Keysight N7700 Photonic Application Suite

Fast Spectral Loss Measurement Engine

User's Guide

KEYSIGHT TECHNOLOGIES

Notices

Manual Part Number

N7700-91102

Edition

Edition 12.0, August 2021

Printed in Germany

Keysight Technologies Deutschland GmbH
Herrenberger Strasse 130,
71034 Böblingen, Germany

Technology Licenses

The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license.

U.S. Government Rights

The Software is “commercial computer software,” as defined by Federal Acquisition Regulation (“FAR”) 2.101. Pursuant to FAR 12.212 and 27.405-3 and Department of Defense FAR Supplement

(“DFARS”) 227.7202, the U.S. government acquires commercial computer software under the same terms by which the software is customarily provided to the public. Accordingly, Keysight provides the Software to U.S. government customers under its standard commercial license, which is embodied in its End User License Agreement (EULA), a copy of which can be found at http://www.keysight.com/find/sweula. The license set forth in the EULA represents the exclusive authority by which the U.S. government may use, modify, distribute, or disclose the Software. The EULA and the license set forth therein, does not require or permit, among other things, that Keysight: (1) Furnish technical information related to commercial computer software or commercial computer software documentation that is not customarily provided to the public; or (2) Relinquish to, or otherwise provide, the government rights in excess of these rights customarily provided to the public to use, modify, reproduce, release, perform, display, or disclose commercial computer software or commercial computer software documentation. No additional government requirements beyond those set forth in the EULA shall apply, except to the extent that those terms, rights, or licenses are explicitly required from all providers of commercial computer software pursuant to the FAR and the DFARS and are set forth specifically in writing elsewhere in the EULA. Keysight shall be under no obligation to update, revise or otherwise modify the Software. With respect to any technical data as defined by FAR 2.101, pursuant to FAR 12.211 and 27.404.2 and DFARS 227.7102, the U.S. government acquires no greater than Limited Rights as defined in FAR 27.401 or DFAR 227.7103-5 (c), as applicable in any technical data.

Warranty

THE MATERIAL CONTAINED IN THIS DOCUMENT IS PROVIDED "AS IS," AND IS SUBJECT TO BEING CHANGED, WITHOUT NOTICE, IN FUTURE EDITIONS. FURTHER, TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, KEYSIGHT DISCLAIMS ALL WARRANTIES, EITHER EXPRESS OR IMPLIED WITH REGARD TO THIS MANUAL AND ANY INFORMATION CONTAINED HEREIN, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. KEYSIGHT SHALL NOT BE LIABLE FOR ERRORS OR FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH THE FURNISHING, USE, OR PERFORMANCE OF THIS DOCUMENT OR ANY INFORMATION CONTAINED HEREIN. SHOULD KEYSIGHT AND THE USER HAVE A SEPARATE WRITTEN AGREEMENT WITH WARRANTY TERMS COVERING THE MATERIAL IN THIS DOCUMENT THAT CONFLICT WITH THESE TERMS, THE WARRANTY TERMS IN THE SEPARATE AGREEMENT WILL CONTROL.

Safety Notices

CAUTION

A CAUTION notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in damage to the product or loss of important data. Do not proceed beyond a CAUTION notice until the indicated conditions are fully understood and met.

WARNING

A WARNING notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in personal injury or death. Do not proceed beyond a WARNING notice until the indicated conditions are fully understood and met.

Compliance and Environmental Information

Table 1 Compliance and Environmental Information

Safety Symbol	Description
	This product complies with WEEE Directive (2002/96/EC) marking requirements. The affixed label indicates that you must not discard this electrical/electronic product in domestic household waste. Product Category: With reference to the equipment types in WEEE Directive Annex I, this product is classed as a “Monitoring and Control instrumentation” product. Do not dispose in domestic household waste. To return unwanted products, contact your local Keysight office, or see http://about.keysight.com/en/companyinfo/environment/takeback.shtml for more information.

Contents

Keysight N7700 Photonic Application Suite

User's Guide

1 Quick Start Information

This section demonstrates how to connect your instruments optically and electrically for fast wavelength-swept insertion loss measurements using the Photonic Application Suite. It is intended as a quick start guide, although it is strongly recommended that you read through Chapter , "Getting Started," on page 15 and Chapter , "Fast Spectral Loss Measurement Application," on page 23 of this User Guide and through the User Guides of all instruments used in your setup prior to operating the instruments.

NOTE You need to be logged in as an administrator to install the *Photonic Application Software Suite*.

KEYSIGHT TECHNOLOGIES

Please install the *Photonic Application Suite* before you connect any applicable instrument(s) to the USB port of the PC! This ensures that the drivers are available when you connect the instrument(s). If you have connected the instrument(s) prior to software installation, you may have to delete the instrument(s) manually from the Windows Device Manager. This does not apply to instruments that are connected via GPIB.

After installation of the Photonic Application Suite, connect the instruments to the local area network (LAN), USB, or GPIB. You may use USB hubs to increase the number of USB ports on your PC.

Several instruments can also be accessed through your local network. For the connected instruments to be found by Photonic Application Suite engines, they need to be configured in VISA, for example with the Connection Expert which is part of the IO Libraries Suite if Keysight VISA is primary.

After the instruments have been connected and recognized, start the *Launch Pad* located in the start menu. Then click on “Fast Spectral Loss”. On first start, the *Configuration Wizard* appears and searches for all suitable instruments. Follow the instructions given by the *Configuration Wizard*. After finishing the wizard, the application is displayed with all the configured instruments. If you change your hardware configuration, run the wizard again by clicking on *File->Run Configuration Wizard*.

Fast Spectral Loss Measurement Setup

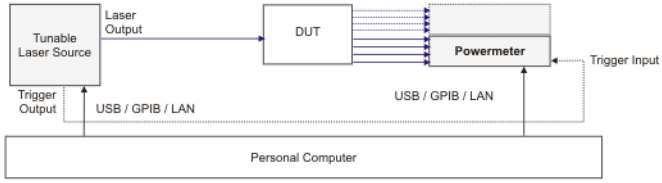
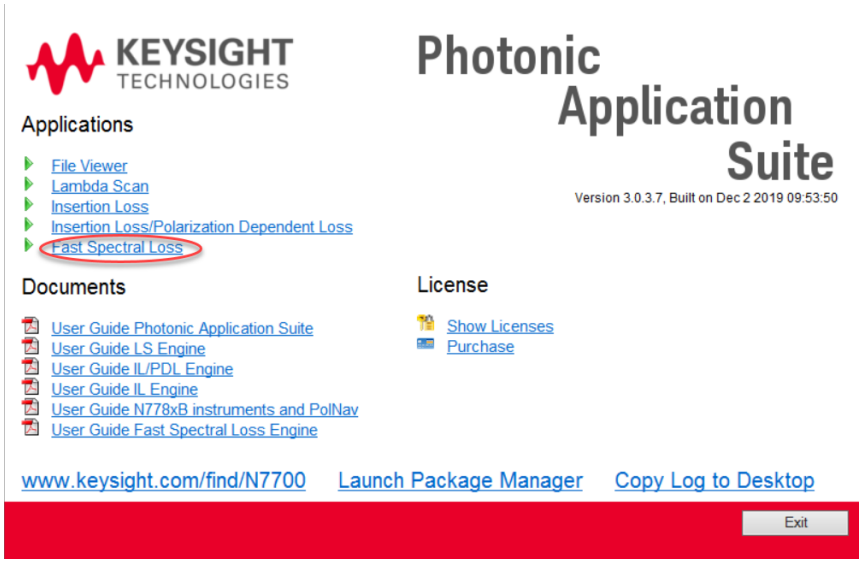


Figure 1 Insertion Loss Setup

NOTE Refer to Measurement Setup on page 49 for more information on measurement setup.

Software User Interface



The *Launch Pad* gives you access to the installed components. Click on “Fast Spectral Loss” to start the Fast Spectral Loss Measurement Engine

NOTE

The Fast Spectral Loss Measurement Engine consists of a *Client Software* which handles the graphical input/output and a *Server Software* which handles the communication with the hardware. For remote control or automation, the controlling program communicates directly with the *Server Software*. The server will be automatically started when you click on “Fast Spectral Loss”. The presence of the server is indicated in the task bar:

Keysight N7700 Photonic Application Suite

User's Guide

2 Getting Started

KEYSIGHT TECHNOLOGIES

About this manual

This manual covers the fast swept-wavelength multi-port insertion loss application that can be performed by combining one or more of supported Keysight's optical power meters with one of supported Keysight's tunable laser sources.

You might also refer to the *N7700 Keysight Photonic Application Suite User's Guide* to get information on the core Photonic Application Suite functionality including the File Viewer, Plugins, and the general COM API.

System Requirements

- Keysight tunable laser source with continuous-sweep capability
- One or more Keysight optical power meter(s) N774xA/C
- One suitable mainframe (8163B or 8164B) only if the tunable laser source is an LMS module
- BNC cables (50 Ohms)
- LAN, GPIB or USB cables
- Personal Computer:
  - Operating System: Microsoft Windows 10 (64 bit)
  - Depending on the instruments, PC interfaces to LAN, USB, and GPIB can be used.

NOTE **Using the TLS in an 8163B / 8164B mainframe will reduce data transfer time, thus increase the measurement repetition rate.**

Software Installation

The Fast Spectral Loss engine communicates with the instruments over a VISA software layer that should be installed before connecting the instruments. Keysight VISA is provided by installing the IO Libraries Suite, which also provides drivers that may be required by some devices, like GPIB interface adapters and instruments connected by USB. Consult the instrument documentation for establishing connections with VISA.

All drivers and supplements necessary for using Keysight instruments with the Fast Spectral Loss engine are included in the Photonic Application Suite installation package. The Package Manager will indicate if any required packages are missing and still need to be installed.

Refer to the *N7700 Photonic Application User's Guide* for information on how to install this software.

Instruments connected via GPIB and often USB will usually be recognized automatically for VISA, for example by Connection Expert when using Keysight VISA. Instruments connected via LAN might need to be added manually. Note that the N77xxC family of instrument will also be identified as LAN instruments in VISA and do not require extra USB drivers. For use with the Fast Spectral Loss engine, instruments should be connected with the primary VISA if more than one VISA installation is used.

NOTE You need to be logged in as an administrator to install the *Photonic Application Software Suite*.

Connecting the instruments

Install the software before you connect the hardware.

Depending upon the type of TLS and power meter mainframes you are going to use for the setup, you have different options for connecting the instruments to the PC. Refer to the user guides of all instruments used, to check how to connect those instruments.

The Photonic Application Suite installs the Keysight IO Libraries, including the Keysight Connection Expert. Make sure that all instruments are shown in the Keysight Connection Expert, then start the Photonic Application Suite software.

NOTE

Instruments connected to the LAN interface are not found automatically by clicking **Refresh All**, but need to be added manually once. Refer to the Keysight Technologies USB/LAN/GPIB Connectivity Guide for further information.

NOTE

The Fast Spectral Loss Measurement application requires the trigger output of the tunable laser source to be connected to the trigger input port of the power meter.

Running the Configuration Wizard

NOTE

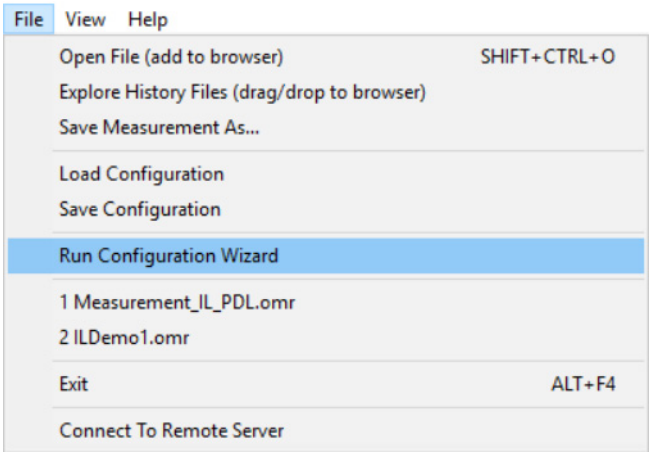
If Keysight VISA is used, all the instruments need to be identified using the Keysight Connection Expert, before the Photonic Application Suite Configuration Wizard is used to configure the Fast Spectral Loss application to use these instruments. The Connection Expert is part of the Keysight IO Libraries. Start the Connection Expert and click the Refresh All button. When all the connected instruments have been identified, proceed with the steps below.

