TekExpress® TDEC Solution

Printable Application Help

077-1204-00

TekExpress® TDEC Solution

Printable Application Help

www.tektronix.com
077-1204-00

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.

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

Pre-measurement calibration

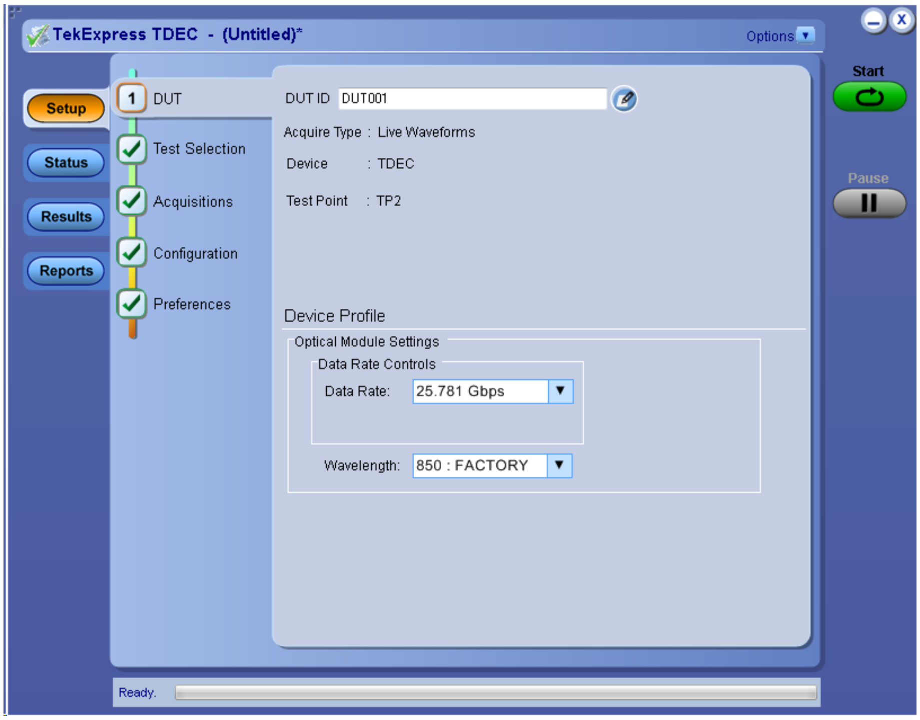
Saving and recalling test setup files

TDEC compliance measurements

TekExpress programmatic interface

Reference

Welcome



DUT ID DUT001

Acquire Type : Live Waveforms

Device : TDEC

Test Point : TP2

Device Profile

Optical Module Settings

Data Rate Controls

Data Rate: 25.781 Gbps

Wavelength: 850 : FACTORY

Ready.

Welcome to the TekExpress TDEC (Transmitter and Dispersion Eye Closure) Automated Measurement Solution Software applications. The TekExpress TDEC provides turnkey testing and characterization of Tx Optical properties through TDEC measurement as per SR4 Short Reach Ethernet Specifications outlined in IEEE 802.3bm. Customers can change the test limits and perform margin testing as part of extended product characterization.

Key features of TekExpress TDEC include:

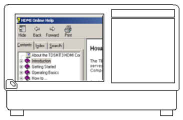
- 80STDEC offers streamlined high performance Transmitter and Dispersion Eye Closure (TDEC) on a variety of Tektronix optical acquisition modules. Ideal for manufacturing settings and minimal conformance validation applications.

Getting help and support

Related documentation

The following documentation is available as part of the TekExpress® TDEC Solution application.

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 PDF file that ships with TDEC Solution software distribution (*TekExpress TDEC-Automated-Test-Solution-Software-Printable-Help-EN-US.pdf*).

See also

Technical support

Conventions

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* at the front of this document for contact 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
- Modules 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

Getting started

Minimum system requirements

The following table shows the minimum system requirements to install and run the TekExpress TDEC Solution.

Table 3: System requirements

Component	Description
Oscilloscope	■ Tektronix DSA8300 Digital Serial Analyzer ■ Frimware Version: 6.3.1.3 or greater ■ 80SJNB Software Version: 3.2.4.0 or greater ■ Opt ADVTRIG ■ Opt JNB01/02 ■ 80CXX series Optical Sampling module
Processor	Same as the oscilloscope
Operating System	Same as the oscilloscope:
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 6.3.1.3 or greater (for Windows 7) ■ 80SJNB Software Version: 3.2.4.0 or greater
Software	■ IronPython 2.7.3 installed ■ PyVisa 1.0.0.25 installed ■ Microsoft .NET 4.0 Framework ■ Microsoft Internet Explorer 7.0 SP1 or greater, or other Web browser for viewing reports ■ Adobe Reader software 7.0 or greater for viewing portable document format (PDF) files
Other Devices	■ Microsoft compatible mouse or compatible pointing device. ■ Two USB ports (four USB ports recommended).

[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.

Instruments and accessories required

TDEC application is launched on DSA8300 sampling scope. The following table lists the instruments and accessories required for this application.

Table 4: Instruments and accessories required for 100GBASE-SR4 application

Instrument/Accessory	Model number	Quantity
Sampling Oscilloscope	DSA8300	1
Clock Recovery Unit	CR286A	1
Optical Modules	80C15-CRTP-MM for Compliance mode 80C10/80C14/80C15/80C15-CRTP-MM for User Defined mode	1
Phase Reference	82A04B (Optional) [2]	1
SMA Cables	174-6023-00 : To connect sub rate clock out of CRU to trigger [3]	1
	174-6023-00 : To connect clock out of CRU to phase reference module	1
	174-6023-00 : To connect electrical output of the 80C15 module to CRU input. Deskew the cables before use.	2
VBR (Variable Back Reflector)	A Splitter and a Variable Back Reflector is required for TDEC measurement, as specified in *Equipment setup*. The splitter should be the smallest split possible to meet the back reflection requirement. For most accurate result measure the back reflection.	1

[2] Required to reach jitter noise floors below 100fsec

[3] Available with 80A08 accessory kit

Installing the software

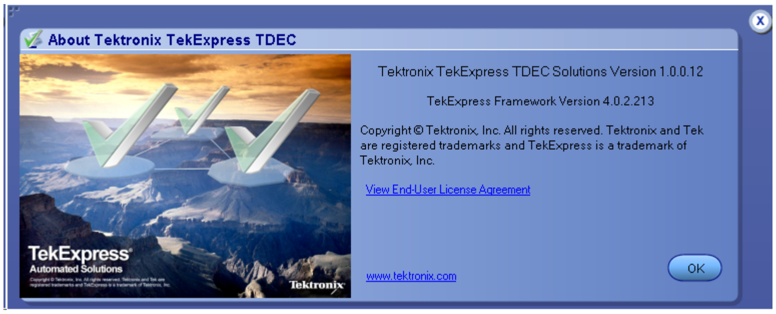
Follow the steps to download and install the latest TekExpress TDEC Solution. See *Minimum system requirements* for compatibility.

1. Type the URL *www.tek.com* in the address bar of web browser and click Software Downloads
2. Enter **TekExpress TDEC Solution** in the *Enter your keywords* field, and click **Search**
3. Select the latest version of software and follow the instructions to download. Copy the executable file into the oscilloscope.
4. Double-click the executable and follow the on-screen instructions. The software is installed at `C:\Program Files\Tektronix\TekExpress\TekExpress TDEC\`
5. Map *My TekExpress folder to drive X:* and Set *My TekExpress folder permissions*
6. Select **Application** > **TDEC** from the TekScope menu to *launch the application*

View software version

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

To view version information for TDEC, click ▼ button in the TekExpress application and select **About TekExpress**.



About Tektronix TekExpress TDEC

Tektronix TekExpress TDEC Solutions Version 1.0.0.12

TekExpress Framework Version 4.0.2.213

Copyright © Tektronix, Inc. All rights reserved. Tektronix and Tek are registered trademarks and TekExpress is a trademark of Tektronix, Inc.

View End-User License Agreement

www.tektronix.com

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.*

My TekExpress folder settings

Map My TekExpress folder to drive X:

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

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

Shared **My TekExpress** folder is mapped to drive **X:** on the instrument running the TDEC application. TDEC uses this shared folder to save session waveform files and for other application file transfer operations.

Follow the below procedure 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 My TekExpress folder permissions*

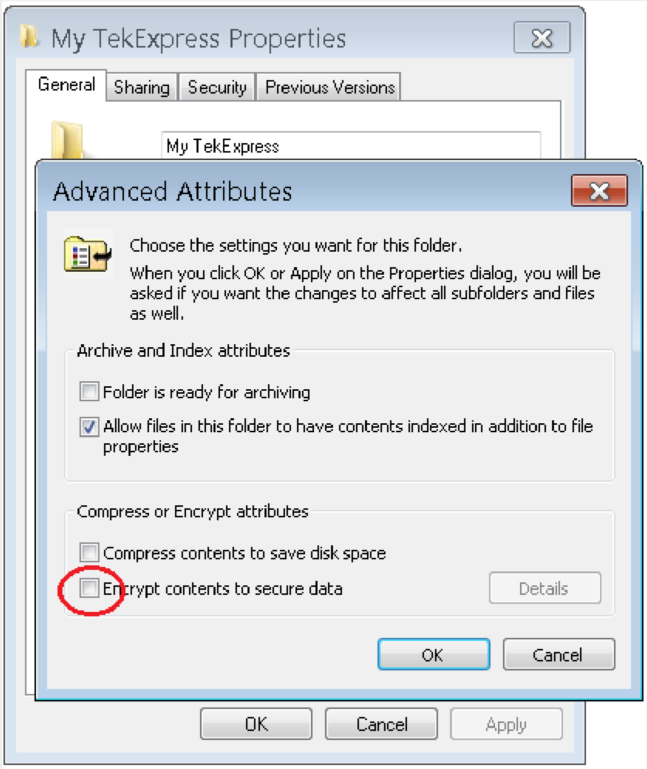
Application directories

File name extensions

Set 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.


My TekExpress Properties

Advanced Attributes

Choose the settings you want for this folder.

When you click OK or Apply on the Properties dialog, you will be asked if you want the changes to affect all subfolders and files as well.

Archive and Index attributes

□ Folder is ready for archiving

☑ Allow files in this folder to have contents indexed in addition to file properties

Compress or Encrypt attributes

□ Compress contents to save disk space

□ Encrypt contents to secure data

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

See also. *Map My TekExpress folder to Drive X:*

Application directories

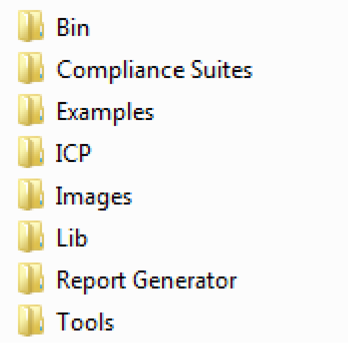
File name extensions

Application directories

TekExpress TDEC application

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

```
C:\Program Files\Tektronix\TekExpress\TekExpress TDEC
```


Bin
Compliance Suites
Examples
ICP
Images
Lib
Report Generator
Tools

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

Table 5: Application directories and usage

Directory names	Usage
Bin	Contains TekExpress TDEC application libraries
Compliance Suites	Contains compliance-specific files
Examples	Contains various support files
ICP	Contains instrument and TekExpress TDEC application-specific interface libraries
Images	Contains images of the TekExpress TDEC application
Lib	Contains utility files specific to the TekExpress TDEC application
Report Generator	Contains style sheets for report generation
Tools	Contains instrument and TekExpress TDEC application-specific files

See also

View test-related files

File name extensions

File name extensions

The TekExpress TDEC 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 Application log files
.csv	Test waveform files
.mht	Test result reports (default) Test reports can also be saved in HTML format
.pdf	Test result reports Application help document
.xslt	Style sheet used to generate reports

See also

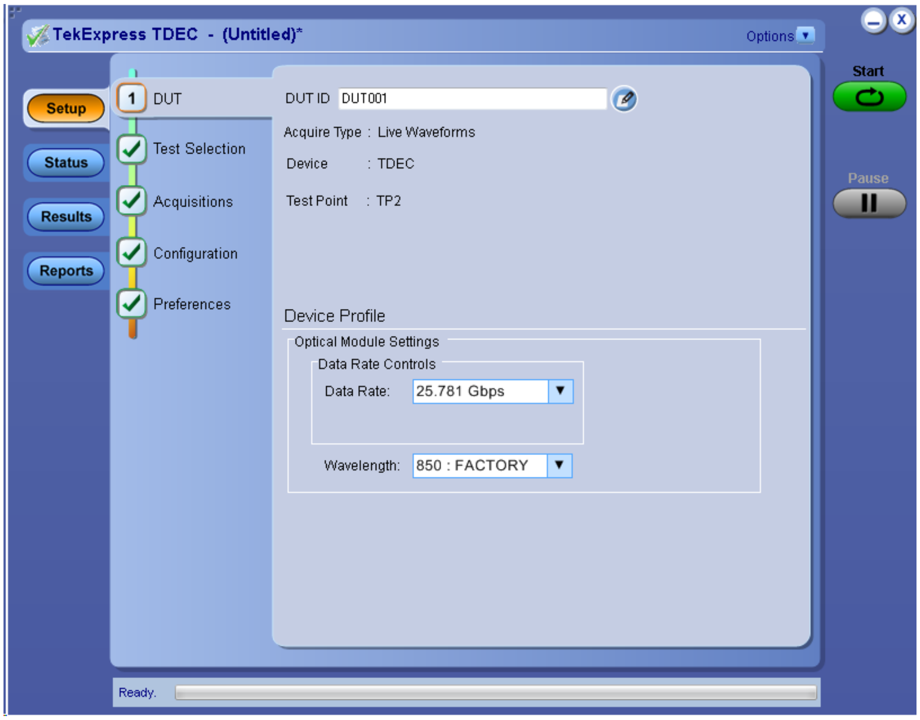
View test-related files

Application directories

Operating basics

Launch the application

To launch the TekExpress TDEC Solution, select **Application > TDEC** from the TekScope menu.



DUT ID DUT001

Acquire Type : Live Waveforms

Device : TDEC

Test Point : TP2

Data Rate: 25.781 Gbps

Wavelength: 850 : FACTORY

When you launch the application for the first time, the file `C:\Users\<username>\My Documents\My TekExpress \TDEC\Resources.xml` is mapped to drive `X:`. This file contains information about available network-connected instruments. The session files are stored in `X:\TDEC\`. If this file is not found, then the application runs Instrument Discovery Program to detect the connected instruments before launching TDEC Solution.

NOTE. *Map My TekExpress folder to drive X: and set My TekExpress folder permissions before running tests for the first time.*

If the application goes behind the oscilloscope application, click **Application > TDEC** to bring it to the front. To keep the TDEC application window on top, select **Keep On Top** from the TDEC *Options menu*.

