Get Status
Status= 'Ready'
No
Yes
Get Results
Process the Results
Unlock Server
Disconnect from Remote Object
Stop


Handler for Status Change Notification
Status = "Wait"/ "Error"
No
Get Status Info
Do Actions based on Status
Set Response
Return to Main Flow

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 (see page 60)

# Program remote access code example

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

**Table 15: Remote access code example**

| Task | Code |
|---|---|
| Start the application | |
| Connect through an IP address. | `m_Client.Connect("localhost") 'True or False`<br>`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`<br>`Thread.Sleep(500)`<br>`m_Client.Application_Status(clientID)`<br>`Select Case status`<br>`Case "Wait"` |
| Get the current state information | `mClient.GetCurrentStateInfo(clientID, WaitingMsbBxCaption, WaitingMsbBxMessage, WaitingMsbBxButtontexts)` |
| Send the response | `mClient.SendResponse(clientID, WaitingMsbBxCaption, WaitingMsb-BxMessage, WaitingMsbBxResponse)`<br>`End Select`<br>`Loop Until status = "Ready"` |
| Save results | `'Save all results values from folder for current run`<br>`m_Client.TransferResult(clientID, logDirname)` |
| Unlock the server | `m_Client.UnlockServer(clientID)` |
| Disconnect from server | `m_Client.Disconnect()` |
| Exit the application | |

# ApplicationStatus()

### 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.<br>clientId example (see page 62) |

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

*The server is LOCKED and the message displayed is "Server is locked by another client".*

*The session is UNLOCKED and the message displayed is "Lock Session to execute the command".*

*The server is NOTFOUND and the message displayed is "Server not found...Disconnect!".*

*When none of these fail conditions occur, then the message displayed is "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)

GetCurrentStateInfo (see page 68)
QueryStatus (see page 76)
SendResponse (see page 81)

Status (see page 103)

**in string clientId example**

clientId = <client_id_number>-<client_IP_address>.

For example, 1065–192.157.98.70

# ChangeDutId()

## ChangeDutId(clientId, dutName)

This command changes the DUT id of the set-up. The client has to provide a valid DUT id.

### Parameters

| Parameter | Type | Direction | Description |
|---|---|---|---|
| clientId | String | IN | Identifier of the client that is performing the remote function.<br>clientId example (see page 62) |
| dutName | String | IN | The new DUT id of the set-up. |

### Return value

String that indicates the status of the operation upon completion.

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

*The server is LOCKED and the message displayed is "Server is locked by another client".*

*The session is UNLOCKED and the message displayed is "Lock Session to execute the command".*

*The server is NOTFOUND and the message displayed is "Server not found...Disconnect!".*

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

### Example

```
If (dut Id.Length <=0 && locked == true)
   return “Enter a valid DUT-ID”;
returnVal = remoteObject.ChangeDutId(clientId, dutId);
if ((OP_STATUS)returnVal == OP_STATUS.SUCCESS)
   return “DUT Id Changed...”;
```

```
else
   return CommandFailed(returnVal);
```

### Comments

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

### Related command(s)

GetDutId (see page 70)

# CheckSessionSaved()

### CheckSessionSaved(clientID, out 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.<br>clientId example (see page 62) |
| 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)

### Comments

### Related command(s)

CheckSessionSaved
RecallSession
SaveSession
SaveSessionAs

# Connect()

### Connect(hostIPAddress, clientInterface, out 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. Any number of clients can be connected 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.<br>clientId example (see page 62) |

### 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:*

*The server is LOCKED and the message displayed is "Server is locked by another client".*

*The session is UNLOCKED and the message displayed is "Lock Session to execute the command".*

*The server is NOTFOUND and the message displayed is "Server not found...Disconnect!".*

*When none of these fail conditions occur, then the message displayed is "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;
}
```

### Comments

The server has to be active and running for the client to connect to the server. Any number of clients can be connected to the server at a time. Each client will get a unique id.

### Related command(s)

Disconnect (see page 64)

# 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.<br>clientId example (see page 62) |

### 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 (see page 64)

# GetCurrentStateInfo()

### GetCurrentStateInfo(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.<br>clientId example (see page 62) |
| 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:*

*The server is LOCKED and the message displayed is "Server is locked by another client".*

*The session is UNLOCKED and the message displayed is "Lock Session to execute the command".*

*The server is NOTFOUND and the message displayed is "Server not found...Disconnect!".*

*When none of these fail conditions occur, then the message displayed is "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)

### Comments

### Related command(s)

ApplicationStatus (see page 61)
QueryStatus (see page 76)
SendResponse (see page 81)

# GetDutId()

### GetDutId(clientId, out dutId)

This command returns the DUT id of the current set-up.

### Parameters

| Parameter | Type | Direction | Description |
|---|---|---|---|
| clientId | string | IN | Identifier of the client that is performing the remote function.<br>clientId example (see page 62) |
| dutId | string | OUT | The DUT id of the set-up. |

### Return value

String that gives the timeout period (in seconds) of the client.

### Example

```
returnVal = remoteObject.GetDutId(clientId, out dutId);
if ((OP_STATUS)returnVal == OP_STATUS.SUCCESS)
{
   return id;
}
else
   return CommandFailed(returnVal);
```

### Comments

The dutId is an OUT parameter whose value is set after the server processes the request.

### Related command(s)

ChangeDutId (see page 62)
SetDutId (see page 85)

# GetPassFailStatus()

### 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<br>clientId example (see page 62) |
| 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 pass or fail status. |

### Return value

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

### Example

GetPassFailStatus(clientId, “Device”, “Device Connector”, test);

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

# GetReportParameter()

### GetReportParameter(clientId, device, suite, test, parameterString)

This command gets the general report details such as oscilloscope model and TekExpress version.

### Parameters

| Parameter | Type | Direction | Description |
|---|---|---|---|
| clientId | string | IN | Identifier of the client that is connected to the server<br>clientId example (see page 62) |
| 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 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:*

*The server is LOCKED and the message displayed is "Server is locked by another client".*

*The session is UNLOCKED and the message displayed is "Lock Session to execute the command".*

*The server is NOTFOUND and the message displayed is "Server not found...Disconnect!".*

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

### Return value

The return value is the connected oscilloscope model, TekExpress base software version, or USB-TX application version.

### Example

GetReportParameter(clientId, "Device", "Device Connector", test, "Application Version")

# GetResultsValue()

## 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<br>clientId example (see page 62) |
| 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 |

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

*The server is LOCKED and the message displayed is "Server is locked by another client".*

*The session is UNLOCKED and the message displayed is "Lock Session to execute the command".*

*The server is NOTFOUND and the message displayed is "Server not found...Disconnect!".*

*When none of these fail conditions occur, then the message displayed is "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, "Device", "Device Connector", test, "Value");

# GetTimeOut()

## GetTimeOut(clientId)

Returns the current timeout period set by the client.

### Parameters

| Parameter | Type | Direction | Description |
|---|---|---|---|
| clientId | string | IN | Identifier of the client that is connected to the server<br>clientId example (see page 62) |

### Return value

String value that indicates the status of the operation upon completion. The default return value is 1800000. Returnval as string.

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

*The server is LOCKED and the message displayed is "Server is locked by another client".*

*The session is UNLOCKED and the message displayed is "Lock Session to execute the command".*

*The server is NOTFOUND and the message displayed is "Server not found...Disconnect!".*

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

### Example

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

returnval as string

returnval=m_Client.GetTimeOut()

### Comments

### Related command(s)

SetTimeOut (see page 101)

# LockSession()

### 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.<br>clientId example (see page 62) |

### 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)

UnlockSession (see page 107)

# QueryStatus()

## QueryStatus(clientID, out status)

This command transfers Analyze panel status messages from the server to the client.

### Parameters

| Parameter | Type | Direction | Description |
|---|---|---|---|
| clientID | string | IN | Identifier of the client that is connected to the server<br>clientId example (see page 62) |
| 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:*

*The server is LOCKED and the message displayed is "Server is locked by another client".*

*The session is UNLOCKED and the message displayed is "Lock Session to execute the command".*

*The server is NOTFOUND and the message displayed is "Server not found...Disconnect!".*

*When none of these fail conditions occur, then the message displayed is "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)

ApplicationStatus (see page 61)
GetCurrentStateInfo (see page 68)

SendResponse (see page 81)

# RecallSession()

### RecallSession(clientId,sessionName)

Recalls a saved session. The name of the session is provided by the client.

### Parameters

| Parameter | Type | Direction | Description |
|---|---|---|---|
| clientId | string | IN | Identifier of the client that is performing the remote function.<br>clientId example (see page 62) |
| sessionName | string | IN | The name of the session being recalled. |

### Return value

String that indicates the status of the operation upon completion.

### Example

```
returnVal = remoteObject.RecallSession(clientId,sessionName);
if ((OP_STATUS)returnVal == OP_STATUS.SUCCESS)
   return “Session Recalled...”;
else
   return CommandFailed(returnVal);
```

### Comments

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

### Related command(s)

SaveSession (see page 79)
SaveSessionAs (see page 80)

# Run()

## Run(clientId)

Runs the setup. Once the server is set up and configured, it can be run remotely using this function.

### Parameters

| Parameter | Type | Direction | Description |
|---|---|---|---|
| clientId | string | IN | Identifier of the client that is connected to the server<br>clientId example (see page 62) |

### Return value

String that returns the status of the operation after completion.

### Example

```
returnVal = remoteObject.Run(clientId);
if ((OP_STATUS)returnVal == OP_STATUS.SUCCESS)
   return “Run started...”;
else
   return CommandFailed(returnVal);
```

### Comments

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

# SaveSession()

### SaveSession(clientId,sessionName)

Saves the current session. 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<br>clientId example (see page 62) |
| 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.SaveSession(clientId,sessionName);
if ((OP_STATUS)returnVal == OP_STATUS.SUCCESS)
   return “Session Saved...”;
else
   return CommandFailed(returnVal);
```

### Comments

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

Once the session is saved under ‘name,’ you cannot use this command to save the session with a different name. Use SaveSessionAs to save the session to a new name.