This procedure is only required if a certain supported instrument is connected to the PC for the first time.

Once the software is installed and the instruments are connected and turned on, you can start the **Fast Spectral Loss Engine** and run the **Configuration Wizard**.

On first startup the **Configuration Wizard** will come up automatically.

If you want to rerun the **Configuration Wizard**, you can select on Run Configuration Wizard from the File menu.



Please follow the instructions shown by the **Configuration Wizard**.

Keysight N7700 Photonic Application Suite

User's Guide

3 Fast Spectral Loss Measurement Application

KEYSIGHT TECHNOLOGIES

General

You can run the various applications of the Photonic Application Suite from the Launch pad that is found in the Start menu.

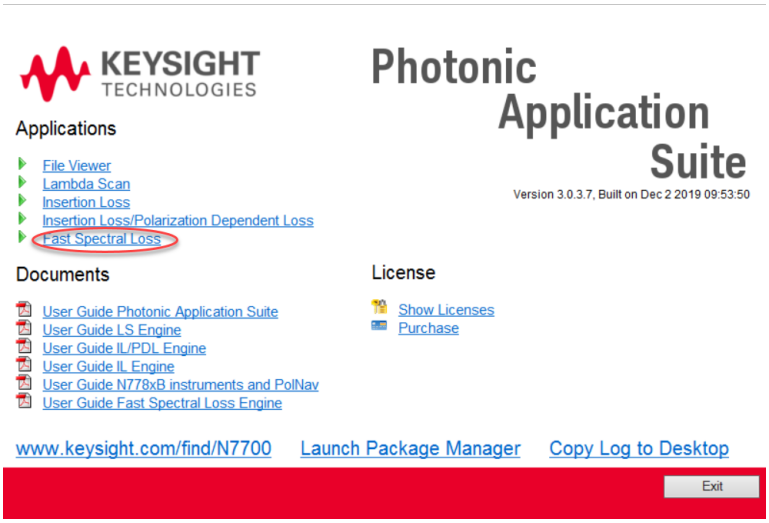


Figure 2 The *Launch Pad* gives you access to the installed components. Click on “Fast Spectral Loss” to start the Fast Spectral Loss Engine.

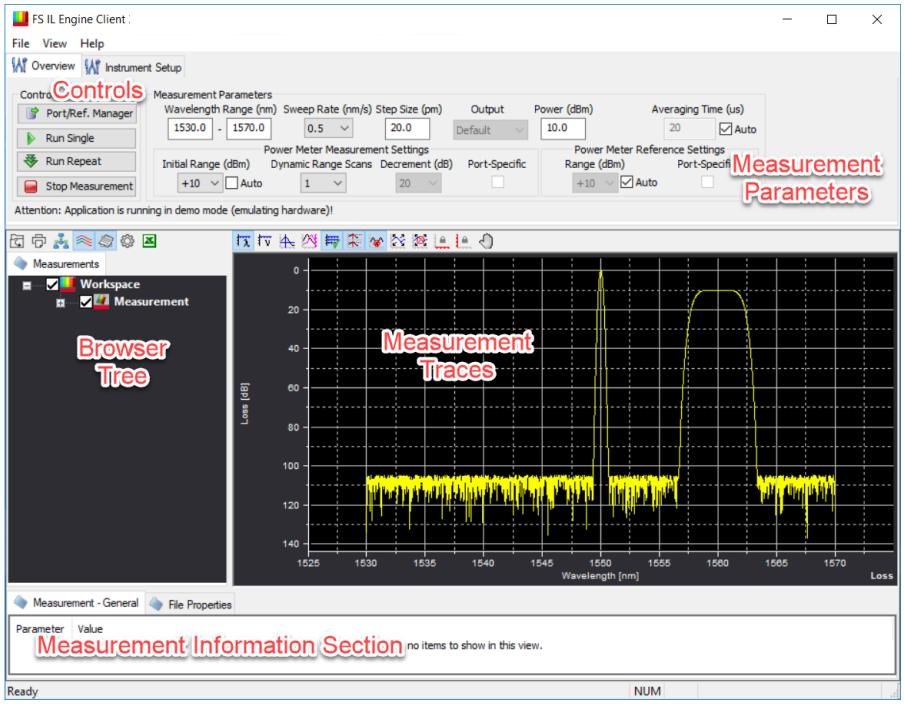
Fast Spectral Loss Measurement Application

Use this application for fast-swept wavelength-resolved multi-channel Insertion Loss measurements. It uses a continuous sweep of the tunable laser source (TLS).

See Measurement Setup on page 49 for details on how to connect the DUT and the trigger cables.

Before starting a measurement you should make sure that the measurement parameters suit your measurement task.

The user interface of the Fast Spectral Loss Measurement engine looks like this:



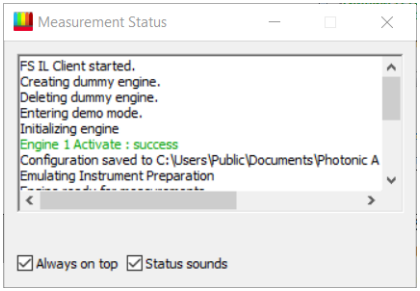
The measurement control buttons are displayed in the top left portion of the application window, the measurement parameters are shown to the right of these. Details on the parameters are given in the tables in the next section.

The browser tree to the left of the application window lists the current measurement as well as all open files. It allows to configure which measurements, which traces (e.g. IL) and which channels are displayed in the measurement view to the right. The measurement information pane can be toggled by clicking the **Show/Hide Info Pane** icon above the browser tree.

For information on how to configure the Measurement File Viewer area, i.e. the browser tree and the display of the measurement traces, refer to the *Photonic Application Suite User Guide*.

Information on the current operation progress is displayed in the *Measurement Status* window. Sometimes user inputs are required, e.g. choosing whether to continue or to abort a certain operation. These user input requests are shown in the Measurement Status window. There will be different response buttons at the bottom of the status window then.

Enable the *Status sounds* check box to receive auditory notifications (beeps) when the status of the engine operations change, for example, when a measurement completes or when an error is encountered.



NOTE

If the *Measurement Status* window is closed, it will open automatically, once any user input is required. It can be opened manually by checking *Measurement Status* in the *View* menu. If the *Measurement Status* window is minimized, it will not be restored automatically if a user request is required. In such case, the application may appear locked, although it is just waiting for a user input. If you're waiting for an operation to complete and are unsure about the current progress, make sure the *Measurement Status* window is visible, so you are aware of any user input requests.

If the Measurement Status window is not visible at all, it may have been moved to a screen region that is currently not visible. By choosing "Reset Status Window Position" from the "View" menu, the Measurement Status window will be put to the top left corner of the application window.

Application Setup Parameters

The **Measurement Setup** parameters are explained in the table below.

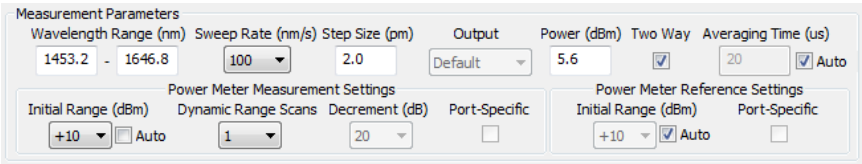


Table 2 Fast Spectral Loss Measurement Application Setup Parameters

Parameter	Description
Start wavelength (nm): (left parameter of Wavelength Range)	Defines the start of the wavelength sweep.
Stop wavelength (nm): (right parameter of Wavelength Range)	Defines the stop of the wavelength sweep.
Sweep Rate (nm/s)	Defines the sweep speed of the TLS. Choosing high sweep rates reduces the time required for each measurement.

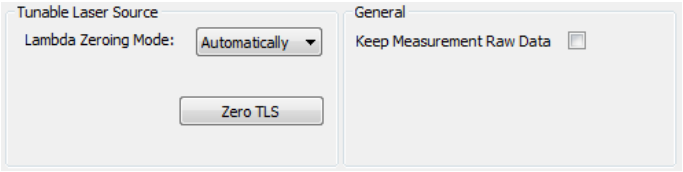
Parameter	Description
Step Size (pm):	Wavelength step size of the measurement results.
Output:	If the TLS supports more than one optical output, the output is chosen here. The choices are “High Power”, “Low SSE”, “Both HR” and “Both LR”. When selecting “Both HR”, both outputs are active with the high power output being regulated. When selecting “Both LR”, both outputs are active with the Low SSE output being regulated.
TLS Power (dBm):	Defines the optical output power of the laser source.
Two-Way:	If checked, measurement data will be obtained during backward scan of the TLS as well, thus significantly increasing the measurement repetition rate. This is only supported by 81960A TLS.
Averaging Time Auto:	If Auto check-box is set, the measurement engine will choose the appropriate averaging time for the measurement automatically.
Averaging Time (us):	Sets the averaging time of the power meter sampling. Higher averaging time will reduce noise on the signals, but may impact features in the transfer characteristics as well, such as steep filter slopes. The minimum averaging time is 1us. The maximum allowed averaging time is defined by several parameters, such as the sweep rate and the step size. The software will automatically adjust this parameter, if it conflicts with other parameters.

Parameter	Description
Initial Range (dBm) (Measurement Setting):	Sets the power range of the measurement. The value defines the maximum allowed power. Power values that exceed this value by more than 3dB will result in clipped results. In case of low power at the power meter (e.g. DUTs with significant loss) the Initial Range should be reduced to utilize maximum dynamic range. By default all ports will use this range. See Range Override on page 34 for details on how to use different ranges for different power meter ports. If Auto check-box is set, the measurement engine will choose the appropriate power range for the measurement automatically. The range will be adjusted individually for each port. The most sensitive range will be chosen for which no clipping will occur.
Dynamic Range Scans:	For measuring devices with very high dynamic ranges, the engine can combine results obtained from measurements performed at different power range settings (stitching). Dynamic Range Scans defines the number of measurements to be combined this way. Use the Decrement setting to define the step by which the power meter range setting will be changed in each dynamic range scan. By default, all ports will use this setting. See Range Override on page 34” for details on how to use different settings for different power meter ports.

Parameter	Description
Decrement:	For measuring devices with very high dynamic ranges, the engine can combine results obtained from measurements performed at different power range settings (stitching). Decrement defines the step by which the power meter range setting will be changed in each dynamic range scan. Use the Dynamic Range Scan setting to define the number of such scans. By default, all ports will use this setting. See Range Override on page 34" for details on how to use different settings for different power meter ports.
Port-Specific:	If either Auto range mode or applying manual modifications in the Port/Reference Manager (Using the Port/Reference Manager on page 33) lead to power related settings that differ for some ports, this check box will be set to indicate that the global settings won't be applied to some (or even any) port. By clicking the check box, you may clear all port-specific settings at once.
Range (dBm) (Reference Measurement):	Sets the power range used in reference sweeps. The value defines the maximum allowed power. Power values that exceed this value by more than 3dB will result in clipped results. in case of low power at the power meter (e.g. setups with significant loss) the Range setting should be reduced to utilize maximum dynamic range. By default, all ports will use this power range. See Range Override on page 34 for details on how to use different ranges for different power meter ports. If Auto check-box is set, the measurement engine will choose the appropriate power range for the reference sweep automatically. The range will be adjusted individually for each port. The most sensitive range will be chosen for which no clipping will occur.