See also

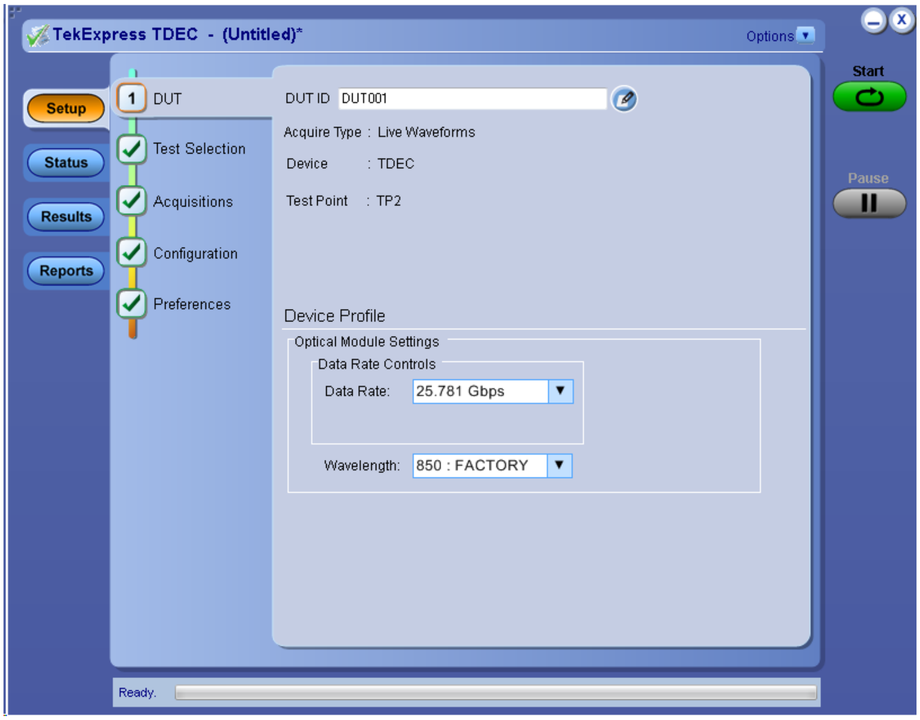
My TekExpress folder settings

Application controls

Application panel overview

Application panels overview

TekExpress TDEC Solution uses panels to group related configuration, test, and results settings. Click any 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.



DUT ID DUT001

Acquire Type : Live Waveforms

Device : TDEC

Test Point : TP2

Device Profile

Optical Module Settings

Data Rate Controls

Data Rate: 25.781 Gbps

Wavelength: 850 : FACTORY

Table 6: Application panels overview

Panel Name	Purpose
Setup panel	The Setup panel shows the test setup controls. Click the **Setup** button to open this panel. Use this panel to: ■ *Set DUT tab parameters* ■ *Select tests* ■ *Set acquisition tab parameters* ■ *Set configuration tab parameters* ■ Set test notification parameters in the *Preferences tab*
Status panel	View the progress and analysis status of the selected tests, and view test logs.
Results panel	View a summary of test results and select result viewing preferences.
Reports panel	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

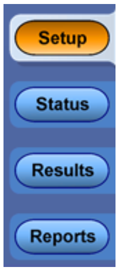
Application controls

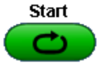
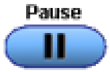
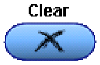
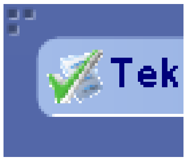
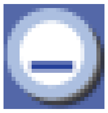
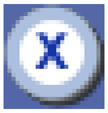
Global application controls

Application controls

This section describes the application controls.

Table 7: Application controls description

Item	Description
Options menu 	Menu to display global application controls.
Test Panel buttons 	Controls that open panels for configuring test settings and options.

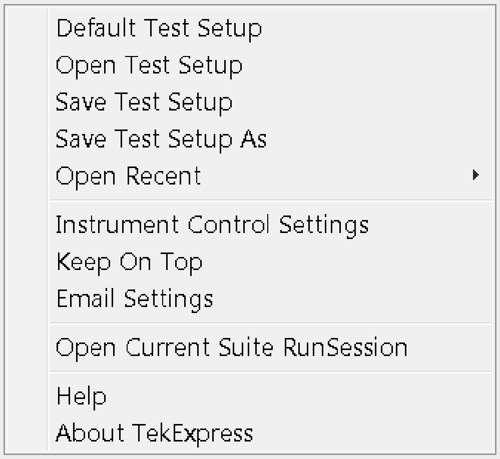
Item	Description
Start / Stop button  	Use the Start button to start the test run of the measurements in the selected order. If prior acquired measurements are not cleared, then new measurements are added to the existing set. The button toggles to the Stop mode while tests are running. Use the Stop button to abort the test.
Pause / Continue button  	Use the Pause button to temporarily pause the acquisition. When a test is paused, this button changes to “Continue.”
Clear button  	Use the Clear button to clear all existing measurement results. Adding or deleting a measurement, or changing a configuration parameter of an existing measurement, also clears measurements. This is to prevent the accumulation of measurement statistics or sets of statistics that are not coherent. This button is available only on *Results panel*.
Application window move icon  	Place the cursor over the three-dot pattern in the upper left corner of the application window. When the cursor changes to a hand, drag the window to the desired location.
Minimize icon 	Click to minimize the application.
Close icon 	Click to close the application.

See also. *Application panel overview*

Options menu overview

To access *Options menu*, click ▼ in the upper-right corner of the application. It has the following:

Options menu



Default Test Setup
Open Test Setup
Save Test Setup
Save Test Setup As
Open Recent ▸
Instrument Control Settings
Keep On Top
Email Settings
Open Current Suite RunSession
Help
About TekExpress

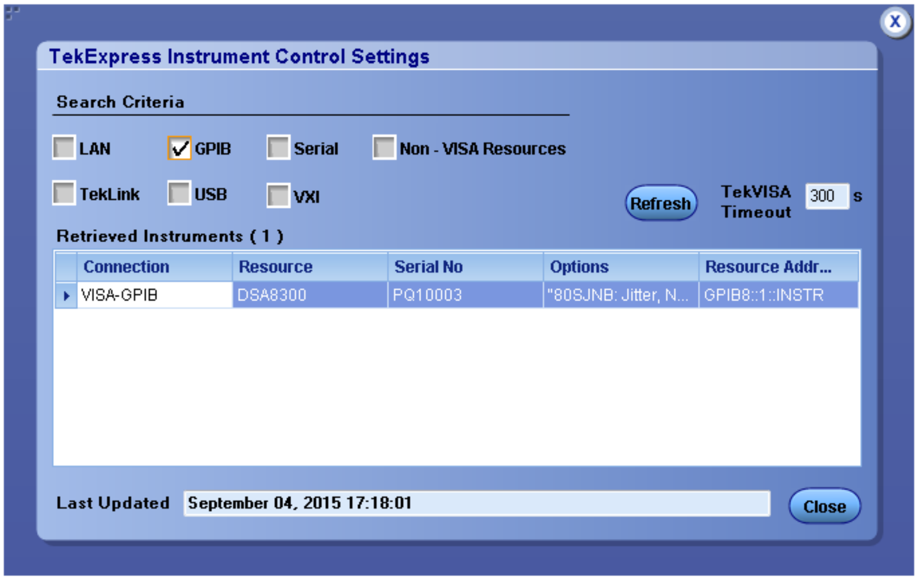
Menu	Function
Default Test Setup	Opens an untitled test setup with defaults selected **Data Rate**: 25,781 Gbps **Wavelength**: 850 nm **OMA Method**: OME-Eye **Trigger Source**: TekCRU **Phase Reference**: Checked **TDEC Filter**: None **TDEC Bandwidth**: 16.8 GHz Values displayed in **Wavelength**, **Filter**, and **Bandwidth** fields depend on the installed optical module.
Open Test Setup	Opens a saved test setup
Save Test Setup	Saves the current test setup
Save Test Setup As	Saves the current test setup with a different file name or file type
Open Recent	Displays the recently opened test setups to open
Instrument Control Settings	Detects, lists, and refreshes the connected instruments found on specified connections (LAN, GPIB, USB, and so on)
Keep On Top	Keeps the TekExpress TDEC application on top in the desktop
Email Settings	Use to configure email options for test run and results notifications
Open Current Suite RunSession	Allows the user to select the specific run setup, from the saved session
Help	Displays the TekExpress TDEC help
About TekExpress	■ Displays application details such as software name, version number, and copyright ■ Provides a link to the end-user license agreement ■ Provides a link to the Tektronix Web site

See also. *Application controls*

TekExpress instrument control settings

Use TekExpress Instrument Control Settings dialog box to search the instruments (resources) connected to the application. You can use the Search Criteria to search the connected instruments depending on the connection type. The details of the connected instrument is displayed in the Retrieved Instruments window.

You can access this dialog box from the **Options** menu.



TekExpress Instrument Control Settings

Search Criteria

□ LAN ☑ GPIB □ Serial □ Non - VISA Resources

□ TekLink □ USB □ VXI

TekVISA Timeout 300 s

Retrieved Instruments (1)

Connection	Resource	Serial No	Options	Resource Addr...
VISA-GPIB	DSA8300	PQ10003	"80SJNB: Jitter, N...	GPIB8::1::INSTR

Last Updated September 04, 2015 17:18:01

The connected instruments displayed here can be selected for use under Global Settings in the test configuration section.

***NOTE.** Under **Instrument Control Settings**, select GPIB Option (Default setting), when using TekExpress TDEC application.*

See also. *Options menu overview*

View connected instruments

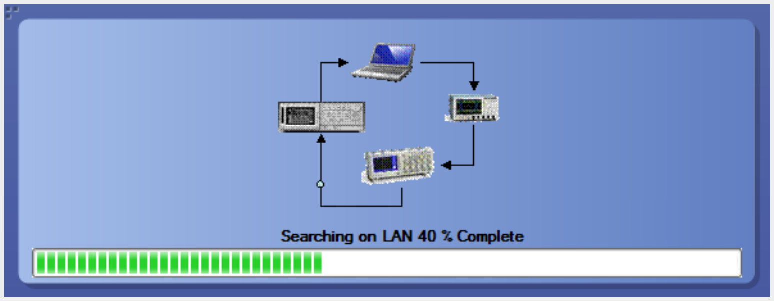
Use TekExpress Instrument Control Settings dialog box to search the instruments (resources) connected to the application. The application uses TekVISA to discover the connected instruments.

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

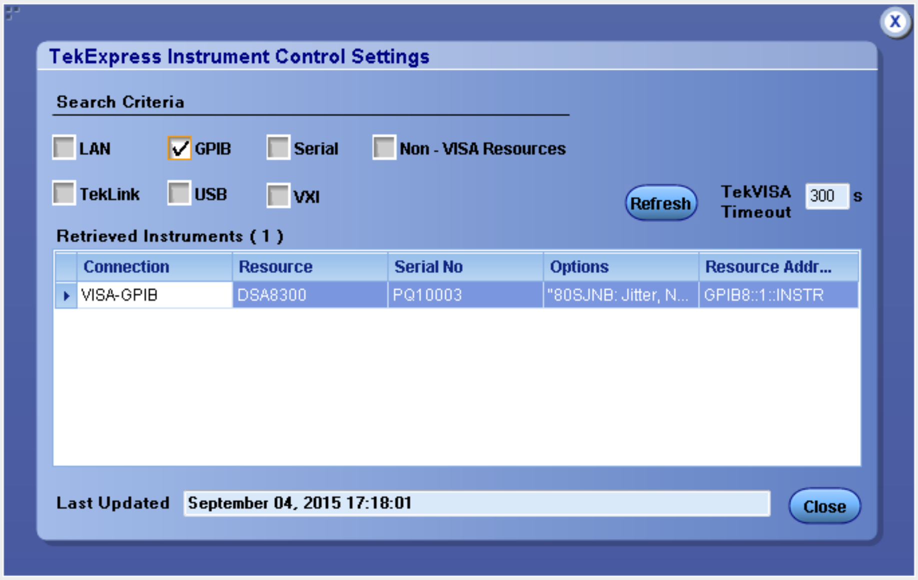
To refresh the list of connected instruments:

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 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.


Searching on LAN 40 % Complete

4. After searching, the dialog box lists the instrument-related details based on the search criteria. For example, For the Search Criteria as LAN and GPIB, the application displays all LAN and GPIB instruments connected to the application.


TekExpress Instrument Control Settings

Search Criteria

□ LAN ☑ GPIB □ Serial □ Non - VISA Resources

□ TekLink □ USB □ VXI

TekVISA Timeout 300 s

Retrieved Instruments (1)

Connection	Resource	Serial No	Options	Resource Addr...
VISA-GPIB	DSA8300	PQ10003	"80SJNB: Jitter, N...	GPIB8::1::INSTR

Last Updated September 04, 2015 17:18:01

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. *Equipment connection setup*

Configure email settings

Use the Email Settings utility to get notified by email when a measurement completes, or produces any error condition. Follow the steps to configure email settings:

1. Select **Options > Email Settings** to open the *Email Settings* dialog box.
2. (Required) For Recipient email Address(es), enter one or more recipient email addresses. 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 ID. For example: DSA8300_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:

   - 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.

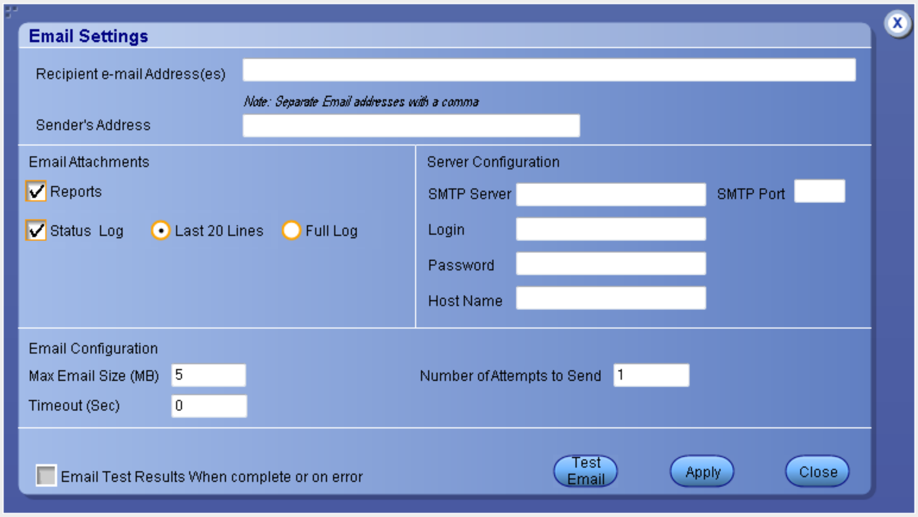
7. Select the **Email Test Results When complete or on error** check box. Use this check box to quickly enable or disable email notifications.