### Related command(s)

RecallSession (see page 77)
SaveSessionAs (see page 80)

# SaveSessionAs()

### 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<br>clientId example (see page 62) |
| 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)

RecallSession (see page 77)
SaveSession (see page 79)

# SendResponse()

## 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 GetCurrentStateInfo 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<br>clientId example (see page 62) |
| 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:*

*The server is LOCKED and the message displayed is "Server is locked by another client".*

*The session is UNLOCKED and the message displayed is "Lock Session to execute the command".*

*The server is NOTFOUND and the message displayed is "Server not found...Disconnect!".*

*When none of these fail conditions occur, then the message displayed is "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)

ApplicationStatus (see page 61)
GetCurrentStateInfo (see page 68)
QueryStatus (see page 76)

# SelectDevice()

### SelectDevice(clientId, device, true)

This command selects the DUT type (Host or Device).

### Parameters

| Parameter | Type | Direction | Description |
|---|---|---|---|
| clientId | string | IN | Identifier of the client that is connected to the server<br>clientId example (see page 62) |
| device | string | IN | String with the device DUT type. Valid values are **Host** and **Device**. |

### Return value

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

### Example

SelectDevice(clientId, "Device", true);

SelectDevice(clientId, "Host", true);

# SelectSuite()

### SelectSuite(clientId, device, deviceConnector, true)

This command selects one of the two suites: "Device Connector" or "Host Connector."

### Parameters

| Parameter | Type | Direction | Description |
|---|---|---|---|
| clientId | string | IN | Identifier of the client that is connected to the server<br>clientId example (see page 62) |
| device | string | IN | String with the device DUT type. Valid values are **Host** and **Device**. |
| deviceConnector | string | IN | string with device connection type. Valid values are **Host Connector** and **Device 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);

# SelectTest()

## SelectTest(clientId, device, deviceConnector, test, true)

This command selects a test.

### Parameters

| Parameter | Type | Direction | Description |
|---|---|---|---|
| clientId | string | IN | Identifier of the client that is connected to the server<br>clientId example (see page 62) |
| device | string | IN | String with the device DUT type. Valid values are **Host** and **Device**. |
| deviceConnector | string | IN | string with device connection type. Valid values are **Host Connector** and **Device Connector** |
| test | string | IN | Name of the USB-TX test. |

### Return value

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

### Example

SelectTest(clientId, device, deviceConnector, "UI-Unit Interval", true);

# SetDutId()

### SetDutId(clientID,newDutId)

This command changes the DUT ID of the setup. The client must provide a valid DUT ID.

### Parameters

| Parameter | Type | Direction | Description |
|---|---|---|---|
| clientId | string | IN | Identifier of the client that is connected to the server<br>clientId example (see page 62) |
| newDutId | string | IN | The new DUT ID of the setup. |

### Return value

String that gives the status of the operation after it was performed.

Return value is “DUT Id Changed” on success.

### Example

```
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)
```

### Comments

### Related command(s)

GetDutId (see page 70)

# SetGeneralParameter()

### SetGeneralParameter(clientId, device, deviceConnection, "", paramString)

This command sets the general parameter and its value based on the "paramString" argument values as listed.

### Parameters

| Parameter | Type | Direction | Description |
|---|---|---|---|
| clientId | string | IN | Identifier of the client that is connected to the server<br>clientId example (see page 62) |
| 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 pass or fail status or a test result value. Enter a null value for this field (""). |
| parameterString | string | IN | Specifies the control to set. See the following links for argument values and examples for this field. |

### Return value

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

### paramString argument values

Use the following links to see the paramString values associated with specific application settings.

Select CTLE filter file (see page 92)
Select de-embed filter file (see page 89)
Select embed filter file (see page 91)
Select test method (see page 87)
Select test point (see page 87)
Set AFG frequency (see page 99)
Set AFG number of cycles (see page 98)
Set AFG voltage level high (see page 99)
Set AFG voltage level low (see page 100)
Set Auto Recovery mode (see page 94)
Set bandwidth for LFPS acquisition (see page 93)
Set CM Measurement TriMode Probe mode (see page 94)
Set CTLE filter mode (see page 92)
Set de-embed filter mode (see page 88)
Set embed filter mode (see page 90)
Set LFPS hysteresis level (see page 98)
Set LFPS mid edge reference level (see page 97)
Set LFPS trigger level (see page 97)
Set LFPS trigger lower limit (see page 96)
Set LFPS trigger upper limit (see page 96)
Set probing configuration (see page 93)

Set record length (see page 95)
Set signal pattern validation mode (see page 100)
Set SSC mode (see page 88)
Set verify toggle mode (see page 95)

# Select test point

Use this paramString value to set the DUT test point used by the application. This is the same as selecting the Test Point control on the DUT tab.

The value in bold font is the default value.

### Values:

- **Version$Compliance (TP1) - Far End**
- Version$Tx Pins - Near End
- Version$Custom

### Example

SetGeneralParameter(clientId, Device, Device connector, "", "Version$Compliance (TP1) – Far End");

SetGeneralParameter(clientId, Host, Host Connector, "", "Version$Tx Pins – Near End");

# Select test method

Use this paramString value to set the test method used by the application. This is the same as using the **Select Test Method** controls on the **DUT** tab.

The value in bold font is the default value.

### Values:

- Test Tool$USB-IF Software (SigTest)
- **Test Tool$DPOJET**
- Test Tool$Both

### Example

SetGeneralParameter(clientId, Device, Device Connector, "", "Test Tool$USB-IF Software (SigTest)")

SetGeneralParameter(clientId, Device, Device Connector, "", "Test Tool$Both");

# Set SSC mode

Use this paramString value to set the enable or disable the Spread Spectrum Clocking (SSC) mode used by the application for supported DUTs. This is the same as selecting the **Spread Spectrum Clocking** control on the **DUT** tab.

The value in bold font is the default value.

### Values:

- **SSC On$true**
- SSC On$false

### Example

SetGeneralParameter(clientId, Device, Device Connector, "", "SSC On$true");

SetGeneralParameter(clientId, Device, Device Connector, "", "SSC On$false");

# Set de-embed filter mode

Use this paramString value to enable or disable de-embedding filter files. This is the same as selecting the **De-Embed** check box on the DUT tab.

The value in bold font is the default value.

### Values:

- **Compliance (TP1) - Far End - Deembed Filter Option$[true** | false]
- Tx Pins - Near End - Deembed Filter Option$[true | false]
- Custom - Deembed Filter Option$[true | false]

### Example

SetGeneralParameter(clientId, device, deviceConnector, "", "Compliance (TP1) - Far End - Deembed Filter Option$true”);

SetGeneralParameter(clientId, device, deviceConnector, "", "Custom - Deembed Filter Option$false”);

# Select de-embed filter file

Use this paramString value to select the de-embed file to use by the application. This is the same as selecting a file from the **De-Embed** control on the **DUT** tab.

Custom filter files must be in the same directory as the application-provided filter files.

The value in bold font is the default value.

### Values:

- **Compliance (TP1) - Far End - Deembed Filter File Path$[Tx_Device_TF_8G.flt** | custom_file_name.flt]
- Tx Pins - Near End - Deembed Filter File Path$[Tx_Device_TF_8G.flt | custom_file_name.flt]
- Custom - Deembed Filter File Path$[Tx_Device_TF_8G.flt | custom_file_name.flt]

### Example

SetGeneralParameter(clientId, device, deviceConnector, "", “Compliance (TP1) - Far End - Deembed Filter File Path$Tx_Device_TF_8G.flt”);

SetGeneralParameter(clientId, device, deviceConnector, "", “Compliance (TP1) - Far End - Deembed Filter File Path$SpecialTestCase.flt”);

# Set embed filter mode

Use this paramString value to enable or disable filter embedding. This is the same as selecting the **Embed** check box on the DUT tab.

The value in bold font is the default value.

### Values:

- **Compliance (TP1) - Far End - Deembed Filter Option$[true** | false]
- Tx Pins - Near End - Deembed Filter Option$[true | false]
- Custom - Deembed Filter Option$[true | false]

### Example

SetGeneralParameter(clientId, device, deviceConnector, "", "Compliance (TP1) - Far End - Embed Filter Option$true”);

SetGeneralParameter(clientId, device, deviceConnector, "", "Custom - Embed Filter Option$false”);

# Select embed filter file

Use this paramString value to select the filter file to embed for analysis. This is the same as selecting a file from the **Embed** control on the **DUT** tab.

Custom filter files must be in the same directory as the application-provided filter files.

The value in bold font is the default value.