Instrument Setup Parameters

The **Instrument Setup** parameters are located on the **Instrument Setup** tab of the user interface.



These parameters are explained in the table Fast Spectral Loss Measurement Application Instrument Setup Parameters on page 31.

Table 3 Fast Spectral Loss Measurement Application Instrument Setup Parameters

Parameter	Description
Lambda Zeroing Mode:	There are a number of reasons (described below) that may require a lambda zeroing operation. Such an operation takes several ten seconds to complete. To minimize the impact on running measurements, but keeping up the wavelength accuracy, choose one of three settings:
	Always Ask (Default): Whenever a zeroing operation is required, the user will be prompted whether to perform the zeroing operation or not. Once this situation occurs, each subsequent measurement will cause such prompt until either the user allows the zeroing to be performed, or performs the zeroing operation manually from the GUI or changes the **Lambda Zeroing Mode** to either of the other two settings.
	Automatically: Whenever a zeroing operation is required, it is performed automatically right before the next measurement. This is the recommended setting for optimum lambda accuracy.

Parameter	Description
	Manual Only: Although a zeroing operation may be required, it is neither performed automatically, nor will there be a prompt for a user response. Instead a silent notification will be shown in the measurement status box. The user will have to take care of the lambda zeroing.
Zero TLS:	Manually starts the lambda zeroing operation.
Keep Measurement Raw Data:	If checked, measurement raw data will be stored in the measurement files as well. Please note that this is not necessarily the data as it is fetched from the instruments, but after some initial processing. Please refer to the *N7700 Photonic Application Suite User's Guide* for details on accessing/displaying measurement raw data.

NOTE

The necessity for a lambda zeroing operation is determined by several aspects.

For maximum wavelength accuracy, a lambda zeroing operation should be performed, once the operating temperature of the running instrument has settled. Certain TLS modules (those operated in 8164A/B mainframe slot 0, except for 81606A, 81607A, 81608A) will be queried by the software whether the temperature has changed by a certain amount. If it has, the software will act determined by the **Lambda Zeroing Mode** setting. Please note that Compact Tunable Laser Sources do not record such temperature changes, thus won't trigger the software to perform a zeroing based on temperature changes.

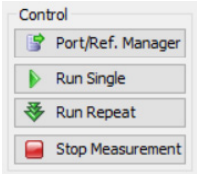
If the TLS is swept across a sweep range smaller than the maximum range very often, the lubrication of the moving parts may no longer be optimal. To mitigate this effect, performing a zeroing operation will cause a uniform distribution of the lubricant. Therefore the software will check how much time has passed since the most recent zeroing operation and will suggest a zeroing operation if a certain amount of time without zeroing is exceeded (this period is 24h or 4h right after running the Configuration Wizard). Please note that the actual number of sweeps and sweep parameters of the TLS have no impact on this period of time.

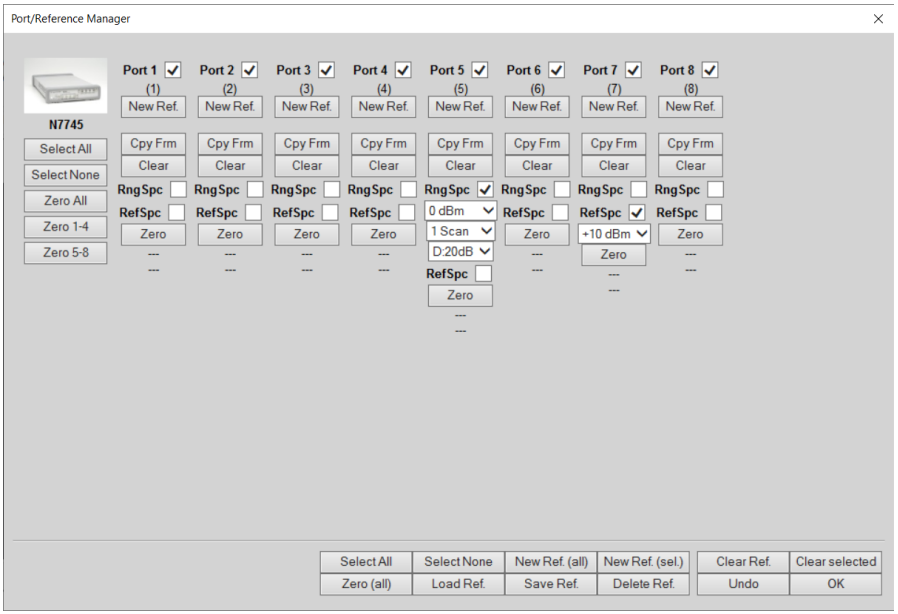
Using the Port/Reference Manager

The Port/Reference Manager is used to:

- Configure which power meter ports should be incorporated into subsequent measurements.
- Configure what power range settings to be used.
- Perform zeroing on individual channels.
- Handle power references.

You can access the Port/Reference Manager by clicking the **Port/Ref. Manager** button to the top-left.





There is a check box for each power meter port, which is used to tell the application whether or not to use that port in subsequent measurements.

The number in parentheses is the overall channel number. In case more than one power meter is used, this number will continue to increase across all power meters used and can be used to match traces from the File Viewer display to the respective physical instruments.

Power Meter Zeroing

Each port has a set of buttons for controlling zeroing and referencing. By clicking the **Zero** button, the corresponding channel is zeroed. There are also buttons for zeroing sets of four ports each (**Zero 1-4, Zero 5-8**) as well as all ports of the current setup (**Zero All**).

NOTE

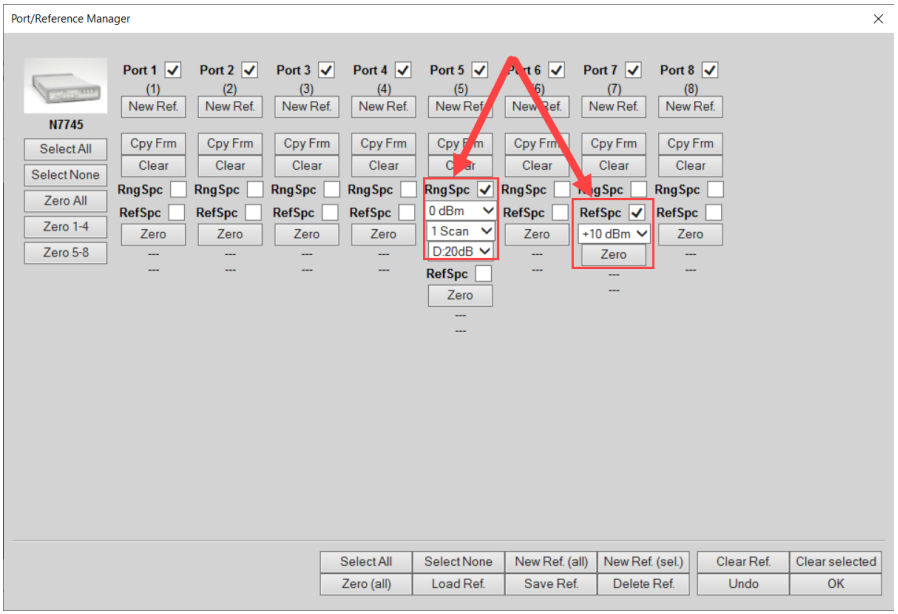
Ensure that the power meter detectors are covered during zeroing. The TLS used by the application is automatically turned off during zeroing. For zeroing the Multiport Power Meters N7744A and N7745A we recommend using at least one zeroing adapter N7740ZI.

Range Override

By default all ports use the measurement range set in the main GUI or through the automation interface (see Application Setup Parameters on page 27 or property PWMSensitivity on page 69). By checking the **RngSpc/RefSpc** check marks, different measurement initial ranges may be assigned to individual ports.

All unchecked ports will use the default value, set in the main GUI. This applies to regular measurement settings (RngSpc) as well as to reference sweep settings (RefSpc).

For regular measurements, you may also set individual stitching settings (number of scans and range decrement per scan).



Taking References

You can perform a reference measurement on a certain port by clicking the corresponding **New Ref.** button. You can do this for any number of ports, one after another. Each port that has been processed in this way will display the timestamp of the reference sweep, the wavelength range used, the port number that the reference was obtained at as well as the average power of the reference sweep on that port.

Reference sweeps will be performed using the wavelength range and laser power defined in the engine's main GUI. Power meters will be operated in the range set in the Power Meter Reference section of the engine's main GUI and the Port/Reference Manager. Auto ranging can be used for reference sweeps as well as port-specific range settings (See “Range Override on page 34” for details on how to use different ranges for different power meter ports.).

NOTE The average power should serve as a verification for the reference validity. Depending upon the TLS power selected in the GUI and the insertion loss of the reference connection, you should compare this value with your expectation of the average power.

You can perform a single reference sweep on multiple ports simultaneously by using the check boxes above the individual ports. All checked ports will be included in the reference sweep, once you click the **New Ref. (sel.)** button.

To obtain a reference on all ports simultaneously, click the **New ref. (all)** button. In this case, it doesn't matter which ports have been checked or unchecked.

Use the **Select All** and **Select None** buttons to set or remove all check marks simultaneously.

You can obtain reference data on one or more ports, then copy those traces to be used as reference for other ports as well or instead. To do so, obtain the actual reference sweep, then check all destination ports (make sure all other ports are unchecked) and click the **Cpy Frm** button at the source port.

All destination ports will show the detail information of the source port afterwards.

You can clear data for individual ports by clicking the **Clear** button of the respective port. By checking several ports and clicking the **Clear selected** button, reference data for those ports is cleared. By clicking the **Clear Ref.** button, the complete reference data is cleared from memory. Also upon restarting the engine, no reference will be loaded automatically, until a new reference has been saved.

NOTE Ports that don't display reference details (such as timestamp, etc.) will be referenced to the TLS power set in the Measurement Parameters section.

You can load, save and delete reference files by clicking the **Load Ref.**, **Save Ref.** and **Delete Ref.** buttons. By clicking either button, a list of reference files will be displayed to choose from.

NOTE

Since the reference data is obtained and required by the engine server, the files will be stored in a certain folder on the server PC, so no different folder may be selected.

This folder should be:

C:\ProgramData\Keysight\Photonic Application Suite\FSIL Application\References\

When uninstalling the PAS main package, you will be prompted whether to uninstall or to keep application data for this engine, such as, e.g., reference files.

Usually the last reference operation can be reverted by clicking the Undo button. Clicking this button will retrieve the reference state before the last reference sweep, clear, copy or load operation.

NOTE

Optimum performance is achieved if a reference measurement is performed with identical sweep parameters as subsequent measurements.

NOTE

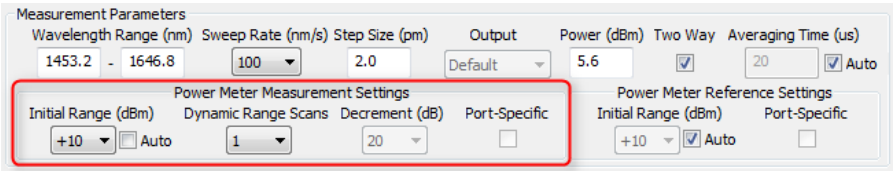
When exiting the application, the current reference is **not** stored automatically. However, the most recently saved reference will be loaded automatically, the next time the application is started.

Configuring Dynamic Range Related Settings

The Fast Spectral Loss measurement engine allows for performing swept wavelength measurements on devices with very high dynamic range.

By default, the engine is configured to automatically choose for each port the most sensitive power meter range, such that the input power does not exceed the maximum allowed at that range, thus avoiding data clipping. By default, it is also set to make a single dynamic range scan. That way, each time a measurement is performed, it will yield data subject to the power meter's dynamic range, depending on the type of power meter used and the averaging time selected (refer to the power meter's specifications for further details).