8. To test your email settings, click **Test Email**.

9. To apply your settings, click **Apply**.

10. Click **Close** when finished.

Email Settings



Email Settings

Recipient e-mail Address(es)

Note: Separate Email addresses with a comma

Sender's Address

Email Attachments

☑ Reports

☑ Status Log ● Last 20 Lines ○ Full Log

Server Configuration

SMTP Server

SMTP Port

Login

Password

Host Name

Email Configuration

Max Email Size (MB) 5

Timeout (Sec) 0

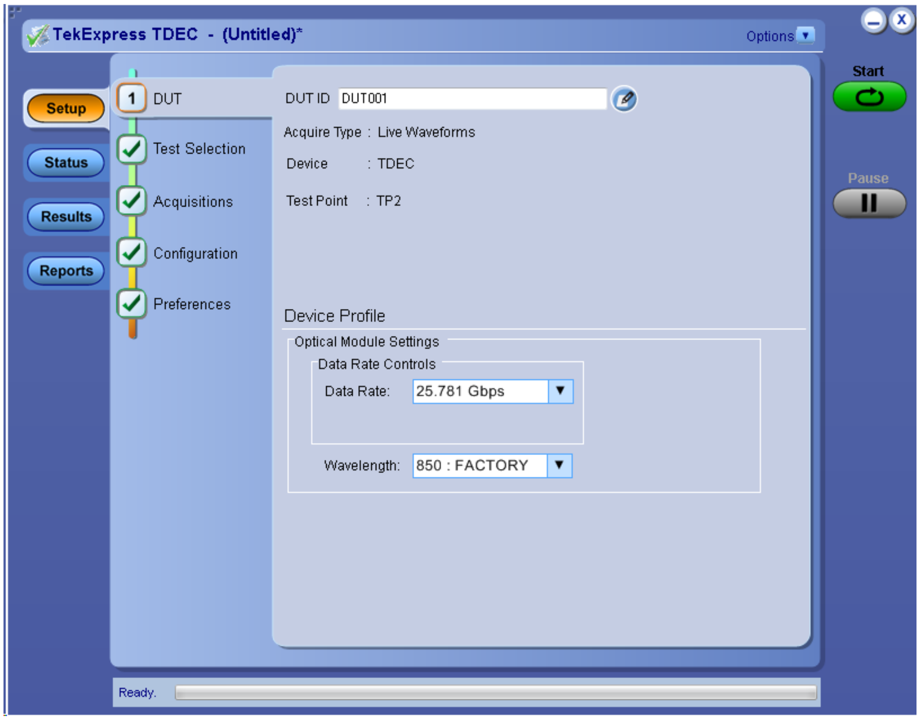
Number of Attempts to Send 1

□ Email Test Results When complete or on error

Setup panel

Setup panel overview

The Setup panel contains sequentially ordered tabs that help you guide through the test setup and execution process.



DUT ID DUT001

Acquire Type : Live Waveforms

Device : TDEC

Test Point : TP2

Device Profile

Optical Module Settings

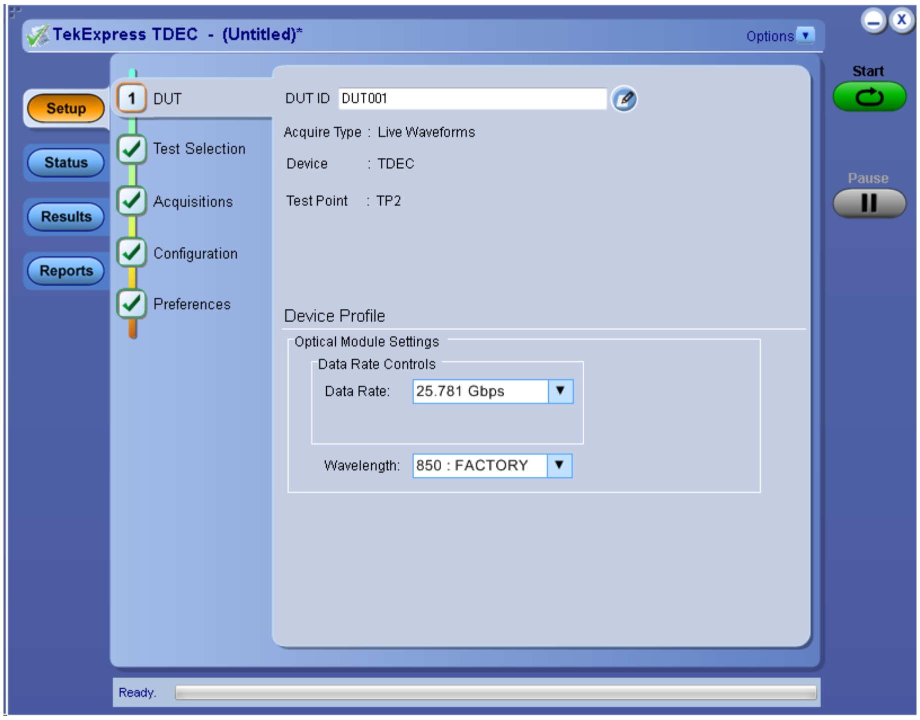
Data Rate Controls

Data Rate: 25.781 Gbps

Wavelength: 850 : FACTORY

Set DUT parameters

Use the DUT tab to select parameters for the device under test. This 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.



DUT ID DUT001

Acquire Type : Live Waveforms

Device : TDEC

Test Point : TP2

Device Profile

Optical Module Settings

Data Rate Controls

Data Rate: 25.781 Gbps

Wavelength: 850 : FACTORY

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

Table 8: DUT tab settings

<table>
<tr><th>Setting</th><th>Description</th></tr>
<tr><td>DUT ID</td><td>Adds an optional text label for the DUT to reports. The default value is DUT001. The maximum number of characters is 32.
You cannot use the following characters in an ID name: (.,..,...,\,/:?”&lt;&gt;|*)</td></tr>
<tr><td>✎ Comments icon (to the right of the DUT ID field)</td><td>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 <i>Select report options</i> for details.</td></tr>
<tr><td>Acquire Type</td><td>Live waveforms.
This application performs analysis on live waveforms only. This field is not editable.</td></tr>
<tr><td>Device</td><td>TDEC measurements are taken on Device only. This field is not editable.</td></tr>
<tr><td>Test Point</td><td>TP2. This field is not editable.</td></tr>
<tr><td colspan="2"><b>Device Profile</b></td></tr>
<tr><td><b>Optical Module Settings</b></td><td></td></tr>
<tr><td>Data Rate Controls - Data Rate</td><td>Set the data rate to be tested.</td></tr>
<tr><td>Wavelength</td><td>Select the wavelength from the drop-down list.</td></tr>
</table>

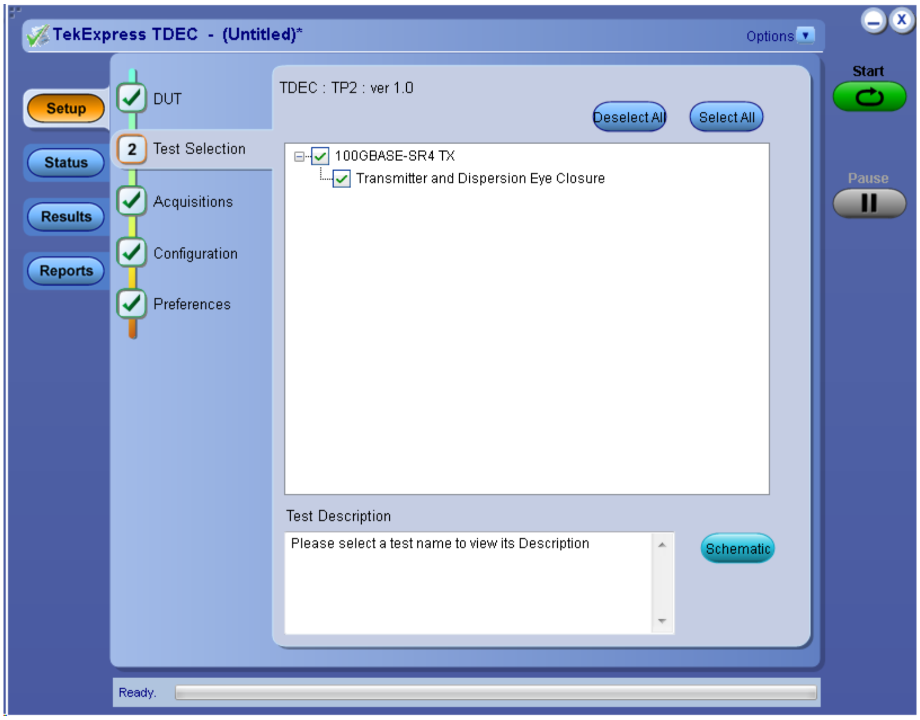
***NOTE.** The installed optical module sets the available wavelength, filter, and bandwidth selections.*

You can set either the filter or bandwidth value, but not both.

See also. *Select a test*

Select tests

Use the Test Selection tab to select the TekExpress TDEC tests.



☑ 100GBASE-SR4 TX
☑ Transmitter and Dispersion Eye Closure

Please select a test name to view its Description

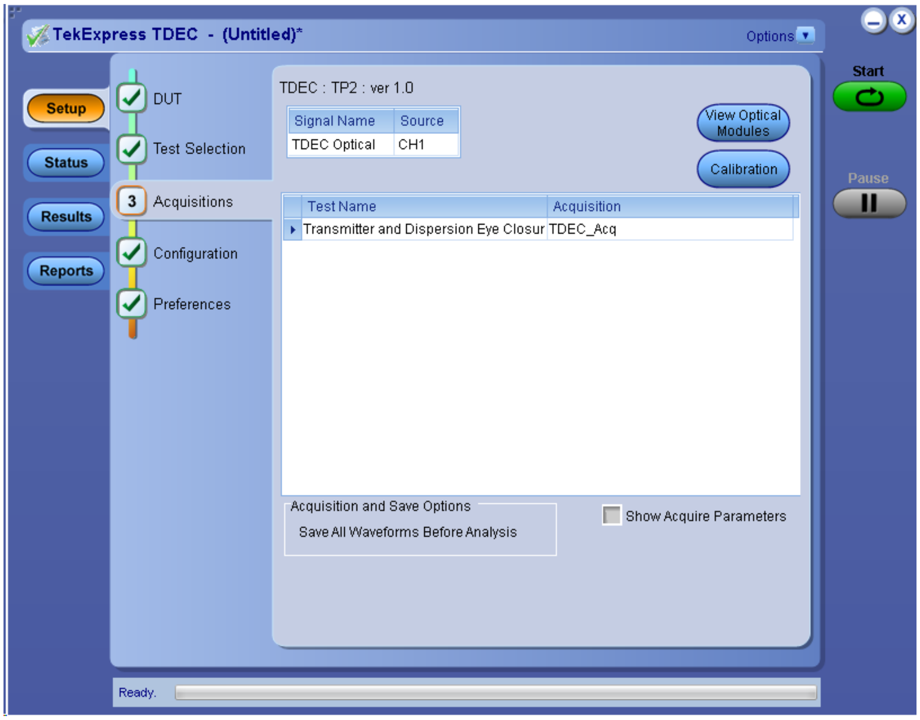
Table 9: Test Selection tab settings

Setting	Description
Tests	Click on a test to select or unselect. Highlight a test to show details in the Test Description pane.
Test Description	Shows a brief description of the highlighted test in the Test field.

See also. Set acquisition tab parameters

Set acquisition tab parameters

Use the Acquisitions tab to view test acquisition parameters. The contents displayed on this tab depends on the DUT type and the tests selected.



TDEC : TP2 : ver 1.0

Signal Name	Source
TDEC Optical	CH1

Test Name	Acquisition
Transmitter and Dispersion Eye Closur	TDEC_Acq

Acquisition and Save Options

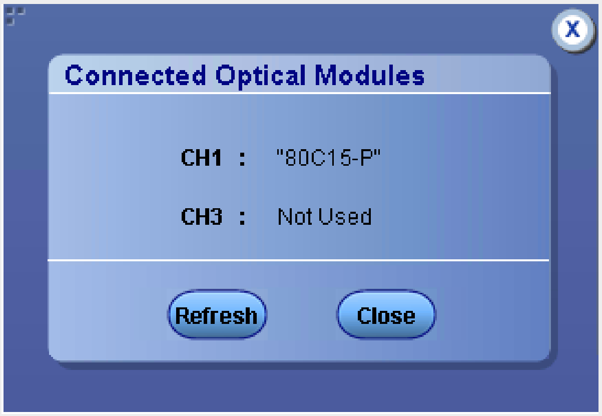
Save All Waveforms Before Analysis

□ Show Acquire Parameters

***NOTE.** TDEC acquires all waveforms needed by each test group before performing analysis.*

Table 10: Acquisitions tab settings

Setting	Description
View Optical Modules button	Click to view the detected optical modules that are installed in the instrument.
Calibration button	Click to view the status of Oscilloscope calibration, External attenuation calibration, and Instrument noise calibration. Update these parameters by clicking the associated Refresh or Measure button. See *Pre-measurement calibration* for details.
Show Acquire Parameters	Select to view the acquisition parameters.



Connected Optical Modules

CH1 : "80C15-P"

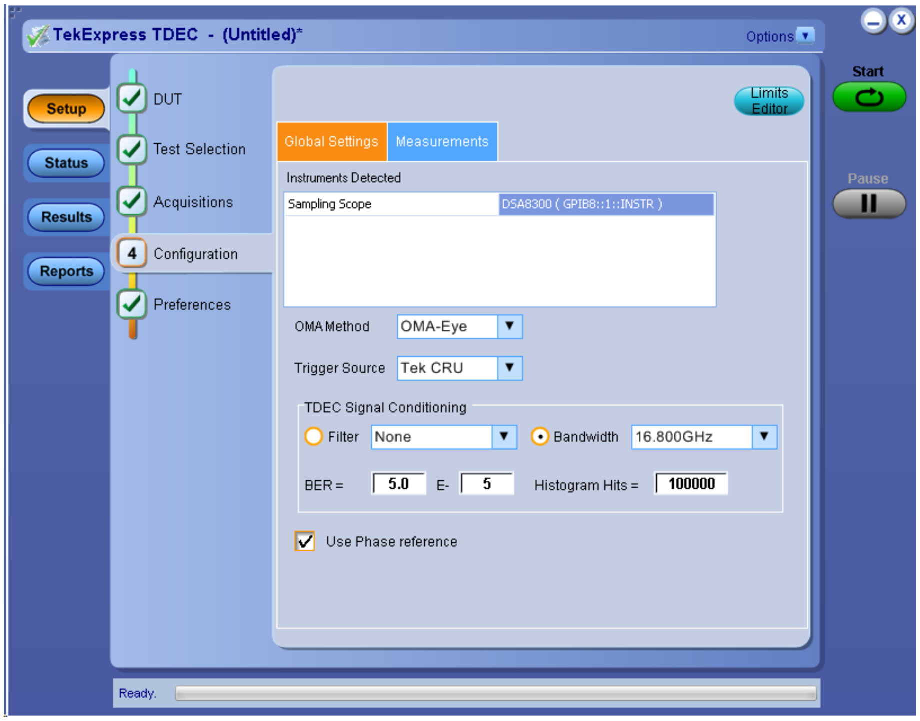
CH3 : Not Used

TekExpress TDEC saves all acquisition waveforms to files by default. Waveforms are saved in a unique folder for each session (a session starts when you click the Start button). The folder path is `X:\TDEC\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.

Saving a session moves the session file contents from the Untitled Session folder to the specified folder name, and changes the session name to the specified name.

Set configuration tab parameters

Use the Configuration tab to view the instruments detected (Global Settings).



