Qualcomm Technologies, Inc.

Qualcomm Sequence Profiling Resource (QSPR) User Guide

80-VB987-1 YA

December 1, 2016

Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

Revision history

Revision	Date	Description
A	January 2011	Initial release
B	August 2011	Numerous changes were made in this document. It should be read in its entirety.
C	December 2011	Changed Table 1-1; added Section 6.2.3; added details in Section 3.7.7
D	March 2012	Added information about Context Menu options in Debug windows, DLL Search Path changes, Show Parameter details, and Output parameters setup for scripts
E	April 2012	Added Section 1.3; updated Section 1.4; updated Figure 3-42; updated Section 3.10.2; updated Section 6.7; updated Section 8.1; updated Section 10.1
F	June 2012	Updated Sections 4.9 and 6.22, and Figure 6-2; added Sections 4.6.10 and 6.3
G	September 2012	Updated Sections 3.10, 4.3, 4.6, 5.1–5.4, 6.2, 6.3, 6.5, 6.7, and 10.1
H	January 2013	Updated Figure 3-1 and Figure 3-21; added Section 3.4.1, Chapter 10, and Chapter 12
J	March 2013	Updated Sections 3.2.2, 3.7.7, and 12.1; added Chapter 9
K	May 2013	Updated Sections 3.1.1.1.3 and 3.10.3 and Figure 3-45; added Figure 3-11
L	July 2013	Updated Section 3.2.2 and Figure 3-18
M	September 2013	Updated Sections 3.2.2, 3.10.2, 6.4.1, and 6.4.2, and Figures 3-44, 6-2, 6-5, and 6-11
N	November 2013	Updated Section 3.3.1 and Figures 4-11 and 5-4; added Sections 4.1.5, 4.3.2, 4.6, 5.2.4, 5.3.5, and 5.4.2.5
P	April 2014	Updated Sections 3.10.2 and 6.2.2 and Figures 6-2, 6-8, 6-11, 6-24, and 6-25
R	July 2014	Updated Figure 3-18 and Figure 6-2; added Section 6.4.3
T	January 2015	Updated Sections 3.4, 3.4.1, 3.5, 4.7.1, 4.7.5, 5.4 and 6.4.3.1
U	April 2015	Updated Chapters 6 and 10 and Figures 3-36 and 6-1
V	September 2015	Updated Chapter 12, Section 4.9, Figures 4.16, 4.18, 6-1, 6-25, 6-26, and 6-33
W	January 2016	Updated graphics throughout to reflect QSPR UI update
Y	April 2016	Numerous changes were made to this document, it should be read in its entirety.
YA	December 2016	Numerous changes were made to this document, it should be read in its entirety.

Note: There is no Rev. I, O, Q, S, X, or Z per Mil. standards.

Contents

1 Introduction

1.1 Purpose

This document discusses how to use the Qualcomm Sequence Profiling Resource (QSPR) application. QSPR is a test executive for running tests and collecting test data. The application is written in C# using .Net framework 4.0. The user machine should have, at a minimum, the .Net framework 4.0 Client Profile installed for running the QSPR application.

1.2 System requirements

QSPR runs on a Windows PC, XP or later. The hardware and memory requirements vary based on the test trees that are loaded. A PC with a 1 GHz single core processor with 512 MB of RAM is sufficient to load the QSPR application and run the sample trees provided by the QSPR installer.

When QSPR is installed, a set of sample files are installed in the location c:\Qualcomm\QSPR\Samples folder.

1.3 Conventions

Function declarations, function names, type declarations, attributes, and code samples appear in a different font, for example, `#include`.

Code variables appear in angle brackets, for example, `<number>`.

Commands to be entered appear in a different font, for example, **`copy a:*.* b:`**.

Button and key names appear in bold font, for example, click **Save** or press **Enter**.

Shading indicates content that has been added or changed in this revision of the document.

1.4 Technical assistance

For assistance or clarification on information in this document, submit a case to Qualcomm Technologies, Inc. (QTI) at https://createpoint.qti.qualcomm.com/. For QSPR-specific questions, file the case with the following problem areas:

- Initial Problem Type – Software
- Problem Area 1 – RF
- Problem Area 2 – RF Cal/Factory RF SW
- Problem Area 3 – RF Tool – QSPR

If you do not have access to the CDMATech Support Service website, register for access or send email to support.cdmatech@qti.qualcomm.com.