The dynamic range of the measurement can also be optimized by choosing the laser power so that the maximum power at the power meters is close to the maximum allowed at the power meter range setting. This is normal 3 dB higher than the nominal range value. For example, the power meter can measure up to +3 dBm when using the 0 dBm power range setting.



To increase the dynamic range of the measurement, the engine may perform multiple sweeps at different ranges, then combine the results (stitching). To use this feature, choose a number of dynamic range scans and a decrement value that determines the amount by which the power meter range should be changed between sweeps. For the N7744A or the N7745A, a typical range decrement would be 30-40dB, while for N7747A/N7748A instruments it would be 20-30dB, based on the single-sweep dynamic range of the models. Depending on the combination of power meter types involved and the initial ranges of each port, it is possible that the minimum range of a power meter is reached before the lowest range value defined by dynamic range scans and range decrement is reached. In that case, the power meter's minimum range is used instead. If the selected settings would lead to multiple sweeps without any change in power meter range on any port, there will be a warning message, suggesting correcting the settings to avoid any unnecessary scans.

You can use the Using the Port/Reference Manager on page 33 to define power range / dynamic range scan settings on a per-port basis. The respective ports will no longer be subject to the global settings configured in the engine's main GUI.

Please note that using the Auto range feature may also lead to different Initial Range settings for different power meter ports.

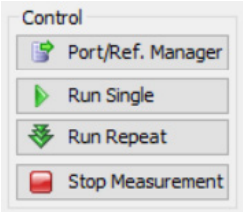
If any port is configured to use individual range settings, this is indicated by the Port-Specific check box in the engine's main GUI being checked. You can use the Port/Reference Manager to check and modify those settings or click on the Port-Specific check box to reset all individual settings, so that all ports use the global settings from the main GUI once again.

The measurement engine performs data synchronization with the wavelength scale, based on power meter type, range and averaging time. For optimum measurement results, the averaging time should be determined by the measurement engine, so the Auto check box next to the Averaging Time control should be checked.

Best dynamic range is achieved for high averaging times, while highest wavelength resolution may require low averaging times unless low sweep rates are used. If the Auto check box next to the Averaging Control is checked, the engine chooses the appropriate averaging time based on the selected wavelength step size. User higher step sizes for reduced noise and increased dynamic range.

All of the above applies to reference sweeps as well, except that those don't support stitching.

Performing a Measurement



Before starting a measurement, perform the following steps:

- Ensure that the Measurement Setup parameters suit your measurement task (see Application Setup Parameters on page 27).
- Check the **Port/Ref. Manager** to make sure that all the power meter ports you want to incorporate into the measurement are selected.
- In the Port/Reference Manager you can also perform reference measurements, that subsequent measurements are referenced to. Any port that has no reference data stored will be referenced to the TLS power set in the measurement parameters. Best performance will be achieved if reference data has been stored for each port.

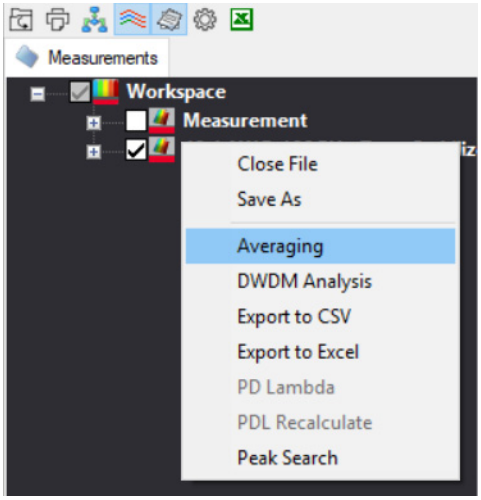
Use the measurement controls as described below:

- **Run Single:** You can perform single measurements by clicking Run Single.
- **Run Repeat:** By clicking Run Repeat the software will perform single measurements in a repeat mode.

- **Stop Measurement:** If measurements are being performed, you can abort the current measurement sweep and stop the repeat operation by clicking Stop Measurement. The operation is stopped, as soon as the current sweep data has been obtained and evaluated.

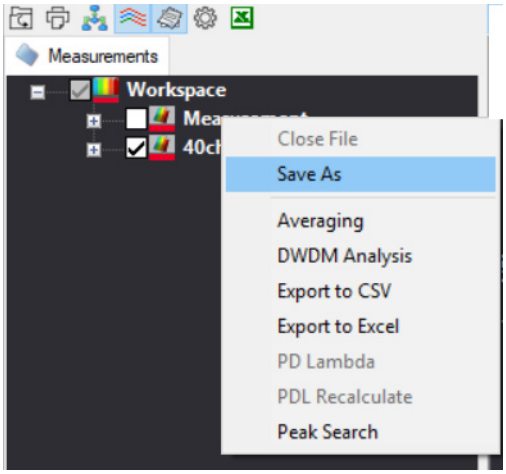
Applying Plugins

You can apply evaluation plugins to the current measurement, e.g. averaging, peak-search or DWDM channel analysis filters. The plugins are listed in the measurement context menu in the browser tree. Which plugins are available depends on the license(s) that are installed on your system. For more information about the available feature packages and licenses, refer to Keysight N7700 Photonic Application Suite - Brochure. Some plugins directly modify the traces of the measurement, some add more traces, some just add information to the table on the bottom of the user interface.



Saving Measurements

You can save measurement data using the **Save As** option from the measurement context menu in the browser tree and providing a filename.

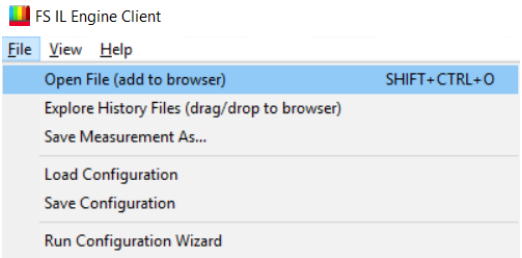


Alternatively, choose File > Save Measurement As.

Loading Measurements

Saved files can be accessed through the **Open File** dialog from the **File** menu, by pressing the **Open** icon above the browser tree. Open files will be listed in the browser tree to the left of the user interface.

You may also load one or more files using drag and drop, dropping OMR files anywhere on the engine GUI except for the measurement trace / graph regions.



Creating .agconfig Files from Measurement Files

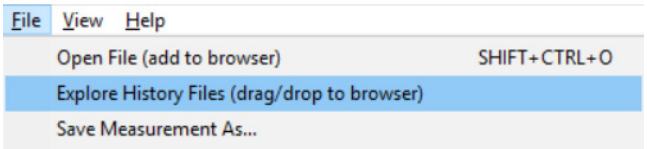
You might require the .agconfig file for a particular measurement that you have performed in the past. In case you did not explicitly save the .agconfig file, you can extract it from a measurement (.omr) file by using ExtractAgconfigFromOmr, a PowerShell utility, available in the Examples folder (C:\Program Files\Keysight\Photonic Application Suite\Examples\DemoPowerShell). This utility takes an OMR file as an input and creates an agconfig file that reflects the settings with which the specified measurement had been performed. The extracted .agconfig file has the same name as the specified OMR file and is created in the same directory as the OMR file. For easy access and execution of this PowerShell utility, use the batch file ExtractAgconfigFromOmr.bat specifying the OMR file as a parameter (either specify the full path of the OMR file or place it in the current working directory).

This utility can also be used for troubleshooting purposes.

Automatic History Saves

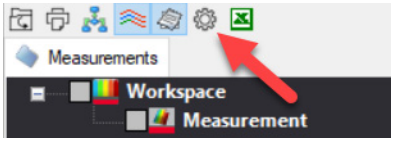
By default the measurement engine will automatically save a number of recent measurements to a specific folder. This is a very helpful feature, since one often performs a number of measurements that are not intended for saving, then realizes, after a certain, subsequent measurement, that comparison with one of the previous measurements would indeed reveal certain insights.

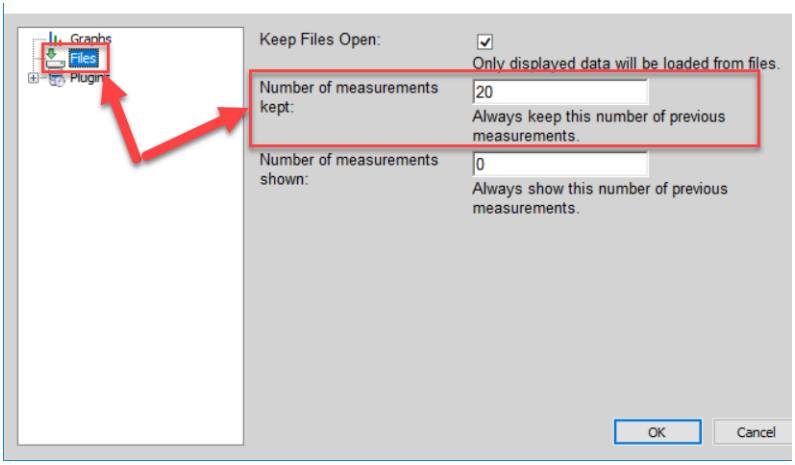
Selecting Explore History Files from the File menu will open this engine's history file save folder. The files are named using the engine acronym followed by a timestamp. Use drag and drop to add any file(s) to the engine's browser tree or double-click any file(s) for display in the File Viewer application.



Especially when working with large amounts of measurement data (many ports, broad range, high resolution), you may want to reduce the number of automatically stored files or even turn off the automatic saving of history files. On the other hand you may want to increase the number for obtaining many measurement files without having to perform manual saves, or having to use measurement automation.

You can do so by modifying the corresponding value in the options:

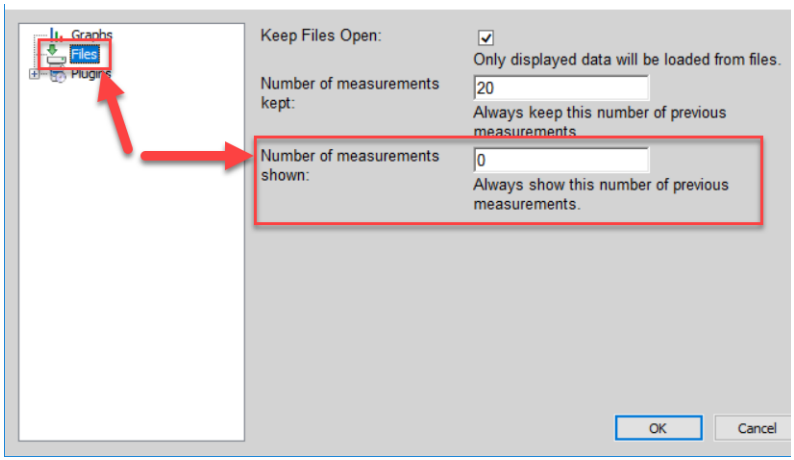




Set the **Number of measurements kept** value to 0 to turn off automatic saves.

When tuning a device or a component, or monitoring stability, it might be desirable to automatically have a certain number of most recent measurements displayed simultaneously.

To do so, set the **Number of measurements shown** value in the options to the desired value or to 0 to disable this feature.