### Values:

- **Compliance (TP1) - Far End - Embed Filter File Path$[Tx_Device_TF_8G.flt** | custom_file_name.flt]
- Tx Pins - Near End - Embed Filter File Path$[Tx_Device_TF_8G.flt | custom_file_name.flt]
- Custom - Embed Filter File Path$[Tx_Device_TF_8G.flt | custom_file_name.flt]

### Example

SetGeneralParameter(clientId, device, deviceConnector, "", “Compliance (TP1) - Far End - Embed Filter File Path$Tx_Device_TF_8G.flt”);

SetGeneralParameter(clientId, device, deviceConnector, "", “Compliance (TP1) - Far End - Embed Filter File Path$SpecialTestCase.flt”);

# Set CTLE filter mode

Use this paramString value to enable or disable CTLE filter embedding. This is the same as selecting the **CTLE** check box on the DUT tab.

The value in bold font is the default value.

### Values:

- **Compliance (TP1) - Far End - CTLE Filter Option$[true** | false]
- Tx Pins - Near End - CTLE Filter Option$[true | false]
- Custom - CTLE Filter Option$[true | false]

### Example

SetGeneralParameter(clientId, device, deviceConnector, "", "Compliance (TP1) - Far End - CTLE Filter Option$true”);

SetGeneralParameter(clientId, device, deviceConnector, "", "Custom - CTLE Filter Option$false”);

# Select CTLE filter file

Use this paramString value to select the CTLE filter file to embed for analysis. This is the same as selecting a file from the **CTLE** control on the **DUT** tab.

Custom filter files must be in the same directory as the application-provided filter files.

The value in bold font is the default value.

### Values:

- **Compliance (TP1) - Far End - CTLE Filter File Path$[USB3CTLE.flt.flt** | custom_file_name.flt]
- Tx Pins - Near End - CTLE Filter File Path$[USB3CTLE.flt | custom_file_name.flt]
- Custom - CTLE Filter File Path$[USB3CTLE.flt | custom_file_name.flt]

### Example

SetGeneralParameter(clientId, device, deviceConnector, "", "Compliance (TP1) - Far End - CTLE Filter File Path$USB3CTLE.flt");

SetGeneralParameter(clientId, device, deviceConnector, "", “Compliance (TP1) - Far End - CTLE Filter File Path$SpecialTestCase.flt”);

# Set probing configuration

Use this paramString value to set the applications probing options. This is the same as using the **Probing Configuration (GHz)** control on the **Configuration** tab.

The value in bold font is the default value.

### Values:

- **Probing Configuration$Single Ended [(Ch1-Ch2)** | (Ch3-Ch4) | (Ch1-Ch3) | (Ch2-Ch4)]
- Probing Configuration$Differential [(Ch1) | (Ch2) | (Ch3) | (Ch4)]

### Example

SetGeneralParameter(clientId, device, deviceConnector, "", "Probing Configuration$Differential (Ch1)");

# Set bandwidth for LFPS acquisition

Use this paramString value to set the bandwidth (in GHz) used to acquire the LFPS signal. This is the same as selecting the XxX control on the DUT tab.

The value in bold font is the default value.

### Values:

- **Bandwidth for LFPS acquisition (GHz)$[5** | 6 | 7 | 8 | 9 | 10 | 11 | 12]

### Example

SetGeneralParameter(clientId, device, deviceConnector, "", "Bandwidth for LFPS acquisition (GHz)$5");

# Set CM Measurement TriMode™ Probe mode

Use this paramString value to enable CM measurement using a Tektronix TriMode™ probe. This is the same as selecting the **CM measurement using a TriMode™ Probe with DPOJET Test Method** control on the **Configuration** tab.

The value in bold font is the default value.

### Values:

- **CM measurement using Trimode probe$[Yes** | No]

### Example

SetGeneralParameter(clientId, device, deviceConnector, "", "CM measurement using Trimode probe$Yes");

SetGeneralParameter(clientId, device, deviceConnector, "", "CM measurement using Trimode probe$No");

# Set Auto Recovery mode

Use this paramString value to enable or disable auto recovery settings mode.

The value in bold font is the default value.

### Values:

- **Auto Recovery Settings$[Yes** | No]

### Example

SetGeneralParameter(clientId, device, deviceConnector, "", "Auto Recovery Settings$Yes");

# Set verify toggle mode

Use this paramString value to set the Verify toggle status. This is the same as selecting the **Verify Toggle Status** control on the **Global Settings** tab of the **Configuration** panel. This is available only when AWG is the toggle tool selected.

The value in bold font is the default value.

### Values:

- **Verify Toggle Status$[Yes** | No]

### Example

SetGeneralParameter(clientId, device, deviceConnector, "", "Verify Toggle Status$Yes");

# Set record length

Use this paramString value to set the acquision record length for CP0 and CP1 signals. This is the same as selecting the **CP0 CP1 CP7 Record Length** control on the **Configuration** tab.

The value in bold font is the default value.

### Values:

- **Record Length for CP0 and CP1$[10000000** | user_entered_value]

### Example

SetGeneralParameter(clientId, device, deviceConnector, "", "Record Length for CP0 and CP1$10000000");

# Set LFPS trigger lower limit

Use this paramString value to set the LFPS signal width trigger lower limit (ns). This is the same as selecting the **Width Trigger Lower limit (ns)** control in the LFPS area of the **Configuration** tab.

The value in bold font is the default value.

### Values:

- **LFPS Width Trigger Lower limit (ns)$[10** | user_entered_value]

### Example

SetGeneralParameter(clientId, device, deviceConnector, "", "LFPS Width Trigger Lower limit (ns)$12");

# Set LFPS trigger upper limit

Use this paramString value to set the LFPS signal width trigger upper limit (ns). This is the same as selecting the **Width Trigger Upper limit (ns)** control in the LFPS area of the **Configuration** tab.

The value in bold font is the default value.

### Values:

- **LFPS Width Trigger Upper limit (ns)$[10** | user_entered_value]

### Example

SetGeneralParameter(clientId, device, deviceConnector, "", "LFPS Width Trigger Upper limit (ns)$10");

# Set LFPS trigger level

Use this paramString value to set the LFPS signal trigger level (mV). This is the same as selecting the **Trigger Level (mV)** control in the LFPS area of the **Configuration** tab.

The value in bold font is the default value.

### Values:

- **LFPS Trigger level (mV)$[140** | user_entered_value]

### Example

SetGeneralParameter(clientId, device, deviceConnector, "", "LFPS Trigger level (mV)$140");

# Set LFPS mid edge reference level

Use this paramString value to set the LFPS signal mid edge reference level (in mV). This is the same as selecting the **Mid Edge Ref Level (mV)** control in the LFPS area of the **Configuration** tab.

The value in bold font is the default value.

### Values:

- **LFPS Mid Edge Ref Level (mV)$[140** | user_entered_value]

### Example

SetGeneralParameter(clientId, device, deviceConnector, "", "LFPS Mid Edge Ref Level (mV)$140");

# Set hysteresis level

Use this paramString value to set the signal hysteresis level (in mV). This is the same as selecting the **Hysteresis Level (mV)** control in the LFPS area of the **Configuration** tab.

The value in bold font is the default value.

### Values:

- **Hysteresis Level (mV)$[50** | user_entered_value]

### Example

SetGeneralParameter(clientId, device, deviceConnector, "", "Hysteresis Level (mV)$50");

# Set AFG number of cycles

Use this paramString value to set the AFG number of cycles per second value. This is the same as selecting the **Num of Cycles** control in the AFG area of the **Configuration** tab.

The value in bold font is the default value.

### Values:

- **AFG Num of Cycles per Second$[2** | user_entered_value]

### Example

SetGeneralParameter(clientId, device, deviceConnector, "", "AFG Num of Cycles per Second$2");

# Set AFG frequency

Use this paramString value to set the AFG frequency in MHz. This is the same as selecting the **Frequency (MHz)** control in the AFG area of the **Configuration** tab.

The value in bold font is the default value.

### Values:

- **AFG Frequency (MHz)$[20** | user_entered_value]

### Example

SetGeneralParameter(clientId, device, deviceConnector, "", "AFG Frequency (MHz)$20");

# Set AFG voltage level high

Use this paramString value to set the AFG voltage level high (in volts). This is the same as selecting the **Voltage Level High** control in the AFG area of the **Configuration** tab.

The value in bold font is the default value.

### Values:

- **AFG Voltage Level High (V)$[0.5** | user_entered_value]

### Example

SetGeneralParameter(clientId, device, deviceConnector, "", "AFG Voltage Level High (V)$0.5");

# Set AFG voltage level low

Use this paramString value to set the AFG voltage level low (in volts). This is the same as selecting the **Voltage Level Low** control in the AFG area of the **Configuration** tab.

The value in bold font is the default value.

### Values:

- **AFG Voltage Level Low (V)$[-5** | user_entered_value]

### Example

SetGeneralParameter(clientId, device, deviceConnector, "", "AFG Voltage Level Low (V)$-5");

# Set signal pattern validation mode

Use this paramString value to set the signal patern validation mode. This is the same as using the **Signal Validation** control on the **Configuration** tab.

The value in bold font is the default value.

### Values:

- **Pattern Validation$[Turn Off Signal Check** | Prompt me if Signal Check Fails]

### Example

SetGeneralParameter(clientId, device, deviceConnector, "", "Pattern Validation$ Turn Off Signal Check");

# SetTimeOut()

### SetTimeOut(clientId, time)

Sets a timeout period specified by the client. After this timeout period expires, the server is unlocked automatically.