Instruments Detected

Sampling Scope	DSA8300 (GPIB8::1::INSTR)

OMA Method: OMA-Eye

Trigger Source: Tek CRU

TDEC Signal Conditioning

○ Filter: None ● Bandwidth: 16.800GHz

BER = 5.0 E- 5 Histogram Hits = 100000

☑ Use Phase reference

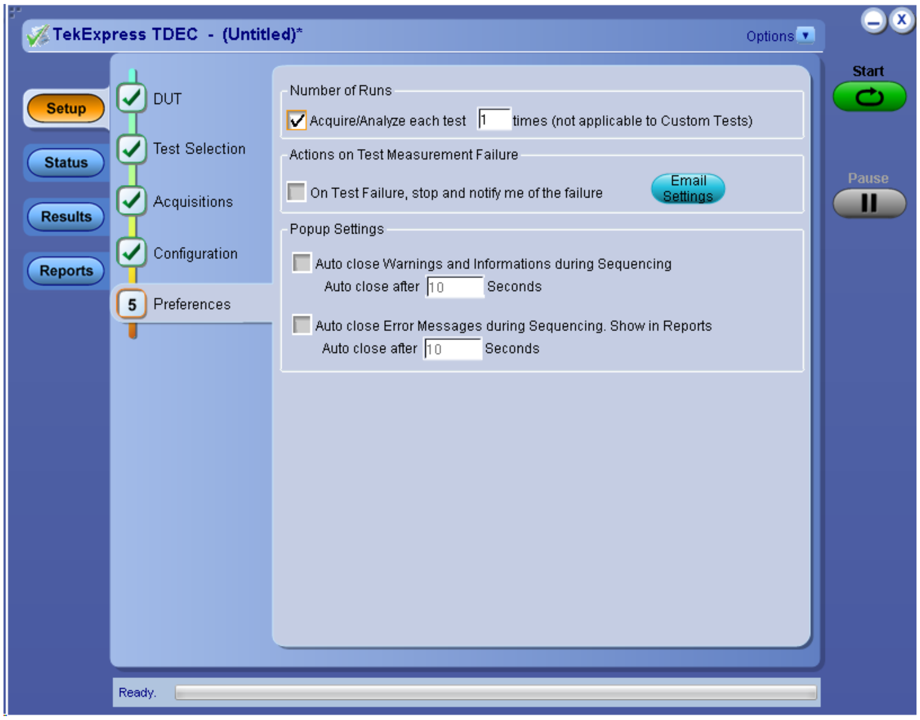
Figure 1: Configuration tab: Global Settings

Table 11: Configuration tab settings

Setting		Description
Global Settings		
Instruments Detected		Displays the instruments connected to this application. Click on the instrument name to open a list of available (detected) instruments. Select **Options > Instrument Control Settings** and click Refresh to update the instrument list. ***NOTE.*** *Verify that the* ***GPIB*** *search criteria (default setting) in the Instrument Control Settings is selected when using TekExpress TDEC application.*
OMA Method		Select the OMA method from the drop-down list. By default, it is OMA-Eye.
Trigger Source		Select the trigger source from the drop-down list. By default, it is Tek CRU.
TDEC Signal Conditioning	Filter	Select the filter from the drop-down list.
	Bandwidth	Select the bandwidth from the drop-down list. If 80C15 is present in main frame, and if the module has TDEC filter (16.8 GHz), then the default will be 16.8 GHz. If 80C15 is not present in main frame then default value will be None
	BER	Set the BER value. By default it is 5.0E-5.
	Histogram Hits	Set the Histogram Hits. By default it is 100000.
Use Phase reference		Select to perform phase reference characterization.
Measurements		
Transmitter and Dispersion Eye Closure		No configuration available for this measurement.

Set preferences tab parameters

Use the Preferences tab to set the application action on completion of a measurement.



Number of Runs

☑ Acquire/Analyze each test 1 times (not applicable to Custom Tests)

Actions on Test Measurement Failure

☐ On Test Failure, stop and notify me of the failure

Popup Settings

☐ Auto close Warnings and Informations during Sequencing

Auto close after 10 Seconds

☐ Auto close Error Messages during Sequencing. Show in Reports

Auto close after 10 Seconds

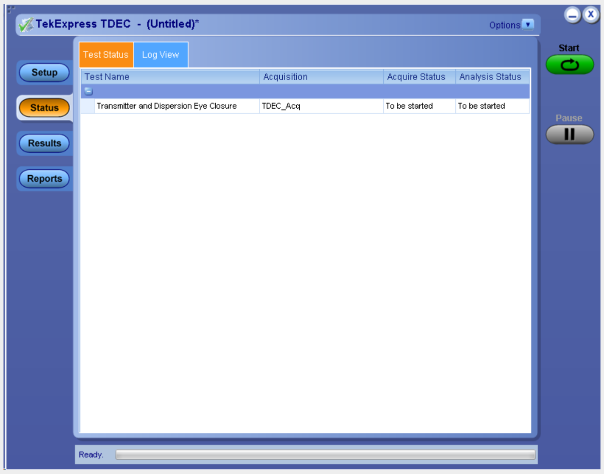
Table 12: Preferences tab settings

Setting	Description
Number of Runs	
Acquire/Analyze each test <no> times (not applicable to Custom Tests)	Select to repeat the test run by setting the number of times. By default, it is selected with 1 run.
Actions on Test Measurement Failure	
On Test Failure, stop and notify me of the failure	Select to stop the test run on Test Failure, and to get notified via email. By default, it is unselected. Click Email Settings to configure.
Popup Settings	
Auto close Warnings and Informations during Sequencing Auto close after <no> Seconds	Select to auto close warnings/informations during sequencing. Set the Auto close time. By default it is unselected.
Auto close Error Messages during Sequencing. Show in Reports Auto close after <no> Seconds	Select to auto close Error Messages during Sequencing. Set the Auto close time. By default it is unselected.

Status panel overview

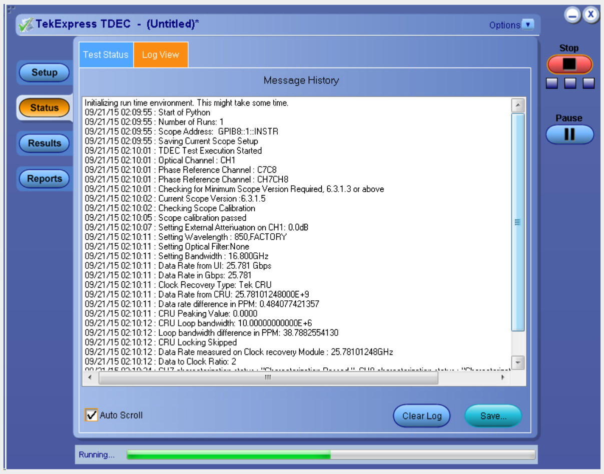
The Status button accesses the Test Status and Log View tabs, which provide status on test acquisition and analysis (Test Status tab) and a listing of test tasks performed (Log View tab). The application opens the Test Status tab when you start a test run. You can select the Test Status or the Log View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

Test status view



Test Name	Acquisition	Acquire Status	Analysis Status
Transmitter and Dispersion Eye Closure	TDEC_Acq	To be started	To be started

Log view



Message History

```
Initializing run time environment. This might take some time.
09/21/15 02:09:55 : Start of Python
09/21/15 02:09:55 : Number of Runs: 1
09/21/15 02:09:55 : Scope Address:  GPIB8::1::INSTR
09/21/15 02:09:55 : Saving Current Scope Setup
09/21/15 02:10:01 : TDEC Test Execution Started
09/21/15 02:10:01 : Optical Channel : CH1
09/21/15 02:10:01 : Phase Reference Channel : C7C8
09/21/15 02:10:01 : Phase Reference Channel : CH7CH8
09/21/15 02:10:01 : Checking for Minimum Scope Version Required, 6.3.1.3 or above
09/21/15 02:10:02 : Current Scope Version :6.3.1.5
09/21/15 02:10:02 : Checking Scope Calibration
09/21/15 02:10:05 : Scope calibration passed
09/21/15 02:10:07 : Setting External Attenuation on CH1: 0.0dB
09/21/15 02:10:11 : Setting Wavelength : 850,FACTORY
09/21/15 02:10:11 : Setting Optical Filter:None
09/21/15 02:10:11 : Setting Bandwidth : 16.800GHz
09/21/15 02:10:11 : Data Rate from UI: 25.781 Gbps
09/21/15 02:10:11 : Data Rate in Gbps: 25.781
09/21/15 02:10:11 : Clock Recovery Type: Tek CRU
09/21/15 02:10:11 : Data Rate from CRU: 25.78101248000E+9
09/21/15 02:10:11 : Data rate difference in PPM: 0.484077421357
09/21/15 02:10:11 : CRU Peaking Value: 0.0000
09/21/15 02:10:12 : CRU Loop bandwidth: 10.00000000000E+6
09/21/15 02:10:12 : Loop bandwidth difference in PPM: 38.7882554130
09/21/15 02:10:12 : CRU Locking Skipped
09/21/15 02:10:12 : Data Rate measured on Clock recovery Module : 25.78101248GHz
09/21/15 02:10:12 : Data to Clock Ratio: 2
```

Table 13: Status panel Log View controls

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

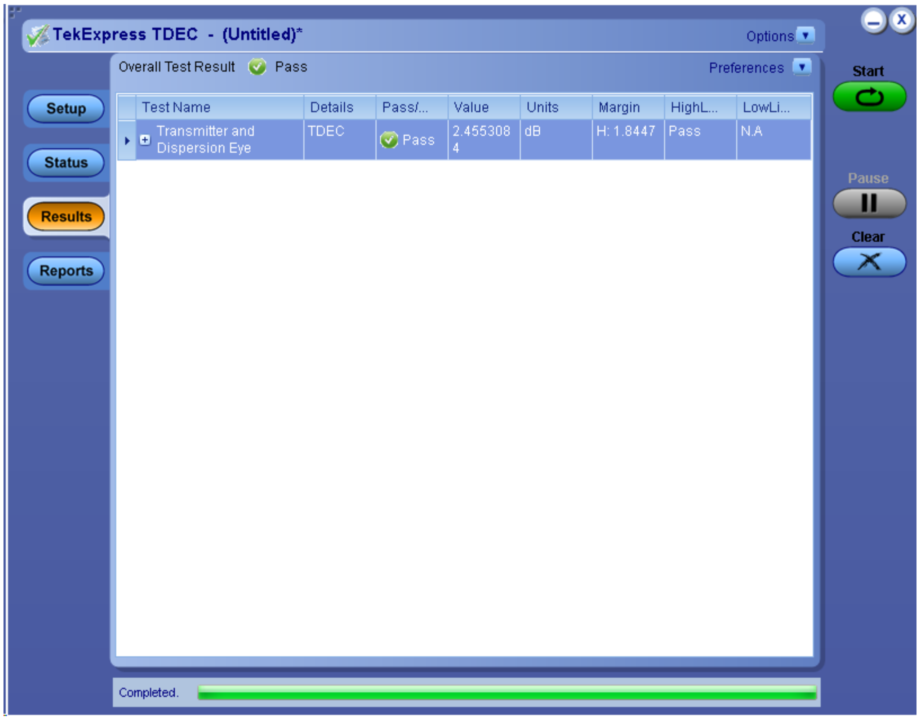
See also

Application panel overview

Results panel

Results panel overview

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



Overall Test Result ✓ Pass

Test Name	Details	Pass/...	Value	Units	Margin	HighL...	LowLi...
Transmitter and Dispersion Eye	TDEC	✓ Pass	2.4553084	dB	H: 1.8447	Pass	N.A

Completed.

See also. *View a report*

Application panels overview

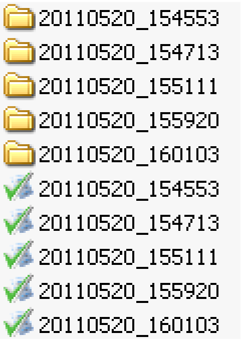
View test-related files

Files related to tests are stored in `My TekExpress\TDEC\`. Each test setup in this folder has both 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:



20110520_154553
20110520_154713
20110520_155111
20110520_155920
20110520_160103
20110520_154553
20110520_154713
20110520_155111
20110520_155920
20110520_160103

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\TDEC\`. 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 TDEC application.

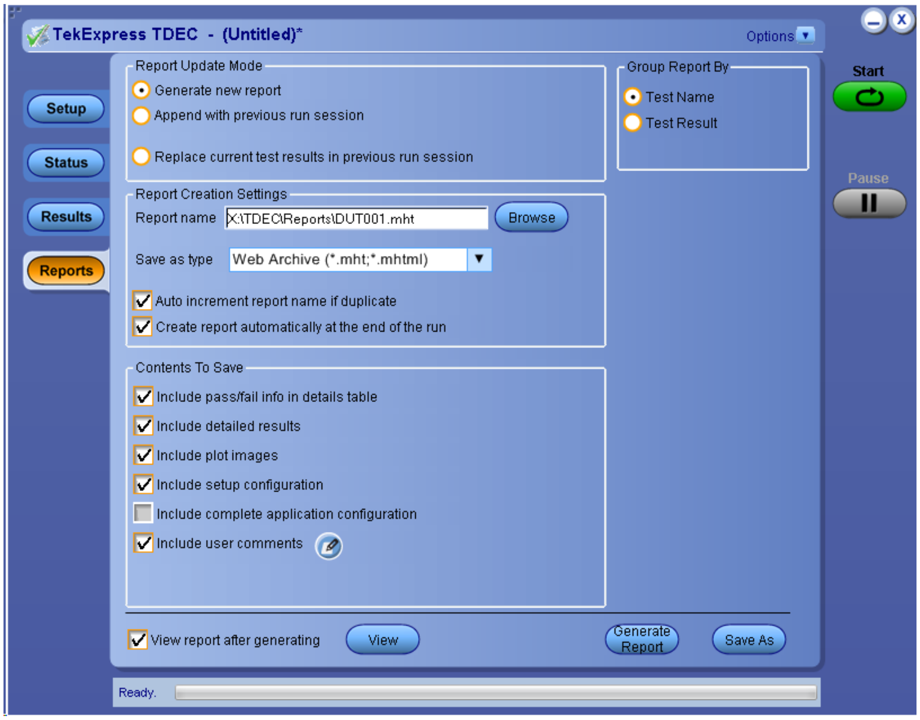
See also. *File name extensions*

My TekExpress folder settings

Reports panel

Reports panel overview

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



TekExpress TDEC - (Untitled)*

Report Update Mode
● Generate new report
○ Append with previous run session
○ Replace current test results in previous run session

Group Report By
● Test Name
○ Test Result

Report Creation Settings
Report name X:\TDEC\Reports\DUT001.mht
Save as type Web Archive (*.mht;*.mhtml)
☑ Auto increment report name if duplicate
☑ Create report automatically at the end of the run

Contents To Save
☑ Include pass/fail info in details table
☑ Include detailed results
☑ Include plot images
☑ Include setup configuration
☐ Include complete application configuration
☑ Include user comments

☑ View report after generating

Ready.

For information on setting up reports, see *Select report options*. For information on viewing reports, see *View a report*.

See also. *Applications panel overview*

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 Update Mode	
Generate new report	Creates a new report. The report can be in either .mht or .pdf file formats.
Append with previous run session	Appends the latest test results to the end of the current test results report.
Replace current test in previous run session	Replaces the previous test results with the latest test results. Results from newly added tests are appended to the end of the report.
Report Creation Settings	
Report name	Displays the name and location from which to open a TDEC report. The default location is at *\My TekExpress\TDEC\Untitled Session*. The report file in this folder gets overwritten each time you run a test unless you specify a unique name or select to auto increment the report name. Change the report name or location. Do one of the following: ■ In the Report Path field, type over the current folder path and name. ■ Double-click in the Report Path field and then make selections from the popup keyboard and click the **Enter** button. 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\TDEC\DUT001.mht. ***NOTE.*** *You cannot set the file location using the Browse button.* Open an existing report. Click **Browse**, locate and select the report file and then click **View** at the bottom of the panel.