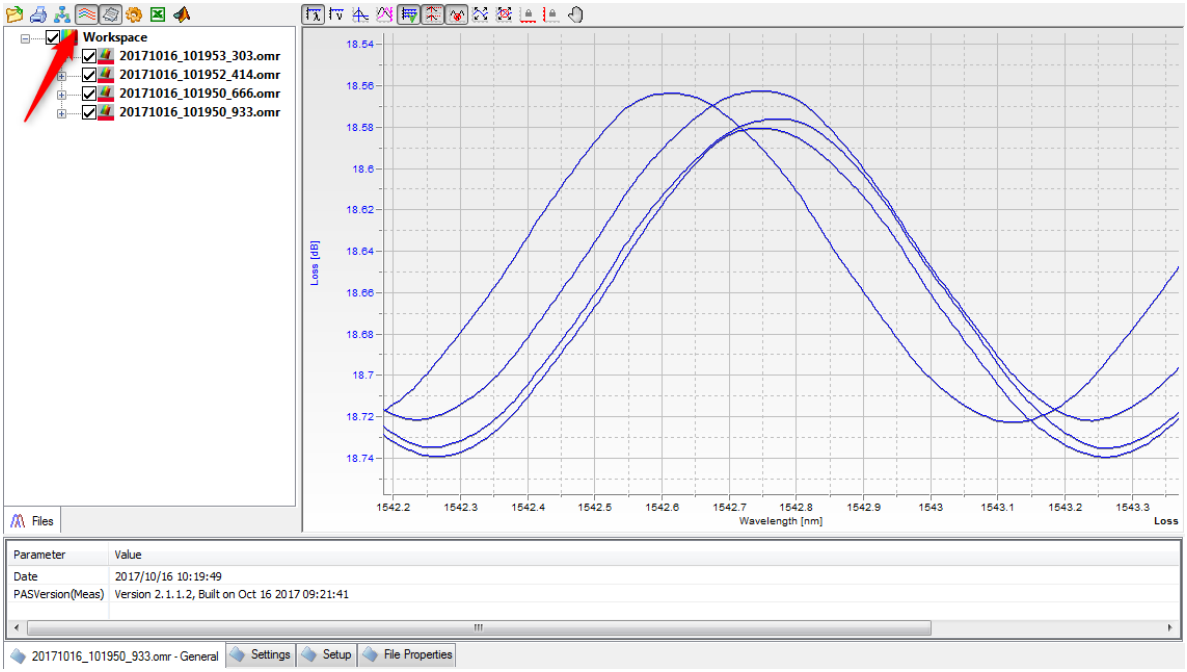
NOTE **Number of measurements kept** must be larger than or equal to **Number of measurements shown**, for this feature to work as intended.

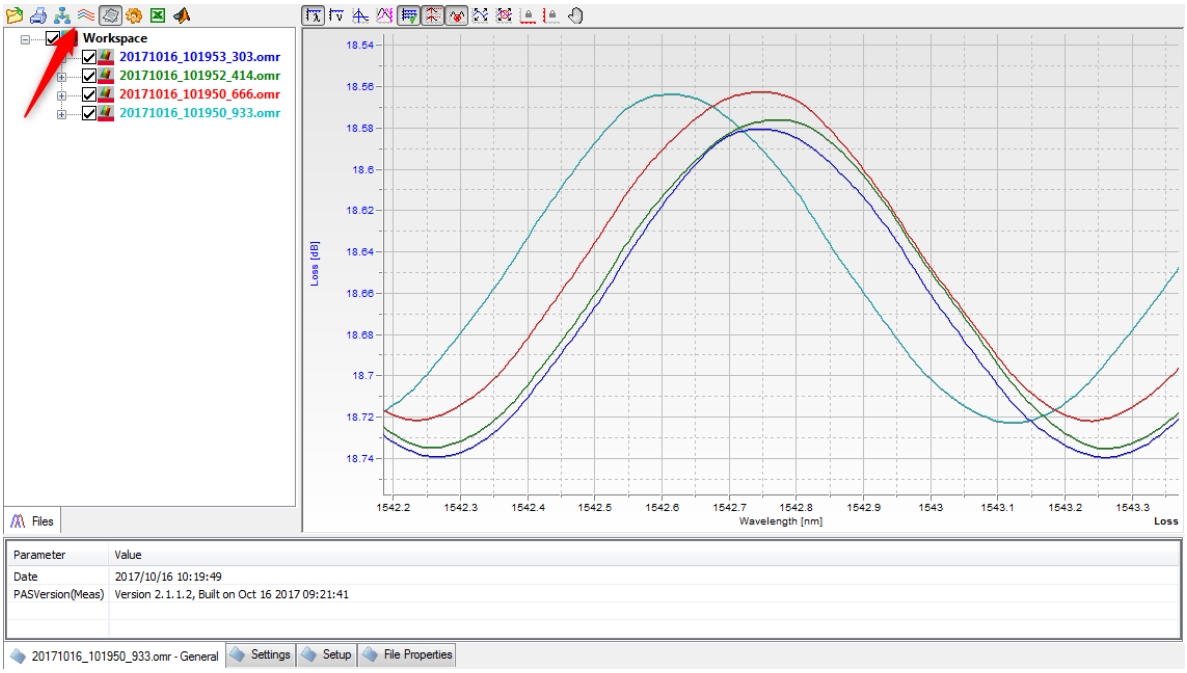
NOTE **Number of measurements kept** and **Number of measurements shown** are global settings used for all Photonic Application Suite engines.

NOTE The **Number of measurements shown** setting affects measurements performed after starting the engine, after changing this setting, or after closing all the open files. It does not open any older files, so after any of the mentioned operations, the number of automatically shown history files will grow with each new measurement until the total number reaches the **Number of measurements shown** value. For each new measurement after that, the oldest history file will be closed automatically.

By default, trace colors indicate different ports, but by toggling the **Color by Channel/File** icon, this can be changed to indicate different measurements instead. This is usually desired, when using automatic history file display with single-port devices, or devices, where individual ports are clearly distinguishable.

See examples for both modes below.





When either replacing the DUT, introducing a significant change to DUT tuning parameters, or running a specific path de-embedding measurement, corresponding measurement results may vary significantly from previous history files. To remove all the old measurements from the Workspace at once, click **Close All** from the context menu of the Workspace node in the tree on the left of the GUI.

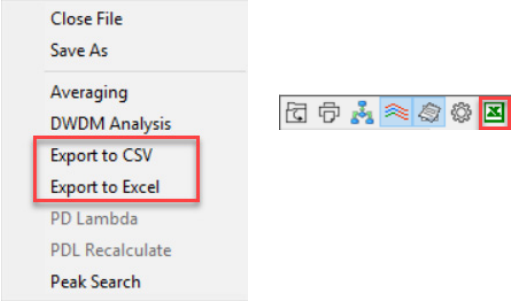
NOTE

History files are stored in an engine specific subfolder of

C:\Users\<username>\Documents\My Photonic Application Suite History\

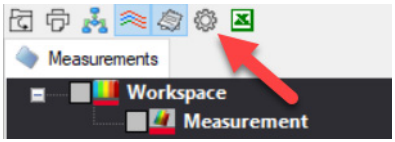
Exporting Measurement Data

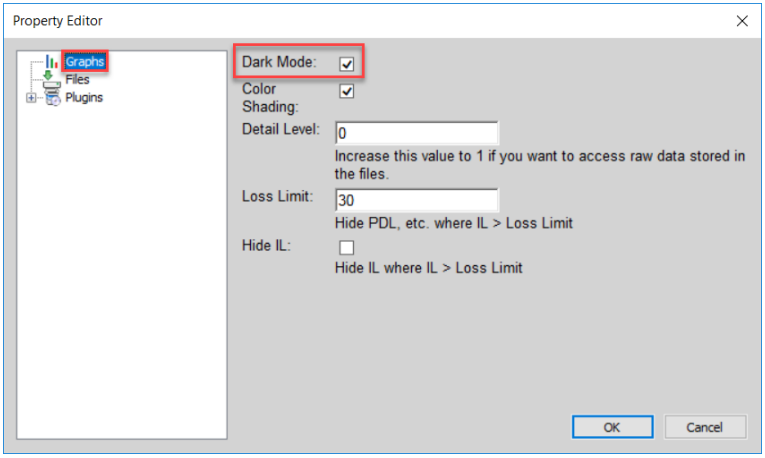
Measurement data can be exported to different applications or file types (e.g. Excel, CSV). You can export the current measurement by selecting the corresponding entry of the measurement context menu in the browser tree or the corresponding icon above the browser tree.



Changing the Color Theme

You can switch to a lighter color theme for the File Viewer within the user interface of Fast Spectral Loss Engine. Click the Settings icon and disable the Dark Mode check box in the Graphs section.



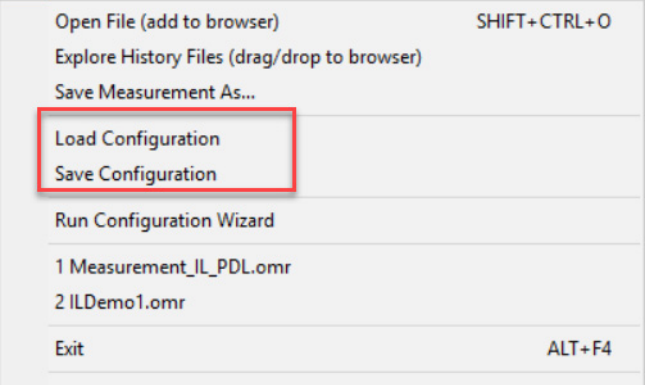


TLS Lambda Zeroing

In order to ensure optimum wavelength sweeping through varying ambient conditions, a lambda zeroing function is available. See Instrument Setup Parameters on page 31 for details. This helps to avoid missing points at the beginning or end of sweeps, due to an offset starting point, and helps evenly lubricate the full mechanical range.

Configuration Handling

The current configuration, i.e. attached hardware as well as current parameters like wavelength range, can be saved and loaded from the **File** menu by selecting **Save Configuration** and **Load Configuration**. A saved configuration file (*.agconfig) can also be started from the Windows Explorer and will launch the appropriate engine if it's not running.



The configuration wizard can be launched from the **File** menu as well.

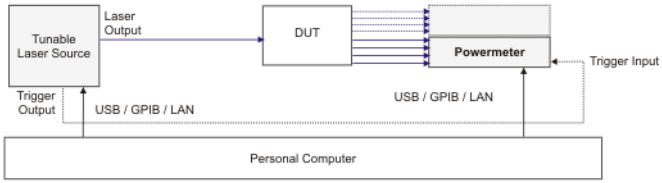
NOTE

If an agconfig file with identical hardware settings is loaded from within an activated engine, only the changed measurement parameters will be applied to the current engine. You can use this to quickly switch between different parameter sets.

If an agconfig file with different hardware settings is loaded instead, the current engine will be deactivated and the chosen agconfig file will be used to activate a new engine.

Measurement Setup

The general setup is illustrated in the following diagram:



NOTE Ensure that you perform the software installation steps described in Software Installation on page 18 before connecting the instruments.

- Connect the TLS mainframe and any power meters to the PC using the GPIB, USB or LAN interface.
- Connect the Trigger Output of the TLS with the Trigger Input of the power meters (using a 50 Ohms BNC cable). When using multiple power meters, you may connect the trigger cable from the TLS to any of the power meters, then connect the trigger output of that power meter to the trigger input of the next power meter. Similarly you may connect all remaining power meters.
- Connect the optical output of the TLS with the optical input of the DUT using a patchcord with appropriate connectors (angled / straight).
- Connect any optical output of the DUT to the optical inputs of the power meter using patchcords with appropriate connectors (angled / straight).

Keysight N7700 Photonic Application Suite

User's Guide

4 Automation

This chapter contains instructions to control the Fast Spectral Loss Measurement Engine using your own software.

KEYSIGHT TECHNOLOGIES

COM Components

Automation is implemented using a mechanism called “COM”. COM has been introduced by Microsoft on the Windows platform to allow a unified way to communicate between different software components. Today, almost every programming language such as C#, C++, LabView, Keysight Vee as well as MATLAB offers ways to use these so-called COM components. Once familiar with handling COM components, programming can be done using function browsers or auto-completion that show you the available properties and functions/methods including their parameter names. It is thus often possible to use the available functions without consulting the documentation.