### Parameters

| Parameter | Type | Direction | Description |
|---|---|---|---|
| clientId | string | IN | Identifier of the client that is connected to the server<br>clientId example (see page 62) |
| time | string | IN | The time in seconds that refers to the timeout period |

### Return value

String value that indicates the status of the operation upon completion. On success the return value is "TimeOut Period Changed".

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

*The server is LOCKED and the message displayed is "Server is locked by another client".*

*The session is UNLOCKED and the message displayed is "Lock Session to execute the command".*

*The server is NOTFOUND and the message displayed is "Server not found...Disconnect!".*

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

### Example

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, time)

### Comments

### Related command(s)

setTimeOut() xxx add link

# setVerboseMode()

### setVerboseMode(clientId, verboseMode)

This command sets the verbose mode to either true or false.

When the value is set to true, any message boxes that appear during the application are routed to the client machine that is controlling TekExpress.

When the value is set to false, all the message boxes are shown on the server machine.

### Parameters

| Parameter | Type | Direction | Description |
|---|---|---|---|
| clientId | string | IN | Identifier of the client that is connected to the server<br>clientId example (see page 62) |
| 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:*

*The server is LOCKED and the message displayed is "Server is locked by another client".*

*The session is UNLOCKED and the message displayed is "Lock Session to execute the command".*

*The server is NOTFOUND and the message displayed is "Server not found...Disconnect!".*

*When none of these fail conditions occur, then the message displayed 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<br>clientId example (see page 62) |
| 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)

ApplicationStatus (see page 61)

# Stop()

## Stop(clientId)

Stops the run operation.

### Parameters

| Parameter | Type | Direction | Description |
|---|---|---|---|
| clientId | string | IN | Identifier of the client that is connected to the server<br>clientId example (see page 62) |

### Return value

String that indicates the status of the operation upon completion.

### Example

```
returnVal = remoteObject.Stop(clientId);
if ((OP_STATUS)returnVal == OP_STATUS.SUCCESS)
   return “Stopped...”;
else
   return CommandFailed(returnVal);
```

### Comments

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

# TransferImages()

## 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<br>clientId example (see page 62) |
| filePath | string | IN | The location where the screen shots must be saved in the client.<br>***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:*

*The server is LOCKED and the message displayed is "Server is locked by another client".*

*The session is UNLOCKED and the message displayed is "Lock Session to execute the command".*

*The server is NOTFOUND and the message displayed is "Server not found...Disconnect!".*

*When none of these fail conditions occur, then the message displayed is "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:\Waveforms”)

# TransferReport()

## TransferReport(clientId, filePath)

This command transfers the report generated after the run to the specified folder (directory). The report contains the summary of the run. The client has to provide the location where the report is to be saved at the client-end.

### Parameters

| Parameter | Type | Direction | Description |
|---|---|---|---|
| clientId | string | IN | Identifier of the client that is connected to the server<br>clientId example (see page 62) |
| filePath | string | IN | Path to the target folder to which to transfer the report file. Enclose the path in quotes. |

### Return value

String that indicates the status of the operation upon completion.

### Example

TransferReport(clientId, "C:\Report")

### Comments

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

# UnlockSession()

## 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<br>clientId example (see page 62) |

### 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

LockSession (see page 76)

# Select panel parameters

| | Parameters | Default value | Options/range | Example |
|---|---|---|---|---|
| Select DUT Type | | | Device, Host | SelectDevice(clientId, device, true); |
| Select Test | | | | SelectTest(clientId, device, devicesuite, "UI-Unit Interval", boolSelect); |
| Select Test Method | | | USB-IF, DPOJET, Both | SetGeneralParameter(clientId, device, devicesuite, "", "Test Tool$USB-IF"); |
| Select Test Point | | | | SetGeneralParameter(clientId, device, devicesuite, "", "Version$Tx Pins - Near End"); |
| Configure DUT | | | true, false | SetGeneralParameter(clientId, device, devicesuite, "", "SSC On$true"); |
| Enable De-embed Link Filter (Test Fixture Effects) | | | true, false | SetGeneralParameter(clientId, device, devicesuite, "", "Deembed Filter Option$false"); |
| Select De-embed Filter File | | | | SetGeneralParameter(clientId, device, devicesuite, "", "Deembed Filter File Path$Tx_Device_TF_8G.flt"); |
| Enable Embed Link Filter (Ref Channel and Cable Effects) | | | true, false | SetGeneralParameter(clientId, device, devicesuite, "", "Embed Filter Option$true"); |
| Select Embed Filter File | | | | SetGeneralParameter(clientId, device, devicesuite, "", "Embed Filter File Path$Tx_Device_TF_8G.flt");<br>SetGeneralParameter(clientId, device, devicesuite, "", "CTLE Filter File Path$Tx_Device_TF_8G.flt"); |
| Enable CTLE Filter | | | true, false | SetGeneralParameter(clientId, device, devicesuite, "", "CTLE Filter Option$true"); |
| Instrument Configuration | Real Time Scope | <Instrument Address> | List from Instrument discovery | SetInstrument(clientId, device, devicesuite, "UI-Unit Interval", " AnalyzeInstrument$Real Time Scope$ DPO71254C (GPIB0::1::INSTR )") |
| | Signal Generator | <Instrument Address> | List from Instrument discovery | SetInstrument(clientId, device, devicesuite, "UI-Unit Interval", "AnalyzeInstrument$Signal Generator$AWG7122B (GPIB0::2::INSTR )"); |

| | Parameters | Default value | Options/range | Example |
|---|---|---|---|---|
| Selecting general parameters from Configuration panel | Probing Configuration | | List from available probing locations | SetGeneralParameter(clientId, device, devicesuite, "", "Probing Configuration$Differential (Ch1)") |
| | Record Length | | 500 – 10000000 | SetGeneralParameter(clientId, device, devicesuite, "", "Record Length$7000"); |

# Handle error codes

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

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

*The server is LOCKED and the message displayed is "Server is locked by another client".*

*The session is UNLOCKED and the message displayed is "Lock Session to execute the command".*

*The server is NOTFOUND and the message displayed is "Server not found...Disconnect!".*

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

# Limits Editor: compare string definitions

The following table lists the definitions of the limit comparison strings:

**Table 16: Limits Editor: comparison strings**

| Comparison string | Description |
|---|---|
| EQ(==) | Equal to |
| NE(!=) | Not equal to |
| GT(>) | Greater than |
| LT(<) | Less than |
| GE(>=) | Greater than or Equal to |
| LE(<=) | Less than or Equal to |
| GTLT(> <) | Greater than and Less than |
| GELE(>= <=) | Greater than or equal to and Less than or equal to |
| GELT(>= <) | Greater than or equal to and Less than |
| GTLE(> <=) | Greater than and Less or equal to |
| LTGT(< >) | Less than and Greater than |

**Table 16: Limits Editor: comparison strings (cont.)**

| Comparison string | Description |
|---|---|
| LEGE(<= >=) | Less than or equal to and Greater than or equal to |
| LEGT(<= >) | Less than or equal to and Greater than |
| LTGE(< >=) | Less than and Greater than or equal to |

# De-Embedding and channel embedding overview

There are five basic filter files to meet the following combinations:

- Host + 3m cable (Host_Channel_Back_Panel_3M_Cable_12.5G.flt
- Device + 3m cable (Device_Channel_3M_cable_12.5G.flt)
- Device fixture (Tx_Device_TF_8G.flt)
- Host fixture (TX_Host_TF_8G.flt)
- USB3CTLE ( same for Device and Host )

***NOTE.*** *There is no filter for Host front + 3m cable.*

There are four S-parameter files; two for test fixtures (Device and Host) and two for Reference channels (Device and Host).

- USB-IF_ENA_DEVICE_CHANNEL_3MCABLE.s4p
- USB-IF_ENA_HOST_CHANNEL_3MCABLE.s4p
- INTEL DEVICE FIXTURE_PLUS SHORT CABLE.s4p
- INTEL HOST FIXTURE.s4p

Front Panel and capacitive devices do not use a short cable. As the S-parameter files represent the combined reference channel and short cable parameters, the USB-IF does not provide S-parameter files for these devices. Front Panel and capacitive devices need a back channel with no cable

***NOTE.*** *There is one set of filter files that support both 25 GS/s (8 GHz BW oscilloscopes) and 50 GS/s (12.5 GHz BW and above oscilloscopes) sampling rates. The difference between the 25 GS/s and 50 GS/s filters is that the 25 GS/s filters have a stop band setting of 10 GHz for embed filters.*

### See also

Host Filter Information (see page 113)
Device Filter Information (see page 116)

DUT/Filter Combinations (see page 118)

# Host filter information

## Host embed filter

- The Host embed filter name is Device_Channel_3M_cable_12.5G.flt.
- This filter is applied for HOST DUT and for the normative CP0 and CP1 measurements.
- The test point location is compliance TP1. The application uses the device and cable compliance channels to test the host designs.
- The filter response is generated using the SDLA when the input is set as SigTest (USB-IF) for the USB-IF_ENA_HOST_CHANNEL_3MCABLE.s4p file. The S-parameter file represents the combined response of the HOST reference channel and the 3 meter cable.
- This filter embeds the response of the Host 5 inch reference channel and a 3 meter cable.
- The filter response BW is 12.5 GHz and the stop band is 15.625 GHz at an -80 dB roll off.