Setting	Description
Save as type	Saves a report in the specified file type, selected from the drop-down list. ***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.
Create report automatically at the end of the run	Creates report at the end of the run.
Contents To Save	
Include pass/fail info in details table	Includes pass/fail info in the details table of the report.
Include detailed results	Includes detailed results in the report.
Include plot images	Includes plot images in the report.
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, the oscilloscope firmware version, and software versions for applications used in the measurements.
Include complete application configuration	Includes complete application configuration in the report.
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.
Group Report By	
Test Name	Select to group the tests in the report by test name.
Test Result	Select to group the tests in the report by test results
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

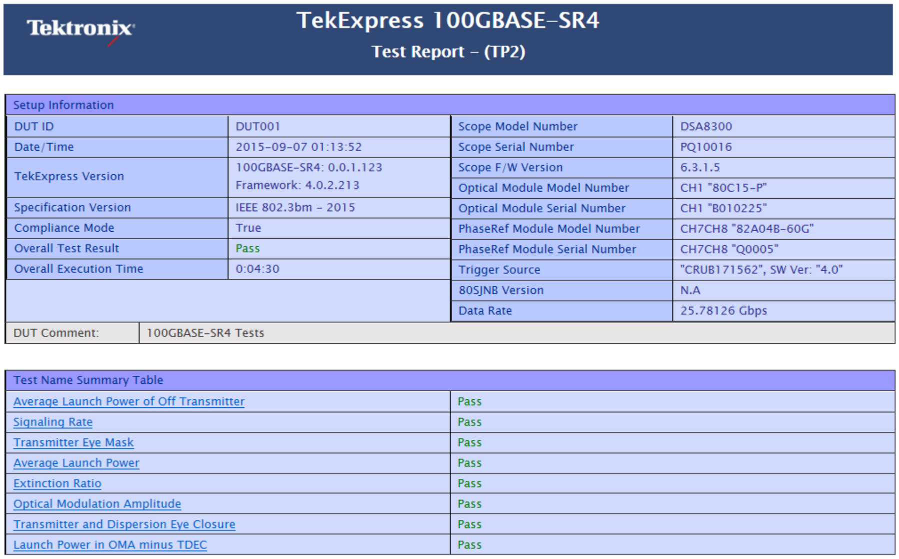
The application automatically generates a report when test analysis is completed and displays the report in your default Web browser (unless you cleared the **View Report After Generating** check box in the Reports panel before running the test). If you cleared this check box, or to view a different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

Report contents

A report shows detailed results and plots, as set in the Reports panel.



Tektronix

TekExpress 100GBASE-SR4

Test Report – (TP2)

Setup Information			
DUT ID	DUT001	Scope Model Number	DSA8300
Date/Time	2015-09-07 01:13:52	Scope Serial Number	PQ10016
TekExpress Version	100GBASE-SR4: 0.0.1.123 Framework: 4.0.2.213	Scope F/W Version	6.3.1.5
		Optical Module Model Number	CH1 "80C15-P"
Specification Version	IEEE 802.3bm – 2015	Optical Module Serial Number	CH1 "B010225"
Compliance Mode	True	PhaseRef Module Model Number	CH7CH8 "82A04B-60G"
Overall Test Result	Pass	PhaseRef Module Serial Number	CH7CH8 "Q0005"
Overall Execution Time	0:04:30	Trigger Source	"CRUB171562", SW Ver: "4.0"
		80SJNB Version	N.A
		Data Rate	25.78126 Gbps
DUT Comment:	100GBASE-SR4 Tests		

Test Name Summary Table	
Average Launch Power of Off Transmitter	Pass
Signaling Rate	Pass
Transmitter Eye Mask	Pass
Average Launch Power	Pass
Extinction Ratio	Pass
Optical Modulation Amplitude	Pass
Transmitter and Dispersion Eye Closure	Pass
Launch Power in OMA minus TDEC	Pass

Setup configuration information

The summary box at the beginning of the report lists setup configuration information. This information includes the oscilloscope model and serial number, optical module model and serial number, and software version numbers of all associated applications.

To exclude this information from a report, clear the **Include Setup Configuration** check box in the Reports panel before running the test.

User comments

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

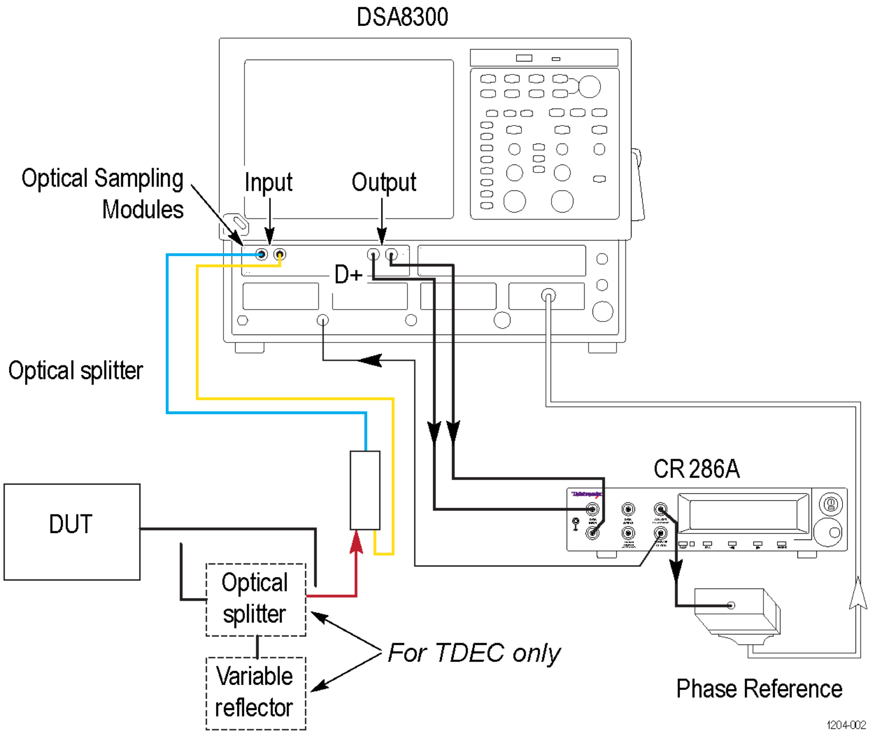
See also. *Results panel overview*

View test-related files

Pre-measurement calibration

Equipment setup

Click **Setup** > **Test Selection** > **Schematic** to view the equipment setup diagram(s).



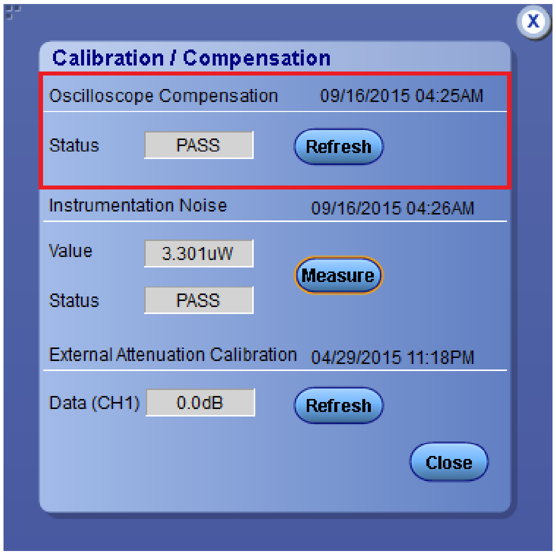
See also

Minimum system requirements

Oscilloscope compensation

Use the following procedure to check oscilloscope calibration status:

1. Select **TekExpress TDEC** > **Setup** > **Acquisition panel** > **Calibration** to open the calibration dialog box.
2. Click **Refresh** (in the Oscilloscope Calibration area).



Calibration / Compensation

Oscilloscope Compensation 09/16/2015 04:25AM

Status PASS

Instrumentation Noise 09/16/2015 04:26AM

Value 3.301uW

Status PASS

External Attenuation Calibration 04/29/2015 11:18PM

Data (CH1) 0.0dB

__NOTE.__ It is recommended to perform Scope Compensation in addition after 20 minutes of warm up. Scope compensation can be accessed from the Oscilloscope main menu, __Utilities > Instrument Compensation__. Click Help in the compensation window for further details.

Instrument noise

The following procedure is used by the TDEC application to measure Instrument noise calibration:

1. Disconnect all signals connected to the sampling oscilloscope
2. Select **Setup > Vert > waveform C1** to **On**
3. Define **MATH1** as Ch1, and switch on MATH1
4. Set the Trigger Source to **Free Run**
5. Select measurement **Setup > Meas > Meas 1 > Pulse Amplitude: AC RMS**
6. Set **Setup > Meas > Source: MATH1**
7. Set WaveformdB source as MATH1
8. Enable and switch on the display of WaveformdB
9. Query the result of measurement1 (AC RMS)

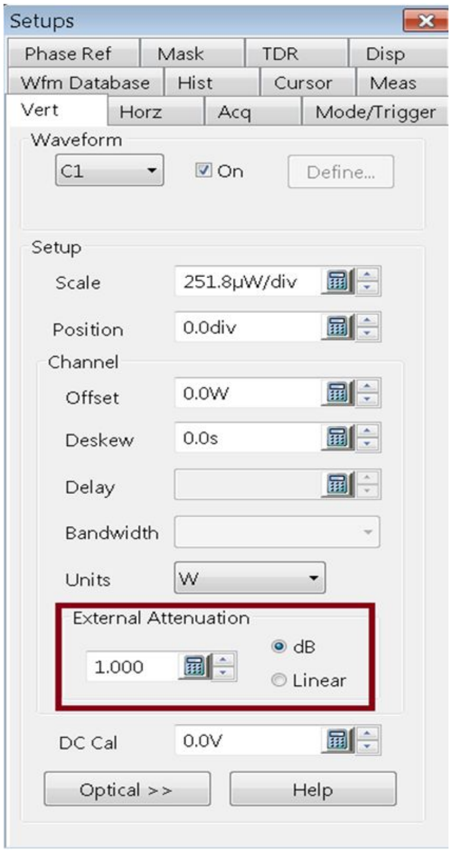
__NOTE.__ Measured noise limit is a function of optical settings (Bandwidth and Filter).

If the noise level measurement is not within the limits, perform an oscilloscope compensation and then perform the instrument noise measurement again. If the measured noise level is still outside of the above limits, please *contact Tektronix customer support*.

External attenuation calibration

Follow the steps to set the external attenuation:

1. In DSA8300, set the optical source as **Ch1**
2. Enter the External Attenuation value for the scope as shown in the following image.



Waveform: C1 ☑ On

Setup

Scale: 251.8µW/div

Position: 0.0div

Channel

Offset: 0.0W

Deskew: 0.0s

Delay:

Bandwidth:

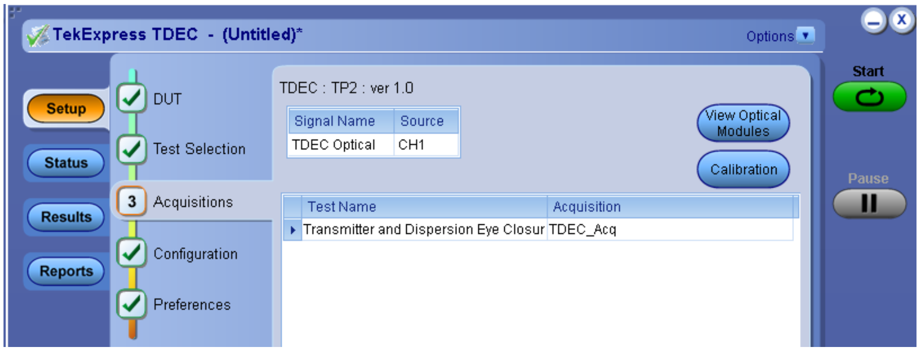
Units: W

External Attenuation: 1.000 ● dB ○ Linear

DC Cal: 0.0V

3. Select **Ch1** from the **TekExpress TDEC** > **Setup** > **DUT** > **Source**

4. Click **TekExpress TDEC** > **Setup** > **Acquisition** > **Calibration** to open the calibration dialog box

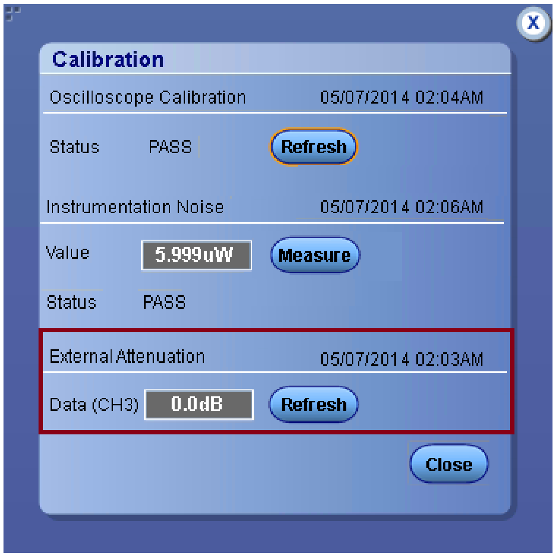


TDEC : TP2 : ver 1.0

Signal Name	Source
TDEC Optical	CH1

Test Name	Acquisition
Transmitter and Dispersion Eye Closur	TDEC_Acq

5. Click Refresh (in External Attenuation area) and check the value



Oscilloscope Calibration 05/07/2014 02:04AM

Status PASS

Instrumentation Noise 05/07/2014 02:06AM

Value 5.999uW

Status PASS

External Attenuation 05/07/2014 02:03AM

Data (CH3) 0.0dB

6. Repeat steps GUID-8933E4E4-F319-4FB7-8B78-E5AFD0BB208F#GUID-8933E4E4-F319-4FB7-8B78-E5AFD0BB208F/ID_1 to GUID-8933E4E4-F319-4FB7-8B78-E5AFD0BB208F#GUID-8933E4E4-F319-4FB7-8B78-E5AFD0BB208F/ID_5 by selecting **Ch3** and check the value

Running tests

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

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 **Alt + Tab** key combination. To keep the TekExpress TDEC application on top, select **Keep On Top** from the TekExpress Options menu.

The application displays report when the tests execution is complete.

NOTE. *The TDEC software stores the DSA8300 oscilloscope setup before test execution and restores the oscilloscope to the same state after test execution.*

Prerun checklist

1. Map *My TekExpress folder to drive X:* and set *My TekExpress folder permissions* before running tests for the first time.
2. Make sure that the instruments are warmed up (approximately 20 minutes) and stabilized.
3. Perform compensation: In the oscilloscope main menu, select **Utilities > Instrument Compensation**. Click **Help** in the compensation window for steps to perform instrument compensation.

Saving and recalling test setup files

Test setup files overview

Saved test setup information (such as the selected oscilloscope, general parameters, acquisition parameters, measurement limits, waveforms (if applicable), and other configuration settings) is saved under the setup name at **X:\TDEC**.

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

Open (load) a saved test setup

Save a test setup

You can save a test setup before or after running a test. You can create a test setup from *already opened test setup*, or using *default test setup*. When you select the default test setup, the parameters are set to the application's default value.

Select **Options > Save Test Setup** to save the opened setup.

Select **Options > Save Test Setup As** to save the setup with different name.

Open (load) a saved test setup

To Open (load) a saved test setup, do the following:

1. Select **Options > Open Test Setup**.
2. Select the setup from the list and click **Open**. Setup files are located at **X:\TDEC\**.