Since the syntax of calling COM components is a little different in every programming language, we focus in our examples on MATLAB code. MATLAB has a very generic syntax and it should be easy to adapt the code to any other language. Further examples are included in the software distribution. These examples get installed with the Photonic Application Suite and can usually be found at the following location:

C:\Program Files\Keysight\Photonic Application Suite\Examples

Here is a simple example which starts a measurement (similar to clicking the “Run Single” button):

```
% Connect to Engine Manager
EngineMgr=actxserver('AgServerFSIL.EngineMgr');

% List all Engines currently running
EngineIDs=EngineMgr.EngineIDs;

% Always connect to first engine
Engine=EngineMgr.OpenEngine(EngineIDs(1));

% Start measurement
Engine.StartMeasurement;

% Release the engine and the engine manager
Engine.release;
EngineMgr.release;
```

This example assumes that you have already started the Client Software and thus a server is already running. Therefore, this example connects to an existing engine.

The next section explains how to create a new instance of the engine, in case the server has not yet been started.

Creating a New Engine

When writing your own software, it is very useful to start the Client Software first. Starting the Client Software will start the Server Software and creates an instance of the “Engine” COM component which handles the hardware communication. If your self-written software connects to this engine, you can see changes immediately in the client user interface. This is extremely useful in the debugging phase.

Finalizing your software, you probably don’t need the user interface anymore. In that case, you can start the server yourself and create an instance of the engine:

```
% Connect to Engine Manager
EngineMgr=actxserver('AgServerFSIL.EngineMgr');

% Create a new engine
Engine=EngineMgr.NewEngine;

% Load configuration file
Engine.LoadConfiguration(’C:\test.agconfig’);

% Activate engine
Engine.Activate;

% Start measurement
Engine.StartMeasurement;

% Deactivate engine
Engine.DeActivate;

% Release the engine and the engine manager
Engine.release;
EngineMgr.release;
```

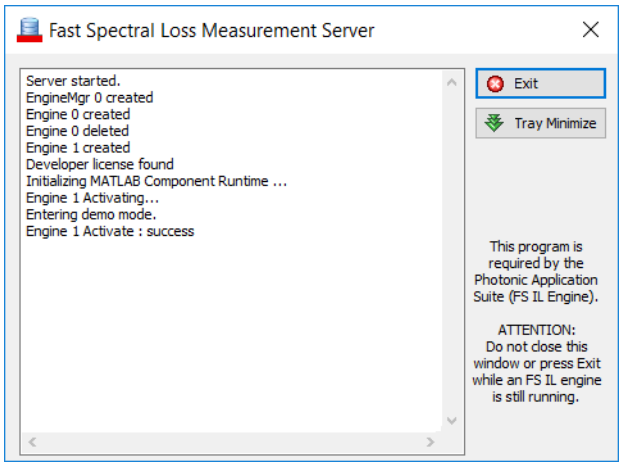
Note that you have to activate/deactivate the engine explicitly. If connecting to an existing engine, this is done by the client. When activating the engine, communication with the hardware is started.

We recommend to configure your engine initially using the Client Software and store the configuration using the “Save Configuration” menu. This example uses the “LoadConfiguration” command to load an .agconfig-file which has been stored before by the client.

NOTE

An icon appears in the Windows taskbar when the server has started (i.e. the engine manager has been invoked):

Left-click on the icon to bring the server window to the front. The server will display a protocol line when an engine has been created or deleted:



The above screenshot is for illustrative purpose only. The message log in your server console might look different from what is displayed above.

Connecting to an Existing Engine

Use the following code to connect to an existing engine:

```
% Connect to Engine Manager
EngineMgr=actxserver('AgServerFSIL.EngineMgr');

% List all Engines currently running
EngineIDs=EngineMgr.EngineIDs;

% Always connect to first engine
Engine=EngineMgr.OpenEngine(EngineIDs(1));

% Start measurement
Engine.StartMeasurement;

% Release the engine and the engine manager
Engine.release;
EngineMgr.release;
```

NOTE In most of the following MATLAB examples, we assume that the engine has been created and activated before.

Performing a Measurement

The following code starts a measurement and waits for the measurement to be finished. The result is stored as an OMR file.

```
% Start
Engine.StartMeasurement;

% Wait for measurement to be finished
while Engine.Busy; pause(1); end;

% Get result object
MeasurementResult = Engine.MeasurementResult;

% Save as OMR file
MeasurementResult.Write('c:\test.omr');

% Release measurement object
MeasurementResult.release;
```

Accessing the Measurement Result

The measurement result is returned as reference to a file object. This file COM object represents an OMR file and wraps all methods to read and write these files (Interface: IOMRFile).

NOTE Refer to the *N7700 Photonic Application Suite User's Guide* for a reference of the common interfaces, e.g. IOMRFile, IOMRGraph and IOMRProperty.

Once the measurement has finished, you can request a reference to this object and manipulate it. Typically you want to store the file somewhere on your hard disk or you want to access the graphs.

Saving the file can be done using the Write-method:

```
% Save as OMR file
MeasurementResult.Write('c:\test.omr');
```

Accessing the graph data of the spectral loss measurement is done like this:

```
% Access IL-graph
Graph = MeasurementResult.Graph('RXTXAvgIL');
noChannels      = Graph.noChannels;
dataPerCurve    = Graph.dataPerCurve;
YData = reshape(Graph.YData,dataPerCurve,noChannels);
xStart = Graph.xStart;
xStep = Graph.xStep;
```

The graphs in the file have names. The name "RXTXAvgIL" corresponds to the Loss measurement data.

The data are returned as a one-dimensional array, even if it contains more than one channel. The data for each channel are concatenated. The array starts with the first channel and continues with the next channels. The MATLAB command "reshape" converts this to a 2-dimensional array, a matrix. You can use the properties "noChannels" to get the total number of contained channels and the property "dataPerCurve" to find out how many values belong to a single channel.

NOTE

Using automation commands it is possible to start a new measurement engine as well as connect to an existing one, for example, one started from the Keysight Client GUI.

Certain automation operations, such as the Write method of the OMRFile interface or the FileSave method of the IEngine interface, lock the measurement object while accessing it. This might cause temporary failures in automation operations that are triggered from other instances connected to the same N7700 engine server.

This might especially be the case when using engine automation in addition to running the Keysight Client, which locks the OMRFile object in memory while updating measurement data in the GUI right after a measurement and when performing any automatic history save (see Automatic History Saves on page 42 to learn how to disable automatic saves).

To deal with this behavior, either implement retries or delays on affected operations or avoid connecting multiple instances to the same measurement engine server.

Reference: Interface “IEngineMgr”

The IEngineMgr interface is invoked using the following PROGID to identify the COM-server:

`AgServerFSIL.EngineMgr`

MATLAB:

```
EngineMgr=actxserver('AgServerFSIL.EngineMgr');
```

property Version

Type-Library:

```
get: HRESULT Version([out, retval] BSTR* pVal)
```

MATLAB:

```
Version=EngineMgr.Version;
```

Shows information about revision and build date of the engine.

method IsVersionGreaterOrEqual

Type-Library:

```
HRESULT IsVersionGreaterOrEqual([in] BSTR
VersionNumber,[out,retval] BOOL* pVal);
```

MATLAB:

```
Engine=EngineMgr.IsVersionGreaterOrEqual('1.1.0.1');
```

Performs a check, whether the engine revision is greater than or equal to the revision number provided as a parameter.

method NewEngine

Type-Library:

```
HRESULT NewEngine([out,retval] IEngine** aEngine);
```

MATLAB:

```
Engine=EngineMgr.NewEngine;
```

Tells the engine manager to create a new engine and returns the corresponding handle.

method OpenEngine

Type-Library:

```
HRESULT OpenEngine([in] LONG EngineID,[out,retval] IEngine**
aEngine);
```

MATLAB:

```
Engine=EngineMgr.OpenEngine(EngineID);
```

Returns the handle to an existing engine. The engine ID is a unique identifier which can be obtained by requesting the property “EngineIDs”.

property EngineIDs

Type-Library:

```
get: HRESULT EngineIDs([out, retval] SAFEARRAY(LONG)* pVal)
```

MATLAB:

```
EngineIDs=EngineMgr.EngineIDs;
```

Returns the IDs of all created engines.

method DeleteEngine

Type-Library:

```
HRESULT DeleteEngine([in] IEngine* aEngine);
```

MATLAB:

```
EngineMgr.DeleteEngine(Engine);
```

To ensure that all memory cleanup operations are executed properly, it is recommended to call the EngineMgr's DeleteEngine method with the current engine object as an argument after running the engine's DeActivate method and before releasing the engine object. After that, either create a new engine or release the EngineMgr object.

Reference: Interface “IEngine”

The IEngine interface is invoked either using the method “NewEngine” or “OpenEngine” of the IEngineMgr interface:

MATLAB:

```
Engine=EngineMgr.NewEngine;
```

property Version

Type-Library:

```
get: HRESULT Version([out, retval] BSTR* pVal)
```

MATLAB:

```
Version=Engine.Version;
```

Shows information about revision and build date of the engine.

method IsVersionGreaterOrEqual

Type-Library:

```
HRESULT IsVersionGreaterOrEqual([in] BSTR
VersionNumber,[out,retval] BOOL* pVal);
```

MATLAB:

```
Engine=Engine.IsVersionGreaterOrEqual('1.1.0.1');
```

Performs a check whether the engine revision is greater than or equal to the revision number provided as a parameter.

method Activate

Type-Library:

```
HRESULT Activate(void);
```

MATLAB:

```
Engine.Activate;
```

Activates the engine. Usually, a configuration file, created using the engine’s client software, should be loaded prior to activating the engine. This is done using the “LoadConfiguration” method. Activating an engine will cause the server to communicate with the instruments.

method DeActivate

Type-Library:

```
HRESULT DeActivate(void);
```

MATLAB:

```
Engine.DeActivate;
```

Deactivates the engine. You should deactivate the engine before loading a different configuration.

NOTE **After deactivating an engine and before releasing the engine object and its corresponding EngineMgr object, the EngineMgr's DeleteEngine method should be called. Refer to** method DeleteEngine on page 61 **for further details.**

---

property Active

Type-Library:

```
get: HRESULT Active([out, retval] BOOL* pVal);
```

MATLAB:

```
x = Engine.Active;
```

Returns 0/1 if the engine is inactive/active.

property Busy

Type-Library:

```
get: HRESULT Busy([out, retval] BOOL* pVal);
```

MATLAB:

```
x = Engine.Busy;
```

Returns 0/1 if the engine is not-busy/busy. The engine might be busy when transferring large amounts of data to or from the instruments or when it's waiting for a measurement operation to complete.

property EmulationMode

Type-Library:

```
get: HRESULT EmulationMode([out, retval] BOOL* pVal);
put: HRESULT EmulationMode([in] BOOL Val);
```

MATLAB:

```
b = Engine.EmulationMode;
```

```
Engine.EmulationMode = b;
```

Sets or gets the emulation (demo) mode setting. If operated in demo mode, no actual communication with instruments will be performed and simulated measurement results will be shown instead.

method LoadConfiguration

Type-Library:

```
HRESULT LoadConfiguration([in] BSTR Filename);
```

MATLAB:

```
Engine.LoadConfiguration('c:\\test.agconfig');
```

Loads a configuration file (extension: .agconfig). The engine should be inactive while loading a new config file, i.e., either before calling Activate or after calling Deactivate.

property Configuration

Type-Library:

```
get: HRESULT Configuration([out, retval] BSTR* ConfigXML);
```

```
put: HRESULT Configuration([in] BSTR ConfigXML);
```

MATLAB:

```
xml = Engine.Configuration;
```

```
Engine.Configuration = xml;
```

Transfers the contents of an .agconfig-file to or from the engine. An .agconfig-file contains XML formatted text.