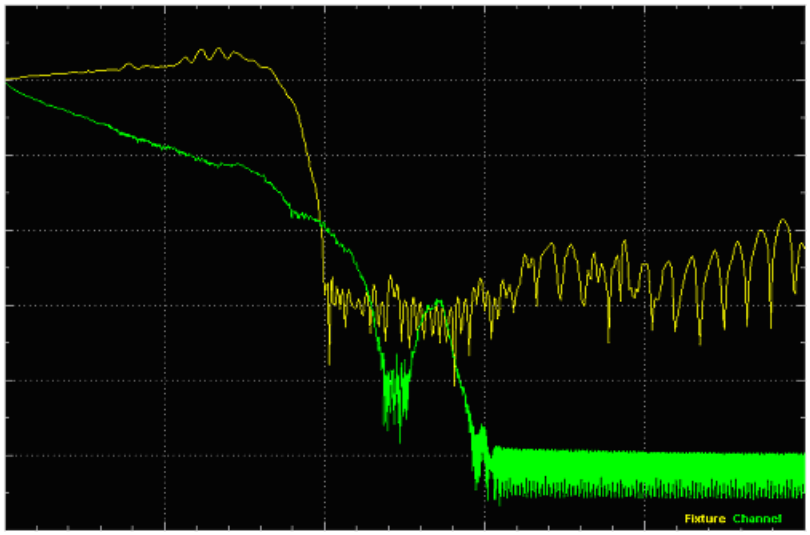
## Host de-embed filter

The Host de-embed filter file is Tx_Host_TF_8G.flt, which de-embeds the Host USB-IF test fixture. This filter is convolved with the Device_Channel_3M_cable_12.5G.flt filter to remove the effects of the test fixture.

The application uses the 'Intel Host fixture.s4p' S-parameter file to generate the filter response used by USB-IF.

The filter response BW is 8 GHz and the stop band is 10 GHz at an -80 db roll off.

The following is a representative plot, generated using SDLA. You can use this plot as a reference plot, or to view the response of the filter file.




Host fixture and channel response

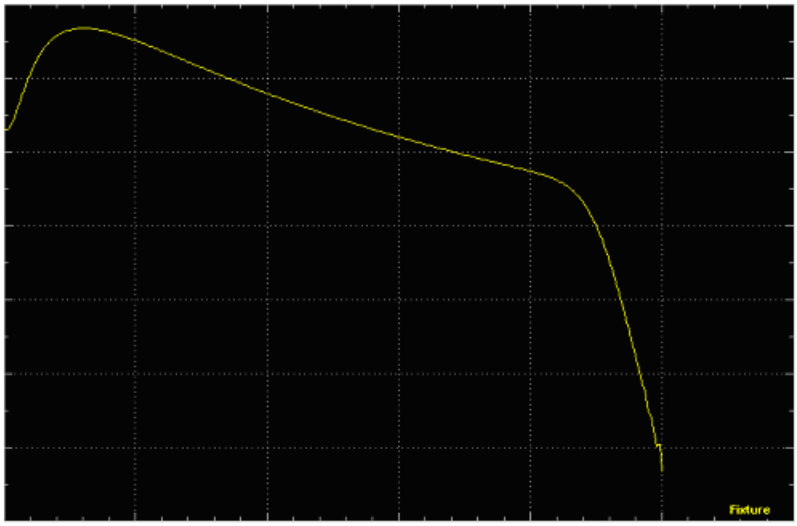
### Continuous time linear equalizer (CTLE) filter

The USB3CTLE.flt filter is a Continuous Time Linear Equalizer (CTLE) filter. Due to the lossy nature of the channel (the combination of the reference channel, cable, and test fixture from TX pins(TP1)), the eye diagram at the receiver may be closed. This filter applies receiver equalization to meet the system timing and voltage margins.

The CTLE filter coefficients are generated by passing the following parameters to the SDLA:

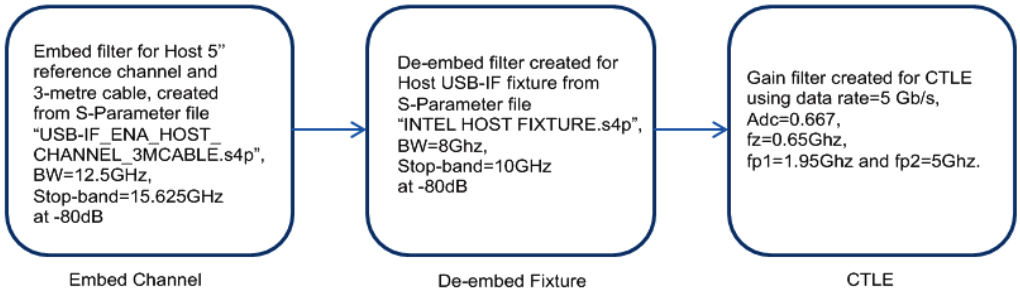
- DC gain (Adc) = 0.667
- Zero frequency (fz) = 650 MHz
- first pole frequency (fp1) = 1.95 Ghz
- Second pole frequency (fp2) = 5 GHz

The following is a representative plot, using SDLA, of the combined response of the convolution of Embed, De-embed and CTLE filters. You can use this plot as a reference plot, or to view the response of the filter file.




CTLE response

The signal connection path for HOST is HOST <-> TF (B connector) <-> 3 meter reference cable <-> standard (1 meter) SMA cable <-> Oscilloscope channel.




***NOTE.*** *These filters are applicable for 12.5 GHz and above oscilloscopes. For 8 GHZ oscilloscopes, only the embed filters BW (8 GHz) and stop band (10 GHz) are different; the de-embed parameters remain the same.*

## See also

De-Embedding and Channel Embedding Overview (see page 112)
Device Filter Information (see page 116)
DUT/Filter Combinations (see page 118)

# Device filter information

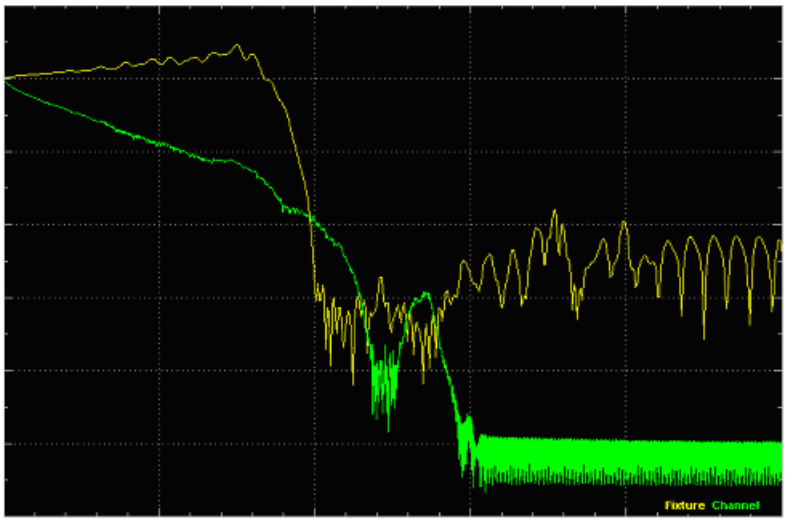
## Device embed filter

- The embed filter file is Host_Channel_Back_Panel_3M_Cable_12.5G.flt, and is applied for the Device DUT for the normative (CP0 and CP1) measurements.
- The test point location is compliance TP1.
- The application uses host and cable compliance channels for testing of device designs.
- This filter response is generated using the Tektronix SDLA filter generation application, with input from the USB-IF S-parameter file (USB-IF_ENA_DEVICE_CHANNEL_3MCABLE.s4p).
- This filter embeds the response of the Device 11 inch reference channel and a 3 meter cable.
- The filter response BW is set to 12.5 GHz and the stop band is set to 15.625 GHz at an -80 dB roll off.

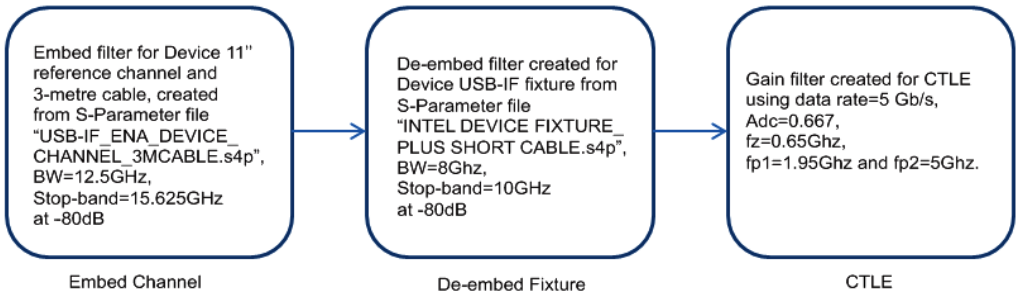
## Device de-embed filter

- The de-embed filter is Tx_Device_TF_8G.flt, and is applicable for Host.
- This is convolved with the Host_Channel_3M_cable_12.5G.flt filter to remove the test fixture effects.
- The filter response is generated using the 'Intel Device fixture.s4p' filter output provided by USB-IF.
- The filter de-embeds the Device USB-IF test fixture.
- The filter response BW is set to 8 GHz and the stop band is set to 10 GHz at an -80 db roll off.

The following is a representative plot, using SDLA, of the combined response of the convolution of Embed, De-embed and CTLE filters. You can use this plot as a reference plot, or to view the response of the filter file.