See also

About test setups

Create a test setup using an existing one

Create a test setup from default settings

Create a test setup from default settings

To create a test setup using default settings, follow the steps:

1. Select **Options > Default Test Setup**. For default test setup, the parameters are set to the application's default value.
2. Click application *Setup* and set the parameters
3. Click application *Reports* and set the report options
4. Optional: Click **Start** to run the test and verify that it runs correctly and captures the specified test information and reports. If it does not, then edit the parameters and repeat this step until the test runs to your satisfaction
5. Select **Options > Save Test Setup**. Enter the file name and click Save. The application saves the file to X:\TDEC \<*session_name*>

Create a test setup using an existing one

To create a test setup using an existing one, follow the steps:

1. Select **Options > Open Test Setup**
2. Select a setup from the list and then click **Open**
3. Click application *Setup* and modify the parameters
4. Click application *Reports* and modify the report options
5. Select **Options > Save Test Setup As**
6. Enter test setup name, and click **Save**

TDEC compliance measurements

Transmitter and Dispersion Eye Closure

This test verifies that the Transmitter and Dispersion Eye Closure (TDEC) of the Device Under Test (DUT) is within the conformance limit, as specified in Table 95-6 of 802.3bm - IEEE Standard for Ethernet

Required test equipment

Click here to check the Instruments and accessories required.

Connect the equipment as shown in the *equipment setup diagram*.

Measurement algorithm

This measurement has two steps:

1. Pre-processing: Setting up an eye, histogram boxes and exporting histogram data
2. Post-processing: Sigma and/or TDEC computation

Pre-processing: Setting up an eye, histogram boxes and exporting histogram data

1. Switch on the channel which is connected to 100GBASE-SR4 optical signal
2. Configure optical settings of the channel – wavelength, filter and bandwidth
3. Set the bit rate in horizontal sub system to user configured bit rate in GUI
4. Run **AutoSet** to set optimal vertical, horizontal settings. It also provides an eye occupying six vertical divisions and one bit (two eye crossings) displayed over about 7.5 horizontal divisions
5. Check On and Display check boxes of “WfmDB1”. Set source of WfmDB1 as the optical source selected by user
6. If phase reference module is present, dial-in the date rate and perform phase reference characterization
7. Run **NRZ Timing - Bit Time** measurement with following settings:
    a. Source as optical source selected by user
    b. Check **Use Wfm Database**
    c. Signal Type as **NRZ**
8. Run **NRZ Timing - Crossing Time** measurement with the following settings:
    a. Source as optical source selected by user
    b. Check **Use Wfm Database**
    c. Signal Type as **NRZ**
9. Run **NRZ Amplitude - Max** measurement with the following settings:
    a. Source as optical source selected by user
    b. Check **Use Wfm Database**
    c. Signal Type as **NRZ**

10. Run **NRZ Amplitude - Min** measurement with the following settings:
    a. Source as optical source selected by user
    b. Check **Use Wfm Database**
    c. Signal Type as **NRZ**
11. In Histogram setup,
    a. Select source of histogram as the optical source selected by the user
    b. Enable **Use Wfm Database**. Switch on **Histogram** of Linear type in display options
    c. Enable vertical histogram with absolute co-ordinates as:
       - Top = Result of Max measurement + (vertical scale / 2)
       - Bottom = Result of Min measurement – (vertical scale / 2)
       - Left = Result of Cross Time Measurement
       - Right = (Result of Cross Time measurement + Result of bit time measurement)
12. Mean of the histogram is taken as Average Launch Power in Watts
13. Determine the histogram box co-ordinates for early top histogram:
    - Early top histogram Left = crossing time + bit time * (Early histogram box position – (histogram width/2))
    - Early top histogram Right = crossing time + bit time * (Early histogram box position + (histogram width/2))
    - Early top histogram Base = AOP
    - Early top histogram Top = Maximum amplitude value + (vertical scale/2)
    - Set histogram configuration same as in Step GUID-243F9146-086A-4634-9D41-88ADF91D0575#GUID-243F9146-086A-4634-9D41-88ADF91D0575/STEP_11A and GUID-243F9146-086A-4634-9D41-88ADF91D0575#GUID-243F9146-086A-4634-9D41-88ADF91D0575/STEP_11B
14. Set acquisition configuration:
    - Stop After Mode as **Condition**
    - Stop After Condition as **HISTOGRAM HITS**
    - Stop After Count as number of histogram hits configured by user in GUI
    - Clear the screen and switch on Acquire state. Once acquisition is complete, export the early top histogram data to a text file.
15. Determine the histogram box co-ordinates for early bottom histogram:
    - Early bottom histogram Left = crossing time + bit time * (Early histogram box position – (histogram width/2))
    - Early bottom histogram Right = crossing time + bit time * (Early histogram box position + (histogram width/2))
    - Early bottom histogram Base = AOP
    - Early bottom histogram Top = Maximum amplitude value + (vertical scale/2)

    Do not re-run the acquisition. Export the early bottom histogram data to a text file.
16. Determine the histogram box co-ordinates for late top histogram:
    - Late top histogram Left = crossing time + bit time * (Late histogram box position – (histogram width/2))
    - Late top histogram Right = crossing time + bit time * (Late histogram box position + (histogram width/2))

- Late top histogram Base = AOP
- Late top histogram Top = Maximum amplitude value + (vertical scale/2)

Do not re-run the acquisition. Export the Late bottom histogram data to a text file.

17. Determine the histogram box co-ordinates for late bottom histogram:

- Late bottom histogram Left = crossing time + bit time * (Late histogram box position – (histogram width/2))
- Late bottom histogram Right = crossing time + bit time * (Late histogram box position + (histogram width/2))
- Late bottom histogram Base = AOP
- Late bottom histogram Top = Maximum amplitude value + (vertical scale/2)

Do not re-run the acquisition. Export the Late bottom histogram data to a text file.

18. Determine OMA. See *Optical Modulation Amplitude* from SR4-TDEC Help for the procedure.

Post-processing:

- Sigma computation

1. Create histogram Y-axis for both top histograms and bottom histograms (Y-axis does not change with reference to early or late):

- Dumped/exported histogram will have 400 points corresponding to the rows of waveform database.
- Histogram Width = (Top Screen Voltage – Bottom Screen Voltage) / 400
- Y-axis of full screen is computed as Top screen voltage to bottom screen voltage with step size as histogram width
- Y-axis corresponding to top histogram is computed as Y-axis points between early top histogram top and early top histogram base
- Y-axis corresponding to bottom histogram is computed as Y-axis points between early bottom histogram top and early bottom histogram base

2. Normalize histograms for all four so that, the sum of the area of the histogram of bins is 1

3. Compute earlySigma value using normalized early top histogram and normalized early bottom histogram

top_scalingFactor(i) = Q([tophistogramYaxis(i) – (AOP)]/earlySigma)

bottom_scalingFactor(i) = Q([(AOP) – bottomhistogramYaxis(i)]/earlySigma)

where i = 0 to number of bins

$$\text{BER} = \frac{\sum_i[normalizedTophistogram(i) * top\_scalingFactor(i)]}{\sum_i[normalizedTophistogram(i)]} + \frac{\sum_i[normalizedbottomhistogram(i) * bottom\_scalingFactor(i)]}{\sum_i[normalizedbottomhistogram(i)]}$$

Adjust earlySigma until absolute difference between BER and target BER is less than 1e-20

4. Repeat step GUID-243F9146-086A-4634-9D41-88ADF91D0575#GUID-243F9146-086A-4634-9D41-88ADF91D0575/STEP_3 for lateSigma value computation. Use normalized late top histogram and normalized late bottom histogram

5. Determine the larger sigma value, Sigma Value = Maximum (earlySigma, lateSigma)

6. Compute R using scope noise, M1 and M2

$$\text{R} = (1 - \text{M1}) * \sqrt{sigmaValue^2 + scopeNoise^2 - M2^2}$$

Where M1 = 0.04

M2 = 0.0175 * Pav

M1 and M2 account for mode partition noise and modal noise that could be added by optical channel.

- TDEC computation

  1. TDEC is computed using the formula,

     $$\text{TDEC} = 10 * log10\left(\frac{OMA}{(2*R*(\sqrt{2}*erfcinv(targetBER*2))}\right)$$

     Computed TDEC is compared with limits to indicate pass/fail

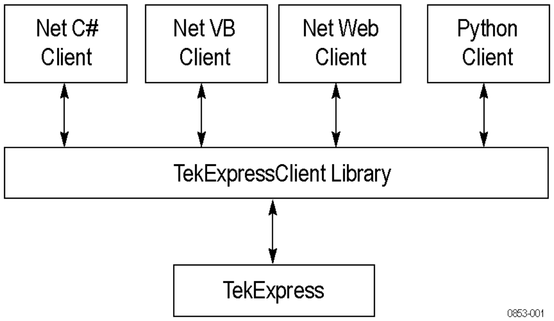
     Limits:

     - Low Limit: NA
     - High Limit: 4.3 dB

TekExpress programmatic interface

About the programmatic interface

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



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

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

TekExpress leverages .Net Marshalling to enable the Programmatic Interface for TekExpress Client. TekExpress provides a client library for TekExpress clients to use the programmatic interface. The TekExpress client library 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

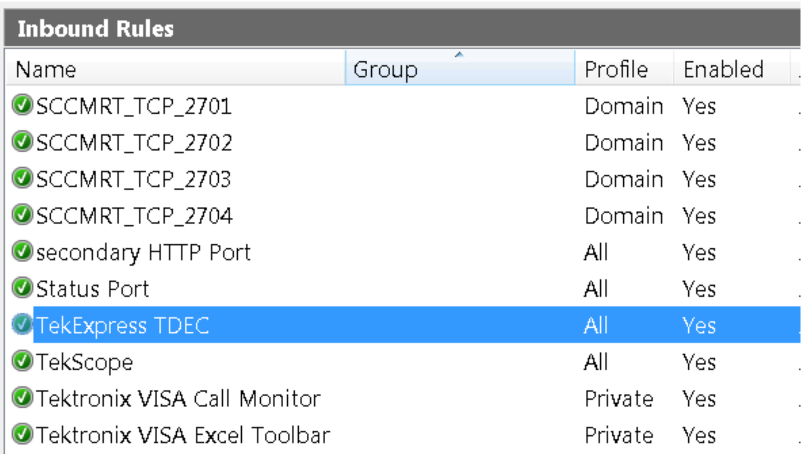
Requirements for developing TekExpress client

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 TDEC
   - TekExpress



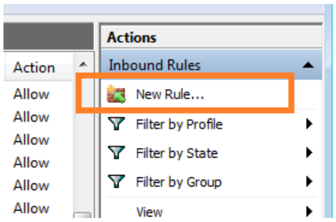
Inbound Rules

Name	Group	Profile	Enabled
SCCMRT_TCP_2701		Domain	Yes
SCCMRT_TCP_2702		Domain	Yes
SCCMRT_TCP_2703		Domain	Yes
SCCMRT_TCP_2704		Domain	Yes
secondary HTTP Port		All	Yes
Status Port		All	Yes
TekExpress TDEC		All	Yes
TekScope		All	Yes
Tektronix VISA Call Monitor		Private	Yes
Tektronix VISA Excel Toolbar		Private	Yes

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

Run the New Inbound Rule Wizard

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

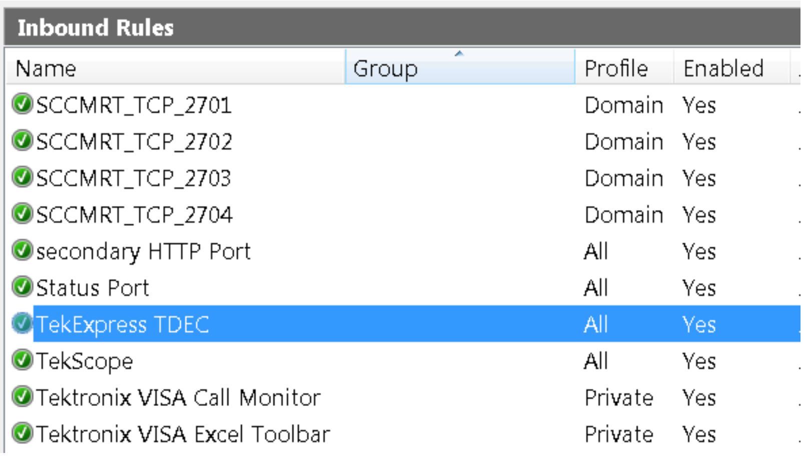


2. Verify that **Program** is selected in the Rule Type panel and click **Next**.
3. Click **Browse** in the Program panel and navigate to and select one of the following TekExpress applications (depending on the one for which you need to create a rule):
   - TekExpress TDEC.exe
   - TekExpress.exe

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

4. Click **Next**.
5. Verify that **Allow the connection** is selected in the Action panel and click **Next**.

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



Inbound Rules

Name	Group	Profile	Enabled
SCCMRT_TCP_2701		Domain	Yes
SCCMRT_TCP_2702		Domain	Yes
SCCMRT_TCP_2703		Domain	Yes
SCCMRT_TCP_2704		Domain	Yes
secondary HTTP Port		All	Yes
Status Port		All	Yes
TekExpress TDEC		All	Yes
TekScope		All	Yes
Tektronix VISA Call Monitor		Private	Yes
Tektronix VISA Excel Toolbar		Private	Yes

10. Repeat steps 1 through 9 to enter the other TekExpress executable if it is missing from the list. Enter **TekExpress PI** as the name.
11. Scroll through the Inbound Rules list and verify that the list shows the rule that you just entered.
12. Exit the Windows Firewall tool.

To use the remote access:

1. Obtain the IP address of the instrument on which you are running TekExpress TDEC. For example, 192.168.34.10.
2. On the PC from which you are accessing the remote instrument, use the instrument IP address as part of the TekExpress TDEC PI code to access that instrument. For example:

```
object obj = piClient.Connect(“192.168.34.10”,out clientid);
```

Requirements for developing TekExpress client

Use the TekExpressClient.dll to develop your client. 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:

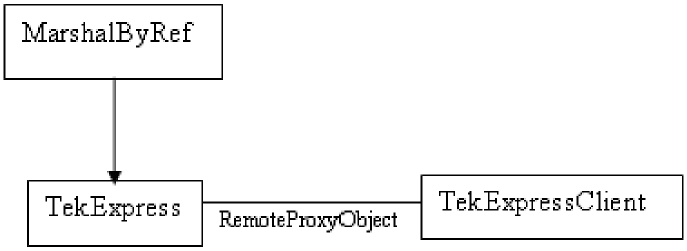
Develop a client UI to access the interfaces exposed through the server. This client loads `TekExpressClient.dll` to access the interfaces. After `TekExpressClient.dll` is loaded, the client UI can call the specific functions to run the operations requested by the client. When the client is up and running, 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. The proxy object is instantiated and exposed at the server-end through marshalling.



The following is an example:

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

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

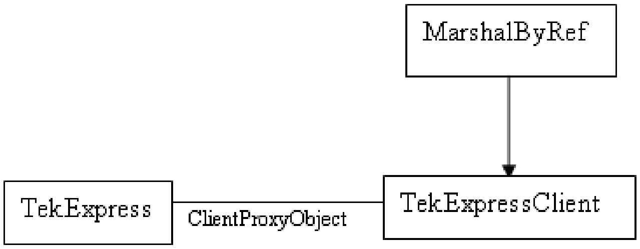
For example,

//Get a reference to the remote object

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

Client proxy object

Client exposes a proxy object to receive certain information.