property SoftwareConfiguration

Type-Library:

```
put: HRESULT SoftwareConfiguration([in] BSTR ConfigXML);
```

MATLAB:

```
Engine.SoftwareConfiguration = xml;
```

Transfers only the software section contents of an .agconfig-file to the engine. An .agconfig-file contains XML formatted text. This can be used to change all measurements parameters at once, without changing any hardware settings or having to reactivate the engine.

property KeepRawData

Type- Library:

```
get: HRESULT KeepRawData([out, retval] BOOL* pVal);
put: HRESULT KeepRawData([in] BOOL Val);
```

MATLAB:

```
b = Engine.KeepRawData;
Engine.KeepRawData = b;
```

Defines whether stored measurement files will contain measurement raw data or not.

property WavelengthStart

Type-Library:

```
get: HRESULT WavelengthStart([out, retval] DOUBLE* pVal);
put: HRESULT WavelengthStart([in] DOUBLE Val);
```

MATLAB:

```
d = Engine.WavelengthStart;
Engine.WavelengthStart = d;
```

Sets or gets the start wavelength in nm.

property WavelengthStop

Type-Library:

```
get: HRESULT WavelengthStop([out, retval] DOUBLE* pVal);
put: HRESULT WavelengthStop([in] DOUBLE Val);
```

MATLAB:

```
d = Engine.WavelengthStop;
Engine.WavelengthStop = d;
```

Sets or gets the stop wavelength in nm.

property WavelengthStep

Type-Library:

```
get: HRESULT WavelengthStep([out, retval] DOUBLE* pVal);
```

```
put: HRESULT WavelengthStep([in] DOUBLE Val);
```

MATLAB:

```
d = Engine.WavelengthStep;
```

```
Engine.WavelengthStep = d;
```

Sets or gets the wavelength step in pm. The TLS will perform continuous sweeps, but the measurement results will be returned at wavelengths defined by WavelengthStart, WavelengthStop and WavelengthStep.

property SweepRates

Type-Library:

```
get: HRESULT SweepRates([out, retval] SAFEARRAY(double)*
pVal);
```

MATLAB:

```
dArray = SweepRates;
```

Returns a double array containing the sweep rates in nm/s that the TLS can sweep at.

property SweepRate

Type-Library:

```
get: HRESULT SweepRate([out, retval] DOUBLE* pVal);
```

```
put: HRESULT SweepRate([in] DOUBLE Val);
```

MATLAB:

```
d = Engine.SweepRate;
```

```
Engine.SweepRate = d;
```

Sets or gets the sweep rate in nm/s.

property UseTwoWay

Type-Library:

```
get: HRESULT UseTwoWay([out, retval] BOOL* pVal);
```

```
put: HRESULT UseTwoWay([in] BOOL Val);
```

MATLAB:

```
b= Engine.UseTwoWay;
```

```
Engine.UseTwoWay = b;
```

If set to true, forward and reverse TLS sweep direction are used for evaluation. If set to false, only the forward sweep direction is used. Two-way operation is supported by 81606A, 81607A, and 81960A tunable laser sources only.

property SupportsTwoWay

Type- Library:

```
get: HRESULT SupportsTwoWay([out, retval] BOOL* pVal);
```

MATLAB:

```
b= Engine.UseTwoWay;
```

If true, currently configured tunable laser source supports forward and reverse TLS sweep direction. If false, only the forward sweep direction is supported. Two-way operation is supported by 81606A, 81607A and 81960A tunable laser sources only.

property PWMAutoRanging

Type- Library:

```
get: HRESULT PWMAutoRanging([out, retval] BOOL* pVal);
```

```
put: HRESULT PWMAutoRanging([in] BOOL Val);
```

MATLAB:

```
b= Engine.PWMAutoRanging;
```

```
Engine.PWMAutoRanging= b;
```

If set to true, the measurement engine will choose appropriate power ranges for each port automatically during regular measurements. If set to false, the power range set by the property PWMSensitivity will be used.

Please note that any port-specific settings will no longer be affected by the global PWMSensitivity setting.

property NumberOfScans

Type- Library:

```
get: HRESULT NumberOfScans([out, retval] LONG* pVal);
```

```
put: HRESULT NumberOfScans([in] LONG Val);
```

MATLAB:

```
i = Engine.NumberOfScans;
```

```
Engine.NumberOfScans= i;
```

Sets or gets the number of dynamic range scans to be performed. Use the property RangeDecrement on page 68 to define the step size by which the power meter range setting will be changed in each dynamic range scan.

Please note that any port-specific settings will no longer be affected by the global NumberOfScans setting.

property RangeDecrement

Type- Library:

```
get: HRESULT RangeDecrement([out, retval] DOUBLE* pVal);
```

```
put: HRESULT RangeDecrement([in] DOUBLEVal);
```

MATLAB:

```
d = Engine.RangeDecrement;
```

```
Engine.RangeDecrement= d;
```

Sets or gets the step size by which the power meter range setting will be changed in each dynamic range scan (in multiples of 10dB). Use the property NumberOfScans on page 67 to define the number of such scans to be performed in each measurement.

Please note that any port-specific settings will no longer be affected by the global NumberOfScans setting.

property PWMSensitivityRef

Type- Library:

```
get: HRESULT PWMSensitivityRef([out, retval] DOUBLE* pVal);
```

```
put: HRESULT PWMSensitivityRef([in] DOUBLE Val);
```

MATLAB:

```
d = Engine.PWMSensitivityRef;
```

```
Engine.PWMSensitivityRef= d;
```

Sets or gets the power meter sensitivity for the reference sweep in dBm. This property has to be set in multiples of 10dBm (- 20dBm to +10dBm).

property PWMAutoRangingRef

Type- Library:

```
get: HRESULT PWMAutoRangingRef([out, retval] BOOL* pVal);
put: HRESULT PWMAutoRangingRef([in] BOOL Val);
```

MATLAB:

```
b= Engine.PWMAutoRangingRef;
Engine.PWMAutoRangingRef= b;
```

If set to true, the measurement engine will choose appropriate power ranges for each port automatically during reference sweeps. If set to false, the power range set by the property PWMSensitivityRef on page 68 will be used.

Please note that any port-specific settings will no longer be affected by the global PWMSensitivityRef setting.

property PWMSensitivity

Type-Library:

```
get: HRESULT PWMSensitivity([out, retval] DOUBLE* pVal);
put: HRESULT PWMSensitivity([in] DOUBLE Val);
```

MATLAB:

```
d = Engine.PWMSensitivity;
Engine.PWMSensitivity = d;
```

Sets or gets the power meter sensitivity for the DUT sweep in dBm. This property has to be set in multiples of 10dBm (-20dBm to +10dBm).

property PWMAvgTime

Type-Library:

```
get: HRESULT PWMAvgTime([out, retval] DOUBLE* pVal);
put: HRESULT PWMAvgTime([in] DOUBLE Val);
```

MATLAB:

```
d = Engine.PWMAvgTime;
Engine.PWMAvgTime = d;
```

Sets or gets the power meter averaging time in us. This property has to be set in multiples of 1us with a minimum of 1us.

property PWMAvgTimeAuto

Type- Library:

```
get: HRESULT PWMAvgTimeAuto([out, retval] BOOL* pVal);
put: HRESULT PWMAvgTimeAuto([in] BOOL Val);
```

MATLAB:

```
b= Engine.PWMAvgTimeAuto;
Engine.PWMAvgTimeAuto= b;
```

If set to true, the measurement engine will choose appropriate averaging times for each port automatically during regular measurements. If set to false, the averaging time set by the property PWMAvgTime will be used.

method ZeroPWMChannels

Type-Library:

```
HRESULT ZeroPWMChannels([in] SAFEARRAY(LONG) Ports);
```

MATLAB:

```
Engine.ZeroPWMChannels(int32([3;6]));
```

Performs a zeroing operation on the selected power meter ports. As a parameter an integer array of port numbers (starting at 0 for port 1) must be provided. There will be a subsequent user input request to make sure that the corresponding ports are covered.

NOTE

When using this method from MATLAB, you need to pass a column vector.

Also, due to the way the parameter is passed from MATLAB to the engine, it is not possible to pass single-element arrays. Pass a line array instead, with a dummy value as a second element, e.g., Engine.MethodName(int32([3 3)).

---

property TLSOutputPort

Type-Library:

```
get: HRESULT TLSOutputPort([out, retval] LONG* pVal);
put: HRESULT TLSOutputPort([in] LONG Val);
```

MATLAB:

```
i = Engine.TLSOutputPort;
Engine.TLSOutputPort = i;
```

Sets or gets the output port to be used for the TLS. This integer value refers to the available Output port configuration list obtained with `OutputPorts`.

property TLSPower

Type-Library:

```
get: HRESULT TLSPower([out, retval] DOUBLE* pVal);
put: HRESULT TLSPower([in] DOUBLE Val);
```

MATLAB:

```
d = Engine.TLSPower;
Engine.TLSPower = d;
```

Sets or gets the laser power in dBm.

property LambdaZeroingMode

Type-Library:

```
get: HRESULT LambdaZeroingMode([out, retval] LONG* pVal);
put: HRESULT LambdaZeroingMode([in] LONG Val);
```

MATLAB:

```
i= Engine.LambdaZeroingMode;
Engine.LambdaZeroingMode = i;
```

Sets or gets the behavior when a lambda zeroing operation is suggested by the software. Set to 0/1/2 for Automatically/Always Ask/Manual Only. Refer to Instrument Setup Parameters on page 31 for further details.

method ZeroAllTLS

Type-Library:

```
HRESULT ZeroAllTLS(void);
```

MATLAB:

```
Engine.ZeroAllTLS;
```

Performs a lambda zeroing operation on the tunable laser source.

The name of the method has been chosen to keep it consistent to the IL/PDL engine, even though there is only a single TLS used in this application.

method StartCustomReference

Type-Library:

```
HRESULT StartCustomReference([in] SAFEARRAY(LONG) Ports);
```

MATLAB:

```
Engine.StartCustomReference(int32([3;6]));
```

Starts a reference measurement. As a parameter an integer array of port numbers (starting at 0 for port 1) must be provided. A reference sweep will be performed and reference data for all provided ports will be stored.

NOTE

When using this method from MATLAB, you need to pass a column vector.

Also, due to the way the parameter is passed from MATLAB to the engine, it is not possible to pass single-element arrays. Pass a line array instead, with a dummy value as a second element, e.g., Engine.MethodName(int32([3 3)).

---

method ClearReferenceChannels

Type-Library:

```
HRESULT ClearReferenceChannels([in] SAFEARRAY(LONG) Ports);
```

MATLAB:

```
Engine.ClearReferenceChannels(int32([3;6]));
```

Clears reference data from the currently loaded reference file. The saved file will not be changed by this operation. To update the saved reference file, the modified reference needs to be saved with the same filename. As a parameter an integer array of port numbers (starting at 0 for port 1) must be provided.

NOTE

When using this method from MATLAB, you need to pass a column vector.

Also, due to the way the parameter is passed from MATLAB to the engine, it is not possible to pass single-element arrays. Pass a line array instead, with a dummy value as a second element, e.g., Engine.MethodName(int32([3 3)).

---

method CopyReferenceChannel

Type-Library:

```
HRESULT CopyReferenceChannel([in] SourcePort, [in]
SAFEARRAY(LONG) TargetPorts);
```

MATLAB:

```
Engine.CopyReferenceChannel(5, int32([3 6]));
```

Copies reference data from the source port to all target ports in the currently loaded reference file. The saved file will not be changed by this operation. To update the saved reference file, the modified reference needs to be saved with the same filename. As parameters, the source port of the copy operation and an integer array of the target port numbers (both parameters starting at 0 for port 1) must be provided.

NOTE

When using this method from MATLAB, you need to pass a column vector.

Also, due to the way the parameter is passed from MATLAB to the engine, it is not possible to pass single-element arrays. Pass a line array instead, with a dummy value as a second element, e.g., Engine.MethodName(int32([3 3)).