The connection path for DEVICE is DEVICE <-> Short USB cable (4 inch) <-> TF (A connector) <-> 3 meter reference cable <-> TF_convertor* <-> standard (1m) SMA cable <-> Oscilloscope channel.




***NOTE.*** *The TF_convertor uses input from the 3 meter USB cable and provides output as SMA. This convertor is needed for testing the far end of the DUT.*

***NOTE.*** *These filters are applicable for 12.5 GHz and above oscilloscopes. For 8 GHZ oscilloscopes, only the embed filters BW (8 GHz) and stop band (10 GHz) are different; the de-embed parameters remain the same.*

## See also

De-Embedding and Channel Embedding Overview (see page 112)
Host Filter Information (see page 113)
DUT/Filter Combinations (see page 118)

# DUT/Filter combinations

Use the following table to help select the appropriate filter for listed DUT and embed/de-embed configurations.

**Table 17: Host/Device DUT embed, de-embed filter combinations**

| DUT | Test point (TP) location | Embed filter (reference channel + cable effects) | De-embed filter (test fixture effects) | USB IF recommendation |
|---|---|---|---|---|
| Host | Compliance TP1 | N/A | N/A | Informative |
| Host | Compliance TP1 | Device_Channel_3M_Cable_12.5G.flt (default) | Tx_Host_TF_8G.flt | Normative |
| Host | Tx Pins (host connector) | N/A | N/A | Informative |
| Host | Custom | Custom_Device_Channel_3M_Cable.flt | Custom_Host_TF.flt | Informative |
| Device | Compliance TP1 | N/A | N/A | Informative |
| Device | Compliance TP1 | Host_Channel_Back_Panel_3M_Cable_12.5Gflt (default) | Tx_Device_TF_8G.flt | Normative |
| Device | Tx Pins (device connector) | N/A | N/A | Informative |
| Device | Custom | Custom_Host_Channel_3M_Cable.flt | Custom_Device_TF.flt | Informative |
| Device | Far end (TP1) | Tx_Device_Channel_3M_Cable.flt | Tx_Device_TF.flt | Normative |

### See also

De-Embedding and Channel Embedding Overview (see page 112)
Host Filter Information (see page 113)
Device Filter Information (see page 116)

# SDLA filter creation requirements

You can use the optional Tektronix Serial Data Link Analysis (SDLA) software to create custom bandwidth filters using supplied or customsp4 files. See the SDLA documentation and application help for information on running the SDLA application.

Verify that you are using the correct versions of software before using SDLA to create filter files:

- SDLA: 1.2.90.34
- TekScope: 5.3.4 Build 25

### See also

Setting up SDLA to generate USB Tx filters (see page 119)
To create a CTLE filter using SDLA (see page 120)
To create a host filter using SDLA (see page 123)
To create a device back panel filter using SDLA (see page 121)

## Setting up SDLA to generate USB Tx filters

Enter the following settings in SDLA to generate filter files for USB Tx:

1. Recall the CP0.wfm file on Ref1 that was acquired with a 50 GS/s oscilloscope with 12.5 GHz and above bandwidth. Acquire waveforms with a 25 GS/s oscilloscope with 8 GHz bandwidth.
2. In TekScope, select **Analyze > Serial Data Link Analysis** (SDLA).
3. Set the SDLA application parameters:
   a. Oscilloscope Source: **Ref 1**.
   b. Bit Rate: **5 Gb/s**.
   c. Set the **Equalizer** parameters:
      - Source: **Ref 1**.
      - Rate: **5 Gb/s**.
      - Select **Standard** and **CTLE**.
      - Adc: **0.667**.
      - fz: **0.65 GHz**.
      - fp1: **1.95 GHz**.
      - fp2: **5 GHz**.
4. In the next panel, make sure No adapt, FFE/DFE, and PcieD are not selected (unchecked).
5. Use the To create a CTLE filter using SDLA (see page 120), To create a device back panel filter using SDLA (see page 121), and To create a host filter using SDLA (see page 123) topics to generate the required filter.
6. Copy the generated filters to C:\Program Files\Tektronix\TekExpress\Setup Files\Filters for use in the USB Tx application.

### See also

SDLA filter creation requirements (see page 118)

## To create a CTLE filter using SDLA

1. In SDLA, click the **Equalizer** block. Unselect all other blocks.
2. Turn off test points **TpA**, **TpB**, and **TpC** (test point text changes to blue).
3. Click on the **Equalizer** block.
4. Click the **Run EQ** button. Wait until the status bar indicates Single Run completed.
5. Click **Ok**. The generated CTLE Filter (sdlaCtle.flt) is saved at C:\TekApplications\SDLA\output filters. Rename the CTLE filter file as needed.
6. Copy the generated filters to C:\Program Files\Tektronix\TekExpress\Setup Files\Filters for use in the USB Tx application.

### See also

SDLA filter creation requirements (see page 118)
Setting up SDLA to generate USB Tx filters (see page 119)
To create a host filter using SDLA (see page 123)
To create a device back panel filter using SDLA (see page 121)

# To create a device back panel filter using SDLA

1. In SDLA, click the Fixture block and set/verify the following parameters:
   - Fixture: **On**.
   - Data input Type: **S-Parameters**.
   - Touchstone Format:**4-Port**.
   - Click **Browse** and select the sparam file **Intel Device fixture_plus short cable.s4p**.
   - Derive Filter From: **Single Ended**.
   - Assign ports: **(1 Tx+, 3Rx+)** and **(2Tx-, 4Rx-)**.
   - Bandwidth Limit: **Custom**.
   - Click Filter: **BW->8 GHz** and **Stopband->10 GHz** (for example, 1.25 * BW) and **-80 dB**. Click **Apply**.
   - Click **OK**.
2. Click Apply. The application saves the generated de-embed filter (sdlaTpA.flt) to C:\TekApplications\SDLA\output filters. Rename the filter file as needed.
3. Click the **Channel** block and set/verify the following parameters:
   - Channel: **On**.
   - Data input Type: **S-Parameters**.
   - Touchstone Format: **4-Port**.
   - Click **Browse** and select the sparam file **AGILENT_ENA_DEVICE_CHANNEL_3MCABLE.s4p**.
   - Derive Filter From: **Single Ended**.
   - Assign ports: **(1 Tx+, 3Rx+)** and **(2Tx-, 4Rx-)**.
   - Bandwidth Limit: **Custom**.
   - Click Filter: **BW->12.5 GHz** and **Stopband->15.625 GHz** (for example, 1.25 * BW) and **-80 dB**. Click **Apply**.
   - Click **OK**. Make sure that the Fixture and Emphasis blocks are set to **Off** (disabled).
4. Click **Apply**. The application saves the generated de-embed filter file (sdlaTpC.flt) to C:\TekApplications\SDLA\output filters. Rename the filter file as needed.

5. Click the **Emphasis** block and set/verify the following parameters:
   - Emphasis: **De-**.
   - Data input Type: **Read From File**.
   - Click **Browse** and select the previously created file **USB3_Ctle.flt**.
   - Bandwidth Limit: **None**.
   - Click **OK**.
6. Turn on (enable) the **Fixture**, **Emphasis**, and **Channel** blocks.
7. Click **Apply**. The status bar shows the message 'Press Analyze to measure with DPOJET.'
8. Verify the SDLA plots.
9. Click **Analyze** to verify the eye using DPOJET. The application saves the generated convolved filter file as sdlaTpC.flt.
10. Copy the generated filters to C:\Program Files\Tektronix\TekExpress\Setup Files\Filters for use in the USB-TX application.

## See also

SDLA filter creation requirements (see page 118)
Setting up SDLA to generate USB Tx filters (see page 119)
To create a CTLE filter using SDLA (see page 120)
To create a host filter using SDLA (see page 123)

# To create a host filter using SDLA

1. In SDLA, click the Fixture block and set/verify the following parameters:
   - Fixture: **On**.
   - Data input Type: **S-Parameters**.
   - Touchstone Format: **4-Port**.
   - Click **Browse** and select the sparam file **Intel Host fixture.s4p**.
   - Derive Filter From: **Single Ended**.
   - Assign ports: **(1Tx+, 3Rx+)** and **(2Tx-, 4Rx-)**.
   - Bandwidth Limit: **Custom**.
   - Click on Filter: **BW->8 GHz** and **Stopband->10 GHz** (for example, 1.25 * BW) and **-80 dB**. Click **Apply**.
   - Click **OK**.
2. Click **Apply**. The application saves the generated de-embed filter (sdlaTpA.flt) to C:\TekApplications\SDLA\output filters.
3. Click the Channel block and set/verify the following parameters:
   - Channel: **On**.
   - Data input Type: **S-Parameters**.
   - Touchstone Format: **Select 4-Port**.
   - Click Browse and select the sparam file **AGILENT_ENA_HOST_CHANNEL_3MCABLE.s4p**.
   - Derive Filter From: **Single Ended**.
   - Assign ports: **(1 Tx+, 3Rx+)** and **(2Tx-, 4Rx-)**.
   - Bandwidth Limit: **Custom**.
   - Click on Filter: **BW->12.5 GHz** and **Stopband->15.625 GHz** (for example, 1.25 * BW) and **-80 dB**. Click on **Apply**.
   - Click **OK**.
4. Turn off (disable) the **Fixture** and **Emphasis** blocks.
5. Click **Apply**. The application saves the generated de-embed filter (sdlaTpC.flt) to C:\TekApplications\SDLA\output filters.