For example,

//Register the client proxy object

```
WellKnownServiceTypeEntry[] e =
RemotingConfiguration.GetRegisteredWellKnownServiceTypes();
```

```
clientInterface = new ClientInterface();
```

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

//Expose the client proxy object through marshalling

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

The client proxy object is used for the following:

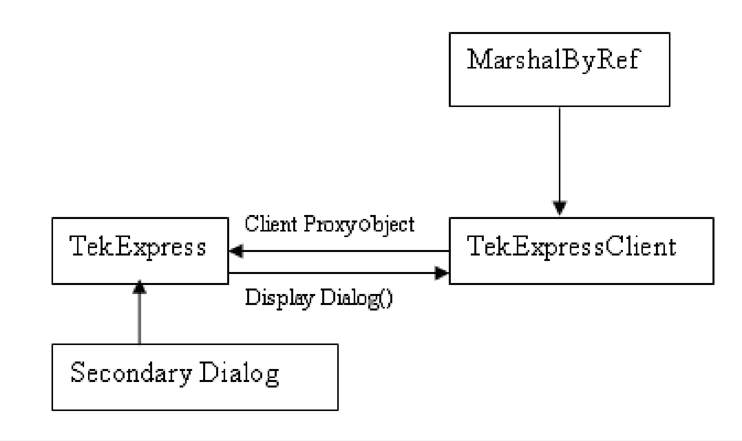
- To get the secondary dialog messages from the server.
- To get the file transfer commands from the server while transferring the report.

Examples

```
clientObject.clientIntf.DisplayDialog(caption, msg,iconType, btnType);
clientObject.clientIntf.TransferBytes(buffer, read, fileLength);
```

For more information, click the following links:

Secondary Dialog Message Handling



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

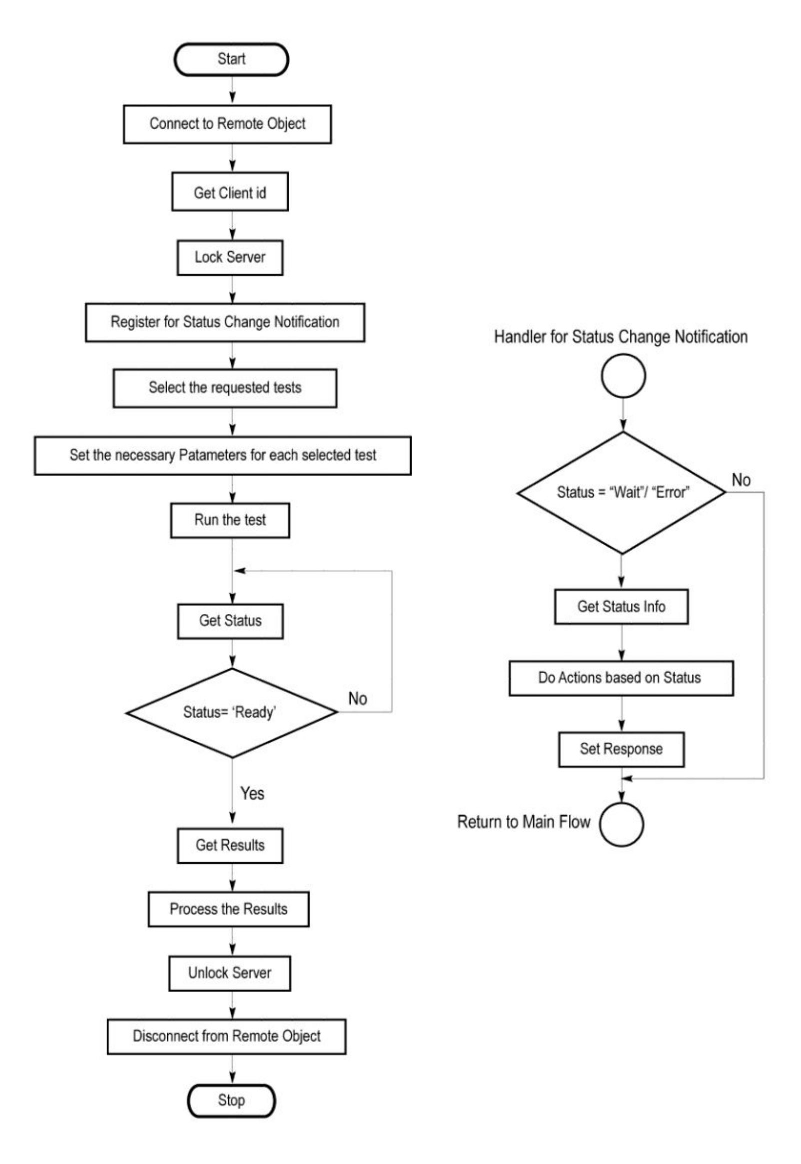
In the secondary dialog library, the assembly that is calling for the dialog box to be displayed is checked and if a remote connection is detected, the messages are directed to the remote end.

File Transfer Events

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

Client programmatic interface example

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



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

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

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

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

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

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

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

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

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

Handler of status change notification

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

See also

Program remote access code example

Program remote access code example

This code example shows how to communicate between a remote PC and TekExpress 100GBASE-SR4.

Table 15: Remote access code example

Task	Code
Start the application	

Connect through an IP address.	'assigns client IP address to variable clientID; address valid until connection or measurement session ends (Disconnect). See Connect()
	`clientID = “ ”`
	`m_Client.Connect("localhost",out clientID)'True or False`

Lock the server	`m_Client.LockServer(clientID)`
Disable the Popups	`m_Client.SetVerboseMode(clientID, false)`
Set the DUT ID	`m_Client.SetDutId(clientID, "DUT_Name")`
Run with set configurations	`m_Client.Run(clientID)`

Wait for the test to complete.	`Do`
	`Thread.Sleep(500)`
	`m_Client.Application_Status(clientID)`
	`Select Case status`
	`Case "Wait"`

Task	Code

Get the current state information	`mClient.GetCurrentStateInfo(clientID,`
	`WaitingMsbBxCaption, WaitingMsbBxMessage,`
	`WaitingMsbBxButtontexts)`

Send the response	`mClient.SendResponse(clientID,`
	`WaitingMsbBxCaption, WaitingMsbBxMessage,`
	`WaitingMsbBxResponse)`
	`End Select`
	`Loop Until status = "Ready"`

Save results	'Save all results values from folder for current run
	`m_Client.TransferResult(clientID,`
	`logDirname)`

Unlock the server	`m_Client.UnlockServer(clientID)`
Disconnect from server	`m_Client.Disconnect()`
Exit the application	

TDEC programmer interface commands

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.
			clientId variable
			clientId variable
			clientId is a user-defined variable that stores the client ID address information. Use the Connect() command to fill this variable:
			`clientID = “ ”`
			`m_Client.Connect("localhost",out clientId)'True or False`
			The clientId variable is stored until you call the Disconnect command.

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

QueryStatus

SendResponse

Status

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.
			clientId variable
			clientId variable
			clientId is a user-defined variable that stores the client ID address information. Use the Connect() command to fill this variable:
			`clientID = “ ”`
			`m_Client.Connect("localhost",out clientId)'True or False`
			The clientId variable is stored until you call the Disconnect command.

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*

CheckSessionSaved()

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

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.
clientId	string	IN	Identifier of the client that is performing the remote function. clientId variable

savedStatus	boolean	OUT	Boolean representing whether the current session is saved
			clientId variable
			clientId is a user-defined variable that stores the client ID address information. Use the Connect() command to fill this variable:
			`clientID = “ ”`
			`m_Client.Connect("localhost",out clientId)'True or False`
			The clientId variable is stored until you call the Disconnect command.

Return value. Return value is either True or False.

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

returnval as string

returnval=m_Client.CheckSessionSaved(m_clientID, out savedStatus)

Related command(s). *GetDutId*

Comments.

Connect()

Connect(string HostIPAddress, out string clientID). This command connects the client to the server; address is the IP address of the server to which the client is trying to connect. This is required to establish the connection between the client and the server.

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	Obtains 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.

clientId	string	OUT	Identifier of the client that is performing the remote function.
			clientId variable
			clientId variable
			clientId is a user-defined variable that stores the client ID address information. Use the Connect() command to fill this variable:
			`clientID = “ ”`
			`m_Client.Connect("localhost",out clientId)'True or False`
			The clientId variable is stored until you call the Disconnect command.

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 {
m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL
clientID = “ ”
m_Client.Connect("localhost",out clientID)'True or False
}
```

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*

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.
			clientId variable
			clientId variable
			clientId is a user-defined variable that stores the client ID address information. Use the Connect() command to fill this variable:
			`clientID = “ ”`
			`m_Client.Connect("localhost",out clientId)'True or False`
			The clientId variable is stored until you call the Disconnect command.

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*

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 t
			clientId variable
			clientId is a user-defined variable that stores the client ID address information. Use the Connect() command to fill this variable:
			`clientID = “ ”`
			`m_Client.Connect("localhost",out clientId)'True or False`
			The clientId variable is stored until you call the Disconnect command.
			he remote function.
			clientId variable
			clientId variable
			clientId is a user-defined variable that stores the client ID address information. Use the Connect() command to fill this variable:
			`clientID = “ ”`
			`m_Client.Connect("localhost",out clientId)'True or False`
			The clientId variable is stored until you call the Disconnect command.

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*

QueryStatus

SendResponse

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.
			clientId variable
			clientId variable
			clientId is a user-defined variable that stores the client ID address information. Use the Connect() command to fill this variable:
			`clientID = “ ”`
			`m_Client.Connect("localhost",out clientId)'True or False`
			The clientId variable is stored until you call the Disconnect command.

dutId	string	OUT	The DUT id of the setup.

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

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*

SetDutId

GetPassFailStatus()

GetPassFailStatus(clientId, device, suite, 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
			clientId variable
			clientId variable
			clientId is a user-defined variable that stores the client ID address information. Use the Connect() command to fill this variable:
			`clientID = “ ”`
			`m_Client.Connect("localhost",out clientId)'True or False`
			The clientId variable is stored until you call the Disconnect command.

device	string	IN	Specifies the DUT type (**TDEC**)
suite	string	IN	string with device connection type. Suite type is TP2
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, “TDEC”, “TP2”, 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
			clientId variable
			clientId variable
			clientId is a user-defined variable that stores the client ID address information. Use the Connect() command to fill this variable:
			`clientID = “ ”`
			`m_Client.Connect("localhost",out clientId)'True or False`
			The clientId variable is stored until you call the Disconnect command.

device	string	IN	Specifies the DUT type (**TDEC**).
suite	string	IN	string with device connection type. Suite type is TP2.
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 100GBASE-SR4 application version.

Example. GetReportParameter(clientId, “Device”, “suite”, test, "Application Version")

GetResultsValue()

GetResultsValue(clientId, device, suite, 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
			clientId variable
			clientId variable
			clientId is a user-defined variable that stores the client ID address information. Use the Connect() command to fill this variable:
			`clientID = “ ”`
			`m_Client.Connect("localhost",out clientId)'True or False`
			The clientId variable is stored until you call the Disconnect command.

device	string	IN	Specifies the DUT type (**100GBASE-SR4**).
suite	string	IN	string with device connection type. Suite type is TP2.
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”, “suite”, 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
			clientId variable
			clientId variable
			clientId is a user-defined variable that stores the client ID address information. Use the Connect() command to fill this variable:
			`clientID = “ ”`
			`m_Client.Connect("localhost",out clientId)'True or False`
			The clientId variable is stored until you call the Disconnect command.

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*

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.
			clientId variable
			clientId variable
			clientId is a user-defined variable that stores the client ID address information. Use the Connect() command to fill this variable:
			`clientID = “ ”`
			`m_Client.Connect("localhost",out clientId)'True or False`
			The clientId variable is stored until you call the Disconnect command.

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*

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
			clientId variable
			clientId variable
			clientId is a user-defined variable that stores the client ID address information. Use the Connect() command to fill this variable:
			`clientID = “ ”`
			`m_Client.Connect("localhost",out clientId)'True or False`
			The clientId variable is stored until you call the Disconnect command.

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*

GetCurrentStateInfo

SendResponse

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. clientId variable
sessionName	string	IN	The name of the session being recalled.

clientId variable

clientId is a user-defined variable that stores the client ID address information. Use the Connect() command to fill this variable:

```
clientID = “ ”
```

```
m_Client.Connect("localhost",out clientId)'True or False
```

The clientId variable is stored until you call the Disconnect command.

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*

SaveSessionAs

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 clientId variable

clientId variable

clientId is a user-defined variable that stores the client ID address information. Use the Connect() command to fill this variable:

```
clientID = “ ”
```

```
m_Client.Connect("localhost",out clientId)'True or False
```

The clientId variable is stored until you call the Disconnect command.

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 clientId variable
sessionName	string	IN	The name of the session being saved.

clientId variable

clientId is a user-defined variable that stores the client ID address information. Use the Connect() command to fill this variable:

```
clientID = “ ”

m_Client.Connect("localhost",out clientId)'True or False
```

The clientId variable is stored until you call the Disconnect command.

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*

SaveSessionAs

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 clientId variable
sessionName	string	IN	The name of the session being saved.

clientId variable

clientId is a user-defined variable that stores the client ID address information. Use the Connect() command to fill this variable:

```
clientID = “ ”
m_Client.Connect("localhost",out clientId)'True or False
```

The clientId variable is stored until you call the Disconnect command.

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*

SaveSession

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 clientId variable
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

clientId variable

clientId is a user-defined variable that stores the client ID address information. Use the Connect() command to fill this variable:

```
clientID = " "
m_Client.Connect("localhost",out clientId)'True or False
```

The clientId variable is stored until you call the Disconnect command.

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*

GetCurrentStateInfo

QueryStatus

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 clientId variable
device	string	IN	Specifies the DUT type (**TDEC**)

clientId variable

clientId is a user-defined variable that stores the client ID address information. Use the Connect() command to fill this variable:

```
clientID = “ ”
m_Client.Connect("localhost",out clientId)'True or False
```

The clientId variable is stored until you call the Disconnect command.

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

Example. SelectDevice(clientId, "TDEC", True)

SelectSuite()

SelectSuite(clientId, device, suite, true). This command selects TP2 suite.

Parameters.

Parameter	Type	Direction	Description
clientId	string	IN	Identifier of the client that is connected to the server clientId variable
device	string	IN	Specifies the DUT type (**TDEC**)
suite	string	IN	string with device connection type. Suite type is TP2.

clientId variable

clientId is a user-defined variable that stores the client ID address information. Use the Connect() command to fill this variable:

```
clientID = “ ”
m_Client.Connect("localhost",out clientId)'True or False
```

The clientId variable is stored until you call the Disconnect command.

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

Example. SelectSuite(clientId,"TDEC","TP2",true);

SelectTest()

SelectTest(clientId, device, suite, 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
			clientId variable
			clientId variable
			clientId is a user-defined variable that stores the client ID address information. Use the Connect() command to fill this variable:
			`clientID = “ ”`
			`m_Client.Connect("localhost",out clientId)'True or False`
			The clientId variable is stored until you call the Disconnect command.

device	string	IN	Specifies the DUT type (TDEC).
suite	string	IN	string with device connection type. Suite type is TP2
test	string	IN	Name of the TDEC test.

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

Example. SelectTest(clientId, device, suite, "Optical Modulation Amplitude", 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
			clientId variable
			clientId variable
			clientId is a user-defined variable that stores the client ID address information. Use the Connect() command to fill this variable:
			`clientID = “ ”`
			`m_Client.Connect("localhost",out clientId)'True or False`
			The clientId variable is stored until you call the Disconnect command.

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

The SetGeneralParameter command

Select data rate. Use this paramString value to set the Data Rate used by the application. This is the same as selecting the Data Rate control on the DUT tab.

The value in bold font is the default value.

Values:.

- **Data Rate$25.781 Gbps**
- Data Rate$27.952 Gbps
- Data Rate$28.050 Gbps
- Data Rate$Custom

Example. SetGeneralParameter(clientId, “TDEC”, “TP2”, “”, “Data Rate (Gbps)$27.952 Gbps”);

NOTE. *To select the custom option in the data rate drop down following command need to be run as a pre-requisite:*

- *SetComplianceMode(clientID, User-Defined)*

Select custom data rate. Use this paramString value to set the custom data rate used by the application. This is the same as selecting the custom data rate control on the DUT tab.

The value in bold font is the default value.

Values:.

- Custom Data Rate$<Enter any value between the low and high limits of the Custom Data rate option>

Example. Custom Data Rate$15

NOTE. *To run this command following pre-requisite commands should be run prior with the same following sequence:*

- *SetComplianceMode(clientID, User-Defined)*
- *Data Rate$Custom*

Select wavelength. Use this paramString value to set the Wavelength used by the application. This is same as selecting wavelength on the DUT tab.

The value in bold font is the default value.

Values:.

- WaveLength(nm)$1550 : FACTORY
- WaveLength(nm)$1310 : FACTORY
- WaveLength(nm)$850 : FACTORY
- WaveLength(nm)$1550 : USER

Example. SetGeneralParameter(clientId, “TDEC”, “TP2”, “”, “WaveLength(nm)$1550 : FACTORY”);

***NOTE.** The example values are provided considering the 80C15 module is connected to a scope.*

Select OMA method. Use this paramString value to set the pattern for OMA method used by the application. This is same as selecting OMA method on Setup -> Configuration -> Global Settings -> OMA Method.

The value in bold font is the default value.

Values:.

- OMA Method$OMA-Eye
- OMA Method$OMA-Pattern

Example. SetGeneralParameter(clientId, “TDEC”, “TP2”, “”, “OMA Method$OMA-Eye”);

Select pattern for OMA-pattern. Use this paramString value to set the pattern for OMA-pattern used by the application. This is same as selecting OMA-pattern on Setup -> Configuration -> Global Settings -> OMA Method -> OMA-pattern -> Pattern.

The value in bold font is the default value.

Values:.

- Data Pattern$PRBS7
- Data Pattern$PRBS9
- Data Pattern$PRBS11
- Data Pattern$PRBS15

Example. SetGeneralParameter(clientId, “TDEC”, “TP2”, “”, “Data Pattern$PRBS9”);

***NOTE.** To run this command, following command is the prerequisite.*

- *SetGeneralParameter(clientId, “TDEC”, “TP2”, “”, “OMA Method$OMA-Pattern”);*

Select trigger source. Use this paramString value to set the Trigger Source used by the application. This is same as selecting SR4 filter on Setup -> Configuration -> Global Settings -> Trigger Source.

The value in bold font is the default value.

Values:.

- Trigger Source$Tek CRU
- Trigger Source$Others

Example. SetGeneralParameter(clientId, “TDEC”, “TP2”, “”, “Trigger Source$TekCRU”);

Select data to clock ratio. Use this paramString value to set the Data to Clock ratio used by the application. This is same as selecting SR4 filter on Setup -> Configuration -> Global Settings -> Trigger Source -> Others -> Clock Divider.

The value in bold font is the default value.

Values:.

- Data to Clock ratio$1
- Data to Clock ratio$2
- Data to Clock ratio$4
- Data to Clock ratio$8

Example. SetGeneralParameter(clientId, “TDEC”, “TP2”, “”, “Data to Clock ratio$1”);

NOTE. *To run this command, following command is the prerequisite.*

- *SetGeneralParameter(clientId, “TDEC”, “TP2”, “”, “Trigger Source$Others”);*

Select TDEC filter. Use this paramString value to set the TDEC filter used by the application. This is same as selecting TDEC filter on Setup -> Configuration -> Global Settings -> TDEC Signal Conditioning -> Filter.

The value in bold font is the default value.

Values:.

- Filter_TDECGlobal$OTU-4
- Filter_TDECGlobal$32GFCr0
- Filter_TDECGlobal$100GBASE-R4
- Filter_TDECGlobal$INF25781

Example. SetGeneralParameter(clientId, “TDEC”, “TP2”, “”, “Filter_TDECGlobal$OTU-4”);

NOTE. *The example values are provided considering the 80C15 module is connected to a scope.*

Select TDEC bandwidth. Use this paramString value to set the TDEC bandwidth used by the application. This is same as selecting TDEC bandwidth on Setup -> Configuration -> Global Settings -> TDEC Signal Conditioning -> Bandwidth.

The value in bold font is the default value.

Values:.

- BandWidth_TDECGlobal$32.00GHz
- BandWidth_TDECGlobal$28.05GHz
- BandWidth_TDECGlobal$22.00GHz

Example. SetGeneralParameter(clientId, “TDEC”, “TP2”, “”, “BandWidth_TDECGlobal$32.00GHz”);

NOTE. *The example values are provided considering the 80C15 module is connected to a scope.*

Select TDEC BER mantissa. Use this paramString value to set the TDEC BER mantissa used by the application. This is same as selecting TDEC BER mantissa on Setup -> Configuration -> Global Settings -> TDEC Signal Conditioning -> BER (Mantissa editbox).

The value in bold font is the default value.

Values:.

- BER_Mantissa_TDECGlobal$<Enter any value between the low and high limits of the TDEC BER Mentissa edit box>

Example. SetGeneralParameter(clientId, “TDEC”, “TP2”, “”, “BER_Mantissa_TDECGlobal$5.0”);

Select TDEC BER exponent. Use this paramString value to set the TDEC BER exponent used by the application. This is same as selecting TDEC BER exponent on Setup -> Configuration -> Global Settings -> TDEC Signal Conditioning -> BER (Exponent editbox).

The value in bold font is the default value.

Values:.

- BER_Exponent_TDECGlobal$<Enter any value between the low and high limits of the TDEC BER Exponent edit box>

Example. SetGeneralParameter(clientId, “TDEC”, “TP2”, “”, “BER_Exponent_TDECGlobal$5”);

Select TDEC histogram hits. Use this paramString value to set the TDEC Histogram hits used by the application. This is same as selecting TDEC histogram hits on Setup -> Configuration -> Global Settings -> TDEC Signal Conditioning -> Histogram Hits.

The value in bold font is the default value.

Values:.

- Histogram_Hits_TDECGlobal$<Enter any value between the low and high limits of the TDEC BER Histogram hits edit box>

Example. SetGeneralParameter(clientId, “TDEC”, “TP2”, “”, “Histogram_Hits_TDECGlobal$100000”);

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 clientId variable
time	string	IN	The time in seconds that refers to the timeout period

clientId variable

clientId is a user-defined variable that stores the client ID address information. Use the Connect() command to fill this variable:

```
clientID = “ ”
```

```
m_Client.Connect("localhost",out clientId)'True or False
```

The clientId variable is stored until you call the Disconnect command.

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.

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 clientId variable
verboseMode	boolean	IN	Sets the verbose mode to be turned ON (true) or OFF (false).

clientId variable

clientId is a user-defined variable that stores the client ID address information. Use the Connect() command to fill this variable:

```
clientID = “ ”
```

```
m_Client.Connect("localhost",out clientId)'True or False
```

The clientId variable is stored until you call the Disconnect command.

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
			clientId variable
			clientId variable
			clientId is a user-defined variable that stores the client ID address information. Use the Connect() command to fill this variable:
			`clientID = “ ”`
			`m_Client.Connect("localhost",out clientId)'True or False`
			The clientId variable is stored until you call the Disconnect command.

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*

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 clientId variable

clientId variable

clientId is a user-defined variable that stores the client ID address information. Use the Connect() command to fill this variable:

```
clientID = “ ”
m_Client.Connect("localhost",out clientId)'True or False
```

The clientId variable is stored until you call the Disconnect command.

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
			clientId variable
			clientId variable
			clientId is a user-defined variable that stores the client ID address information. Use the Connect() command to fill this variable:
			`clientID = “ ”`
			`m_Client.Connect("localhost",out clientId)'True or False`
			The clientId variable is stored until you call the Disconnect command.

filePath	string	IN	The location where the screen shots must be saved in the client. ***NOTE.*** *If the client does not provide the location to save the report, the report is saved at C:\ProgramFiles.*

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

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
			clientId variable
			clientId variable
			clientId is a user-defined variable that stores the client ID address information. Use the Connect() command to fill this variable:
			`clientID = “ ”`
			`m_Client.Connect("localhost",out clientId)'True or False`
			The clientId variable is stored until you call the Disconnect command.

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 clientId variable

clientId variable

clientId is a user-defined variable that stores the client ID address information. Use the Connect() command to fill this variable:

```
clientID = “ ”
```

```
m_Client.Connect("localhost",out clientId)'True or False
```

The clientId variable is stored until you call the Disconnect command.

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*

Reference

100GBASE-SR4 technology overview

100GBASE-SR4 is a 100 Gigabit Ethernet technology based application. It provides point to point 100 Gb/s Ethernet link over four pair of multimode fiber, up to100 meters. This application uses eight fiber links with four 25.78125 Gb/s lanes for each direction.

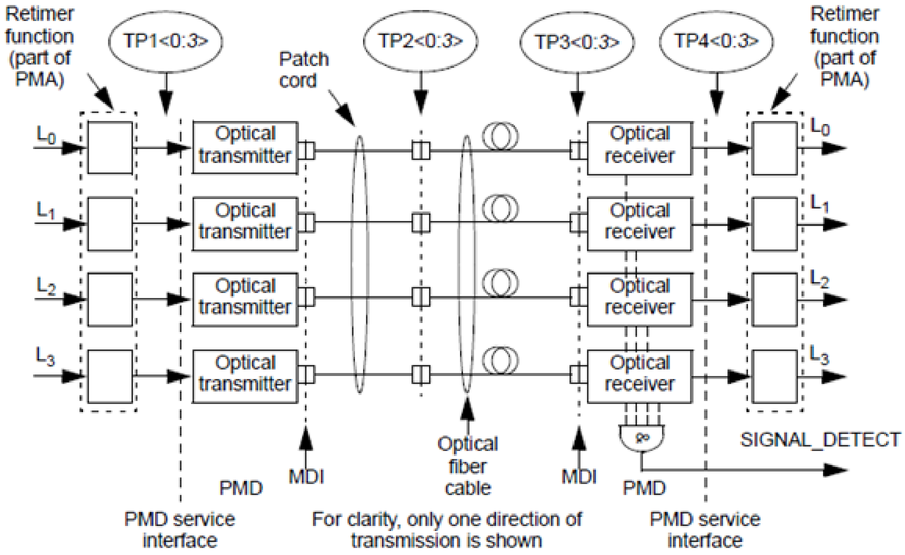
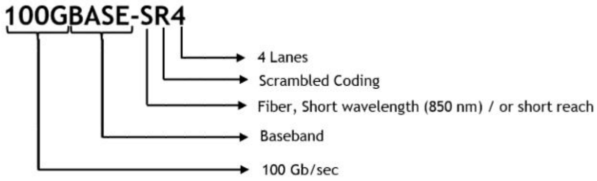


Figure 2: Block diagram of 100GBASE-SR4 Transmit/Receive paths

Interpretation of 100GBASE-SR4 is as follows:



Optical configurations

Table 16: Optical configurations for TDEC measurement

Data Rate (Gbps)	TDEC		
	80C15		
	Filter	Bandwidth (GHz)	Wavelength (nm)
25.78125	None	16.8	850
27.952	None	16.8	850
28.05	None	16.8	850

Clock Recovery Unit (CRU)

Tektronix Clock Recovery Unit is configured with Nominal Data rate (set by user in GUI), Corner frequency of 10 MHz, Slope of 20 dB/decade, i.e, 0 dB peaking, and Lock range of 10 MHz.

The CRU locked data rate value is used as result for Signaling Rate measurement. The following flow diagram gives detailed flow for CRU locking mechanism followed in 100GBASE-SR4 application.

Sub-rate clock output of CRU is given to clock pre scale/trigger input of sampling scope. Clock output of CRU is given to phase reference module.

NOTE. *100GBASE-SR4 application supports only external Clock Recovery Unit.*

Trigger Source

The clock signal (synchronous to data) can be provided using either Tektronix external CRU, or other source having clock signal in synchronous with data, and perform similar to Tektronix external CRU. If the trigger source is others, then you can configure the clock divider parameter.

Clock divider is the ratio of data rate to frequency of clock signal, fed as input to phase reference. It is used to determine the frequency of phase characterization (used only if phase reference module is present in one of the slot in the main frame).

Phase characterization frequency = Data rate / Clock divider

Phase reference characterization

Phase reference module is not a mandatory requirement for 100GBASE-SR4 measurements. If phase reference module is present in any sampling scope slots, then the setup provides a clock signal synchronous with data as input to phase reference module (can use recovered clock from CRU). Phase reference characterization is done with phase correction mode as “triggered” and input frequency equal to frequency of the input clock signal.

***NOTE.** The recovered clock frequency from CR286A is half of the data rate, when the data rate is greater than 14.3 Gb/sec.*

The 100GBASE-SR4 application uses only one phase reference module; if the system has multiple modules, then the lower numbered slot is used and others are ignored. This slot/channel information is obtained from phase reference source query, using instrument programmatic interface internally.

If there is no phase reference module, then query results in C1C2 (default), and perform an additional query of module’s model number. If the model number is 82A04B, then proceed with phase reference characterization, else skip phase reference characterization.

Parameters

About application parameters

This section describes the 100GBASE-SR4 application parameters, and includes the default menu settings.

The parameters for the menus, and options list the selections available for each and include the default values.

Setup panel parameters

DUT tab parameters.

Parameters	Selection	Default Setting
DUTID	-	DUT001
Data Rate	25.781 Gbps, 27.952 Gbps, 28.05 Gbps, Custom (10 to 28.05 Gbps)	25.781 Gbps
Wavelength	List of wavelengths supported by the connected Optical module	850 : FACTORY (if available, else the lowest wavelength supported by the connected optical module)

Test Selection tab parameters.

Parameters	Selection	Default Setting
TDEC	■ Transmitter and Dispersion Eye Closure	Selected

Configuration tab parameters.

Table 17: Global settings parameters

Parameters		Selection	Default Setting
OMA Method		OMA-Eye, OMA-Pattern	OME-Eye
Trigger Source		Tek CRU, Others	Tek CRU
Clock Divider ***NOTE.** It is the ratio of Signaling Rate to Phase Reference characterisation frequency. This is displayed when Trigger Source selection is Others*		1,2,4,8	1
TDEC Signal Conditioning	Filter	List of Filters supported by the connected optical module	Filter default setting is selected based on the available filter options.
	Bandwidth	List of Bandwidths supported by the connected optical module	If 80C15 module is selected, then it is 16.84 GHz or None
	BER	1.0E-18 to 9.0E-0	5.0E-5
	Histogram Hits	1000 to 10000000	100000

Preferences tab parameters.

Parameters	Selection	Default Setting
Acquire/Analyze each test X times	1 to 100	1
Auto close Warnings and Information during Sequencing Auto close after X Seconds	1 to 100	10
Auto close Error Messages during Sequencing, Show in Reports Auto close after X Seconds	1 to 100	10

Reports panel parameters

Parameters	Selection	Default Setting
Report name	-	x:\TDEC\Reports\DUT001.mht
Save as Type	PDF, MHT	Web Archive (*.mht; *.mhtml)

Index