---

method UndoReferenceOperation

Type-Library:

```
HRESULT UndoReferenceOperation(void);
```

MATLAB:

```
Engine.UndoReferenceOperation;
```

Reverts the currently loaded reference to the state prior to the last reference operation. The saved file will not be changed by this operation. To update the saved reference file, the modified reference needs to be saved with the same filename.

method LoadReference

Type-Library:

```
HRESULT LoadReference([in] BSTR Filename);
```

MATLAB:

```
Engine.LoadReference('Reference.omr');
```

Loads a reference file in OMR format for the current TLS.

Note that the file must be present on the computer running the server, not the client. Use `Engine.ReferenceList` to get a list of reference files present on the server.

method SaveReference

Type-Library:

```
HRESULT SaveReference([in] BSTR Filename, [in] BOOL
Overwrite);
```

MATLAB:

```
Engine.SaveReference('Reference.omr', 1);
```

Saves the current reference to an omr file. Will not overwrite an existing file unless Overwrite parameter is set to 1.

Note that the file will be stored on the computer running the server, not the client.

property ReferenceList

Type-Library:

```
get: HRESULT ReferenceList([out, retval] BSTR* pVal);
```

MATLAB:

```
s = Engine.ReferenceList;
```

Returns an xml string containing the filenames and important parameters for all reference files available on the engine server.

method DeleteReferenceFile

Type-Library:

```
HRESULT DeleteReferenceFile([in] BSTR Filename);
```

MATLAB:

```
Engine.DeleteReferenceFile('Reference.pbin');
```

Deletes a reference file on the engine server. Use `Engine.ReferenceList` to get a list of reference files present on the server.

property ChannelSetup

Type-Library:

```
get: HRESULT ChannelSetup([out, retval] BSTR* ChannelXML);
put: HRESULT ChannelSetup([in] BSTR ChannelXML);
```

MATLAB:

```
xml = Engine.ChannelSetup;
Engine.ChannelSetup = xml;
```

Sets or gets an xml string, containing information about the currently loaded reference data per port (XML tag RefAvailText).

Can be used to modify port specific settings, such as whether to include a port in a measurement or to configure or disable specific power range / stitching settings for certain ports.

Contains additional information for internal use.

method StartMeasurement

Type-Library:

```
HRESULT StartMeasurement(void);
```

MATLAB:

```
Engine.StartMeasurement;
```

Starts a measurement.

method StartMeasurementRepeat

Type-Library:

```
HRESULT StartMeasurementRepeat(void);
```

MATLAB:

```
Engine.StartMeasurementRepeat;
```

Starts a series of measurements. Will stop after StopMeasurement method has been called. Will not save any measurement data automatically, except for any automatic history saves that might have been configured in an engine client (FSIL Engine GUI), connected to the same engine server.

method StopMeasurement

Type-Library:

```
HRESULT StopMeasurement(void);
```

MATLAB:

```
Engine.StopMeasurement;
```

Stops a measurement at the next possible time. This mostly means, between measurements, such as in repeat measurement mode.

method FileSave

Type-Library:

```
HRESULT FileSave([in] BSTR Filename);
```

MATLAB:

```
Engine.FileSave('test.omr');
```

Saves the measurement result to an OMR file at the location provided as parameter.

See the last Note in Accessing the Measurement Result on page 58.

property MeasurementResult

Type-Library:

```
HRESULT MeasurementResult([out, retval] IOMRFile** pVal);
```

MATLAB:

```
Result = Engine.MeasurementResult;
```

Returns a reference to the underlying IOMRFile object which contains the measurement result.

NOTE Refer to the *N7700 Photonic Application Suite User's Guide* for a reference of the common interfaces, e.g. IOMRFile, IOMRGraph and IOMRProperty.

See the last Note in "Accessing the Measurement Result".

property EventPropertiesChanged

Type-Library:

```
get: HRESULT EventPropertiesChanged([out, retval] LONG*
pVal);
```

MATLAB:

```
nEvent = Engine.EventPropertiesChanged;
```

This property is a 32 bit integer value. The value will be increased whenever the state of the engine has changed. By polling this value and observing the changes, a client can be notified of any change of the engine state.

property EventMeasurementFinished

Type-Library:

```
get: HRESULT EventMeasurementFinished([out, retval] LONG*
pVal);
```

MATLAB:

```
nEvent = Engine.EventMeasurementFinished;
```

This property is a 32 bit integer value. The value will be increased whenever the engine has finished a measurement. By polling this value and observing the changes, a client can be notified of any change of the engine state.

property EventReferenceOperationFinished

Type-Library:

```
get: HRESULT EventReferenceOperationFinished([out, retval]
LONG* pVal);
```

MATLAB:

```
nEvent = Engine.EventReferenceOperationFinished;
```

This property is a 32 bit integer value. The value will be increased whenever the engine has finished a reference operation, e.g. performing a reference sweep, deleting the reference data for certain ports, etc. By polling this value and observing the changes, a client can be notified of any change of the engine state.

property EventDataAcquisitionFinished

Type-Library:

```
get: HRESULT EventDataAcquisitionFinished([out, retval]
LONG* pVal);
```

MATLAB:

```
nEvent = Engine.EventDataAcquisitionFinished;
```

This property is a 32 bit integer value. The value will be increased whenever the engine has finished a laser sweep and the corresponding data acquisition. The DUT can safely be reconfigured for the next measurement after this event has occurred. By polling this value and observing the changes, a client can be notified of any change of the engine state.

property ProtocolText

Type-Library:

```
get: HRESULT ProtocolText([out, retval] BSTR* pVal)
```

MATLAB:

```
ProtocolText = Engine.ProtocolText;
```

Returns a string containing all engine messages separated by a '|' character. If the number of messages exceeds 1000, the oldest message is deleted from the list for each new message.

property ProtocolMin

Type-Library:

```
get: HRESULT ProtocolMin([out, retval] LONG* pVal)
```

MATLAB:

```
First = Engine.ProtocolMin;
```

Returns the index to the first entry in the message list.

property ProtocolMax

Type-Library:

```
get: HRESULT ProtocolMax([out, retval] LONG* pVal)
```

MATLAB:

```
Last = Engine.ProtocolMax;
```

Returns the index to the most recent entry in the message list.

method GetProtocolTextAt

Type-Library:

```
HRESULT GetProtocolTextAt([in] LONG nMin,[in] LONG nMax,
[out,retval] BSTR* pVal);
```

MATLAB:

```
s = Engine.GetProtocolTextAt(Engine.ProtocolMin,
Engine.ProtocolMax);
```

Returns a portion of the message history defined by nMin and nMax.

property UserInputWaiting

Type-Library:

```
get: HRESULT UserInputWaiting([out, retval] BOOL* pVal);
```

```
put: HRESULT UserInputWaiting([in] BOOL Val);
```

MATLAB:

```
b = Engine.UserInputWaiting;
```

```
Engine.UserInputWaiting = b;
```

Returns 1 if the engine is waiting for a user input, e.g. to choose between different options or to acknowledge an error message. Returns 0 otherwise. Set to 0 after you have handled the user input (see method UserInputResponse on page 80)

property UserInputPrompt

Type-Library:

```
get: HRESULT UserInputPrompt([out, retval] BSTR* pVal)
```

MATLAB:

```
Prompt = Engine.UserInputPrompt;
```

Returns a string containing the prompt of the current user input request, if any.

property UserInputChoice

Type-Library:

```
get: HRESULT UserInputChoice([out, retval] BSTR* pVal)
```

MATLAB:

```
Choices = Engine.UserInputChoice;
```

Returns a string containing all possible responses to the user input requests separated by a '|' character. Each response contains a corresponding number to be used by `Engine.UserInputResponse` and the description of the response, e.g. for labeling a button. Number and description are separated by a ',' character.

method UserInputResponse

Type-Library:

```
HRESULT UserInputResponse([in] LONG nResponse);
```

MATLAB:

```
Engine.UserInputResponse(1);
```

Set the response to a user input request. Must be a number from the list obtained by `Engine.UserInputChoice`.
Set
`Engine.UserInputWaiting` to 0 after setting
`Engine.UserInputResponse` for the engine to continue operation.

Alphabetical Automation Index

Keysight N7700 Photonic Application Suite

User's Guide

5 Troubleshooting

KEYSIGHT TECHNOLOGIES

Symptoms and Solutions

No measurement data shown

Possible Reason	Solution
An error occurred during the measurement, preventing the software from fully evaluating the data.	Check the output in the Measurement Status box for errors.
The trace contains invalid data only.	Under certain circumstances (e.g. certain overrange conditions, or bad/outdated power meter or TLS zeroing data, certain regions may contain NaN values instead of reasonable data. NaN values are not displayed in the engine GUIs / File Viewer.
The corresponding port has its check box unchecked in the Port/Ref. Manager.	Check Port/Ref. Manager.
The specific graph type is not shown in the graph area of the engine GUI / File Viewer.	Especially when switching between different PAS engines or using the File Viewer for viewing data from different PAS engines, often different graph types are selected to be shown in the graph area. This is done by expanding the measurement in the browser tree, right-clicking the corresponding data type (e.g. Insertion Loss or PDL), then clicking **View Graph**. Usually unused graphs are removed from the graph area then, by right-clicking into one of the graphs and clicking **Remove Graph**. This view configuration is shared by the File Viewer and the engine GUIs, so it can happen that the graph types used by the current engine have been removed from the graph area from within another engine. See the N7700 User's Guide for further details.

Measurement Status Box not visible

Possible Reason	Solution
Window minimized.	If you minimized the Measurement Status window, it will stay that way until a user input is required. You can get it back to front anytime by clicking its entry in the task bar.
Window in background.	Unless “always on top” is checked, the measurement status box may get behind other windows. You can get it back to front by clicking its entry in the task bar.
Window closed.	If you closed the status window, it will stay that way until a user input is required. You can get it back by selecting Measurement Status in the View menu.
Window position stored in screen region that is currently invisible.	This could happen when working with multiple screens or varying screen resolutions. By selecting Reset Status Window Position in the View menu, the measurement status box position will be reset to the current top left corner of the engine window.

Graphs are flickering when using high measurement repetition rates

Possible Reason	Solution
Usually the graph x and y range are adjusted automatically when new measurement data is available. Even slight differences in the minimum / maximum values of the new trace compared to the previous trace may require an update of the axes then.	If the graph x and y range is such that all relevant features of the measurement are visible and the same can be assumed for subsequent measurements, use the **Lock X-Range** and **Lock Y-Range** buttons. This will freeze the x and y range, so no redraw of the axes themselves will occur and flickering should be reduced significantly.

The list of values for a drop-down control appears invisible or displaced

Possible Reason	Solution
Different scaling factors for the different screens in some multi-screen setups.	Ensure that the scaling factor for all the screens is consistent.
	Consider using a single screen setup, define another screen as the default screen, or start the engine from an Explorer window on the screen that the engine GUI is intended to show on.
The engine window is maximized.	To make the invisible list of values appear on the screen, restore the window to its normal size ensuring that it is not too close to the top left edge of the screen. Then set the required value in the drop-down control. Note that the list of values will still appear displaced.
	If the list of values is visible (although displaced), select the drop-down list and use the cursor (arrow) and Enter keys to navigate to and select the required value.

COM API registration fails

Possible Reason	Solution
MATLAB Compiler Runtime (MCR) has not been installed prior to measurement engine installation.	Register the COM API after later MCR installation or re-register for any other reasons, by running the **Re-register all N7700 COM APIs** program from the Start menu.

NOTE

A troubleshooting utility called **Get N7700 System Information** has been introduced to help Keysight technical support gather some vital details required to resolve problems with the software at the user end, engine startup problems, for instance. This utility is accessible from the Start menu and should be run to capture information that can be shared with Keysight technical support.

Edition 12.0, August 2021

N7700-91102

www.keysight.com

KEYSIGHT
TECHNOLOGIES