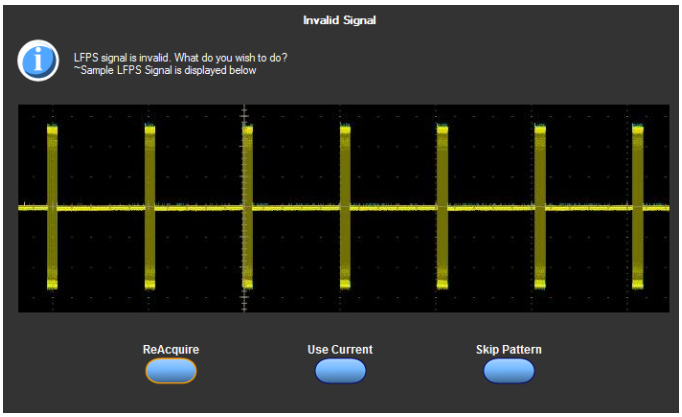
6. Click the **Emphasis** block and set/verify the following parameters:
   - Emphasis: **De-**.
   - Data input Type: **Read From File**.
   - Click **Browse** and select the previously created **USB3_Ctle.flt** filter file.
   - Bandwidth Limit: **None**.
   - Click **OK**.
7. Turn on (enable) the **Fixture**, **Emphasis**, and **Channel** blocks and click **Apply**. The status bar displays the message 'Press Analyze to measure with DPOJET'.
8. Verify the SDLA Plots.
9. Click **Analyze** to verify the eye using DPOJET. The application generates the convolved filter file **sdlaTpC.flt**.
10. Copy the generated filters to C:\Program Files\Tektronix\TekExpress\Setup Files\Filters for use in the USB Tx application.

### See also

SDLA filter creation requirements (see page 118)
Setting up SDLA to generate USB Tx filters (see page 119)
To create a CTLE filter using SDLA (see page 120)
To create a device back panel filter using SDLA (see page 121)

## LFPS pattern type validation

When the Pattern type validation is set to Yes, during the acquisition of LFPS pattern, a signal validation occurs. If the pattern is valid, the measurement continues normally. If the pattern is not valid, the application opens a message dialog box:



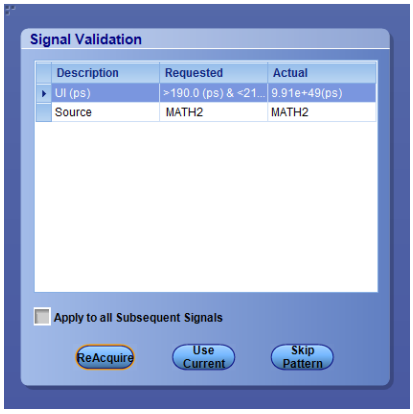
***NOTE.*** *If Pattern type validation is selected as "No", then the measurement continues with the acquired waveform.*

***NOTE.*** *Signal validation is not done for the SIGTest test method.*

- Click **Reacquire** to start the acquisition again.
- Click **Use Current** to continue with the currently acquired waveform.
- Click **Skip Pattern** to skip all LFPS tests. The rest of the selected measurements continue.

# CP0 pattern type validation

When the Pattern type validation is set to Yes, the application validates the CP0 pattern during the acquisition. If the pattern is valid, the measurement continues normally. If the pattern is invalid, the following pop up displays.




***NOTE.*** *If Pattern type validation is selected as "No", then the measurement continues with the acquired waveform.*

- Click **Reacquire** to start the acquisition again.
- Click **Use Current** to continue with the currently acquired waveform.
- Click **Skip Pattern** to skip all CP0 tests. The rest of the selected measurements continue.

# CP1 pattern type validation

Refer to the topic Oscilloscope-based toggle (see page 126) and its related topics.

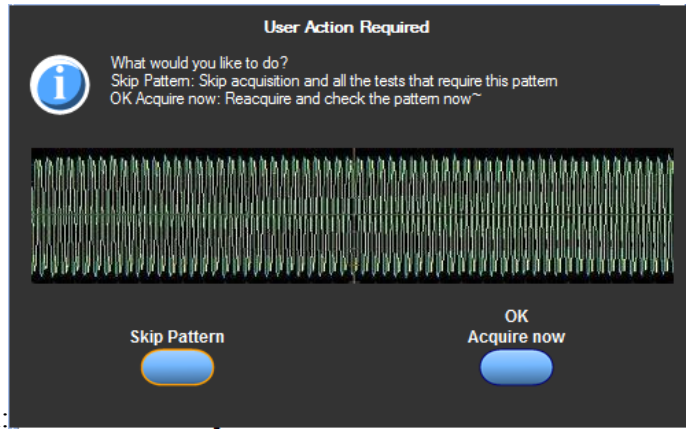
# Oscilloscope-based toggle

To use the oscilloscope based toggle, do the following:

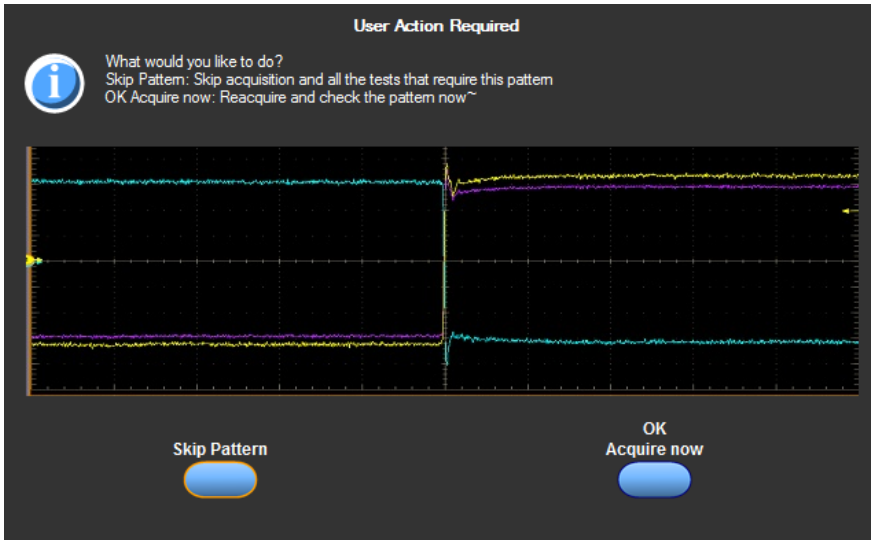
***NOTE.*** *Oscilloscope based toggle is not guaranteed to work for all DUTs.*

1. In the Configuration panel, for the parameter **CP0 CP1 CP7 Toggle using**, select an oscilloscope (For example DPO72004 (TCPIP::192.158.96.152::INSTR)).
2. Connect the AUX OUT from the oscilloscope to the USB 3.0 Device Fixture 2 RX+ and connect a USB cable from USB 3.0 Device Fixture 2 to Device fixture 1.
3. Click the **Run** button. If the CP1 measurements are selected, then when the CP1 pattern is being acquired, a pop up displays to prompt you to make the necessary connections. Select to either skip the pattern or make a new acquisition after the DUT is transmitting CP1.

For CP1 signal:

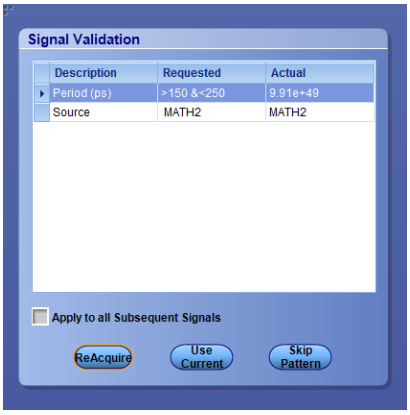



For CP7 signal:



4. If you click **OK (Acquire Now)**, the application takes a new acquisition. If Pattern Type validation is set to Yes, a Pattern Type validation is done on the acquired signal to check if it is a CP1 signal. If it is a CP1 signal, the measurements continue normally. If not, the application shows the following dialog box.

***NOTE.*** *If Pattern type validation is set to No, then the measurement continues with the acquired waveform.*




5. Choose how to continue:

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

   ■ Select **Use Current** to continue measurements using this acquired waveform.

   ■ Select **Skip Pattern** to skip all CP1 tests. The rest of the selected measurements are taken. If CP1 is skipped and CP0 is aquired, TJ and RJ are computed on CP0 for informational purposes.

### See also

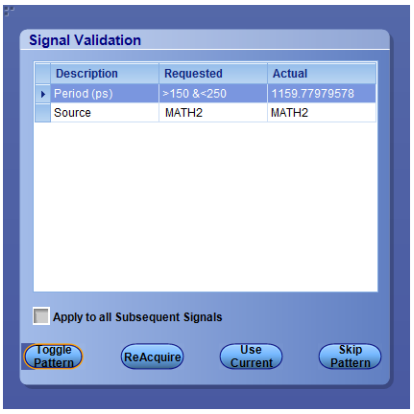
AWG-based toggle (see page 127)
AFG-based toggle (see page 128)
Manual toggle (see page 130)

## AWG-based toggle

To use the arbitrary waveform generator (AWG) based toggle method, do the following:

1. In the configuration panel, for the parameter **CP0 CP1 CP7 Toggle using**, select an AWG (For example AWG7122C (TCPIP::192.158.96.152::INSTR)).
2. Connect the interleave (analog and $\overline{\text{analog}}$) output of Ch1 of the AWG to the USB 3.0 Device Fixture 2 (RX+ and RX-) and connect a USB cable from the USB 3.0 Device Fixture 2 to USB 3.0 Device fixture 1.

3. Click the **Run** button. If the CP1 measurements are selected, then when the CP1 pattern is being acquired, a command is sent to the AWG to send a trigger to toggle the DUT from CP0 to CP1. Next, the waveform is acquired. If Pattern type validation is set to Yes, then the validation occurs. If the pattern is valid, the measurement continues normally. If the pattern is not valid, the application opens a dialog box similar to the following image:




***NOTE.*** *If Pattern type validation is set to No, then the measurement continues with the acquired waveform.*

4. Choose how to continue:
   - Click **Toggle Pattern** to reinitiate the toggle sequence to toggle the DUT. (The pop up remains displayed during this toggle process.) You can visually verify whether the acquired pattern is correct. If not, keep clicking the Toggle Pattern button until the correct pattern is acquired.
   - Click **Reacquire** to start the acquisition again.
   - Click **Use Current** to continue with the currently acquired waveform.
   - Click **Skip Pattern** to skip the current CP tests. The rest of the selected measurements continue.

### See also

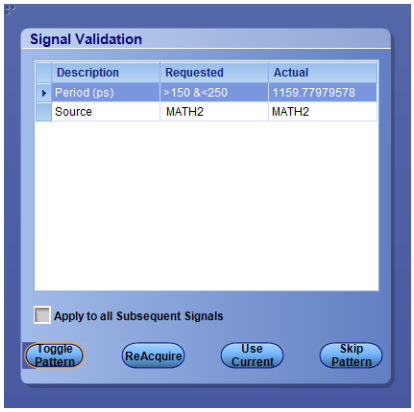
Oscilloscope-based toggle (see page 126)
AFG-based toggle (see page 128)
Manual toggle (see page 130)

# AFG-based toggle

To use the arbitrary function generator (AFG) based toggle, follow this procedure.

1. In the configuration panel, select an AFG instrument for the parameter CP0-CP1 Toggle (For example: AFG3102 (TCPIP::192.158.96.152::INSTR)).
2. Connect Ch1 of the AFG to the Device fixture 2 (RX+).

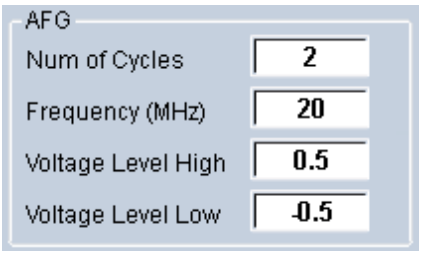
3. Connect a 3 meter USB cable from Device fixture 2 to Device fixture 1.
4. Click the **Run** button. If the CP1 measurements are selected, a command is sent to AFG, when the CP1 pattern is being acquired, to toggle the DUT from CP0 to CP1. Next, the pattern is acquired. If Pattern type validation is set to Yes, then the validation occurs. If the pattern is valid, the measurement continues normally. If the pattern is not valid, the application opens a dialog box similar to the following image:




***NOTE.*** *If Pattern type validation is set to No, then the measurement continues with the acquired waveform.*

5. Choose how to continue:
   - Click **Toggle Pattern** to reinitiate the toggle sequence to toggle the DUT. (The pop up remains displayed during this toggle process.) You can visually verify whether the acquired pattern is correct. If not, keep clicking the Toggle Pattern button until the correct pattern is acquired.
   - Click **Reacquire** to start the acquisition again.
   - Click **Use Current** to continue with the currently acquired waveform.
   - Click **Skip Pattern** to skip the current CP tests. The rest of the selected measurements continue.

## User-Configurable AFG parameters




You can configure the following parameters in the Configuration panel before the start of Test Execution when AFG is set as the toggle tool:

- **Num of Cycles**: Number of cycles per second. The range is from 1 to 5. The default value is 2.
- **Frequency (MHz)** : The range is from 10 MHz to 100 MHz. The default value is 20 MHz.
- **Voltage Level High**: The range is from –5 V to 5 V. The default value is 0.5 V.
- **Voltage Level Low**: The range is from –5 V to 5 V. The default value is –0.5 V.

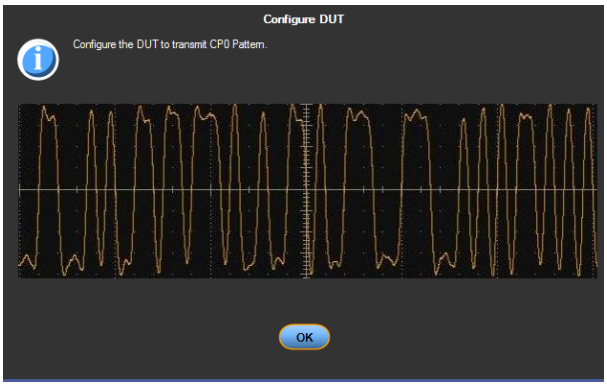
### See also

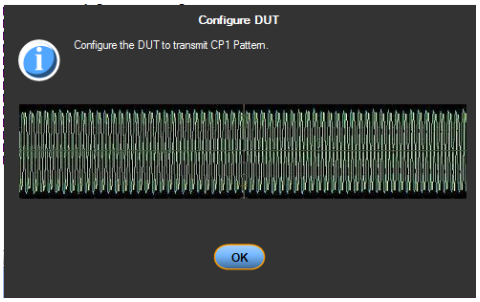
Oscilloscope-based toggle (see page 126)
AWG-based toggle (see page 127)
Manual toggle (see page 130)

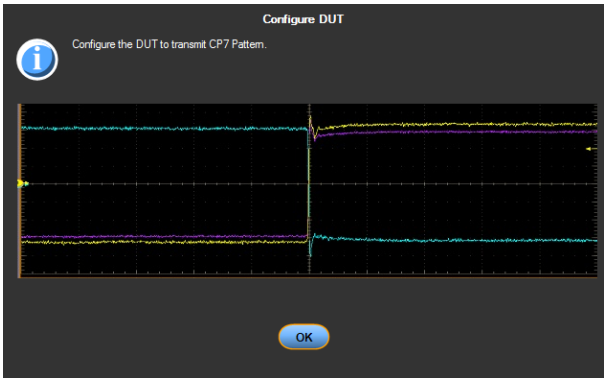
# Manual toggle

To not use the AWG-based toggle capability, do the following:

1. Click **Setup > Configuration > Global Settings**.
2. In the Instruments Detected field, set the **CP0 CP1 CP7 Toggle using** parameter to **Do not use**.
3. Run the test. When the application needs to acquire the CP0, CP1, or CP7 pattern, it opens windows similar to the following graphics, prompting you to manually transmit the pattern signal and acquire the waveform, and validate it against the displayed waveform.



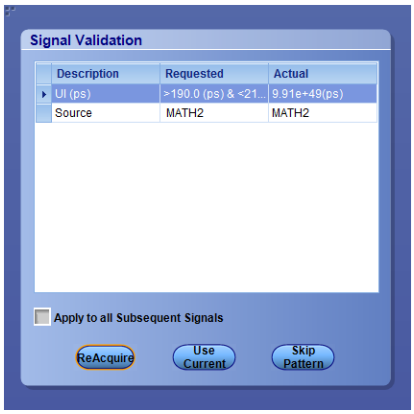





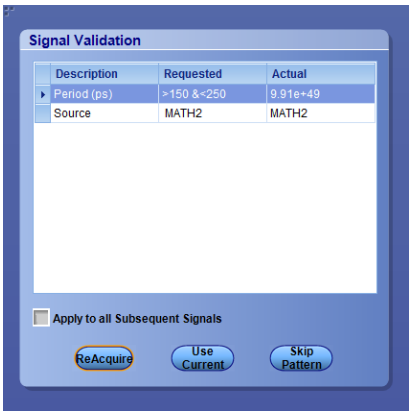

4. Click **OK** to acquire the waveform. If Pattern type validation is set to **Prompt me if Signal Check Fails**, the application runs a pattern type validation on the acquired signal. If the acquired signal is a valid pattern, the measurement continues normally.

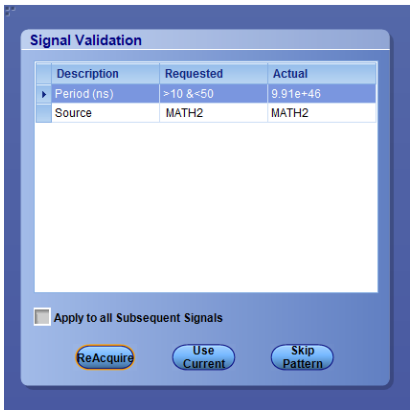
   If it is not a valid pattern, the application shows one of the following message dialog boxes:

CP0:




CP1:



CP7: 


***NOTE.*** *If Pattern type validation is set to Turn Off Signal Check, then the measurement continues with the acquired waveform.*

5. Choose how to continue:
   - Click **Reacquire** to start the acquisition again.
   - Click **Use Current** to continue with the currently acquired waveform.
   - Click **Skip Pattern** to skip the current tests. The rest of the selected measurements continue.

## See also

Oscilloscope-based toggle (see page 126)
AWG-based toggle (see page 127)
AFG-based toggle (see page 128)

# Index

## K



## L



## M



## N



## O



## P



## R



## S

## T



## U



## V



## W



## X