2 QSPR features

Key features of QSPR include:

- A highly customizable UI:
  - Windows can be floated, tabbed, docked, or put in Auto Hide mode to maximize available workspace
  - Minimal UI option – Displays only Run, Pause, and Stop, allowing users to run multiple QSPR instances simultaneously on a single station
  - Plug-in support – Users can create their own plug-ins to customize the way QSPR behaves or to add functionality.
  - Search functionality allows users to search by test name, parameter name, or other properties.
- User created individual and customized test trees composed of .Net functions, common test software (CTS) tests, SUITE tests, and scripts (Perl, Python, etc.):
  - All tests added to a test tree can be placed within folders, allowing the test organization to look like a tree via simple drag-and-drop techniques.
  - Properties that define the way the test runs can be set for each test or folders.
  - Input and output parameters of each test can be edited and saved.
  - Similar to programming, global variables can be used to set and get values for parameters. Values can also be shared among parameters in tests.
  - Pre- and postconditions can be placed upon each test and folder.
  - Test execution can be paused, resumed, or aborted. Breakpoints can be set to pause execution. Test execution can also be simulated.
  - Test trees created in applications such as CTS, older versions of QSPR (QSPR 2.0, formerly called HIVE), or QTM can be imported into a new test tree.
- Loop creation functionality:
  - Loops can be set by time, number of repetitions, or specific values. These specific values can be placed within the loop setup by a start, stop, step, or list format, or even taken from an external .csv or .txt file.
  - Loops can also have break conditions to specify when to break from a running loop.

- Detailed data analysis and logging functions:
  - Data viewing follows a conventional spreadsheet appearance, and data collected can be exported into .xml and .xls formats or be stored in a database. Charts can also be created to show the same data.
  - Trace and debug messages from tests can be easily recorded and saved within the Debug Window.
  - Support for conditional logging using the Logging Enabled property, where selected tests can be enabled to log debug messages.

3 Running a test tree

To run a test tree:

1. Navigate to the location of the .xtt test tree file (provided by QTI customer engineering or created in-house).
2. Select the file and click **Open**.
3. Run the test tree in one of the following ways:
   - Individual test – Right-click a test and select **Run Test**.
   - Folder – Right-click a folder and select **Run Test**. All selected tests within the folder will run (unselected tests are skipped).
   - Full tree – Click ▶ on the toolbar.

After a run is completed, test status is indicated as passed or failed with check marks or red Xs, respectively. Output data is placed in the Test results window (if the test results plug-in has been installed. See Appendix A.2).

4 Creating a test tree

4.1 Add .dll files to the tree registry

4.1.1 Load a .dll

To add .dll files to the tree registry, right-click the **This PC** folder in the Tree Registry window and select one the following options:

- Add SUITE SubSystem DLL – Load DLLs created using SUITE architecture for testing various subsystems, such as RFCal
- Add .Net DLL – Load any .Net assembly
- Add Script – Load script files such as Perl, Python, vb or .exes, and batch files

When a valid DLL is added, it will appear with all the tests it contains beneath it. If a DLL has the same name as another DLL that is already loaded, you will be prompted to either replace the currently loaded version or cancel.

4.1.2 Unload a .dll

To unload a .dll file:

1. Right-click the DLL.
2. Select **Unload DLL**. The path to the DLL should turn red to signify it is unloaded.

4.1.3 Rename a .dll

To rename a .dll file:

1. Right-click the DLL.
2. If the DLL is currently loaded, click **Unload DLL** (a DLL cannot be renamed while it is still loaded).
3. Right-click the DLL again and click **Rename DLL**.
4. Type the DLL's new name.

4.1.4 Remove a .dll

To remove a .dll file:

1. Right-click the DLL.
2. Click **Unload DLL** if it has not already been unloaded.
3. Right-click the DLL again and click **Delete DLL**.

4.1.5 Open a .dll folder path

To open the folder containing the .dll file:

1. Right-click the DLL.
2. Click **Open File Path**.

4.2 Add a .Net assembly

QSPR can load any .Net assembly. To list the public functions in the .Net assembly and use them as a test in the tree:

1. In the QSPR registry window, right-click **This PC** and select **Add Dot Net DLL**.
2. Select a DLL and click **Open**. When the DLL is opened, it shows in the registry.
3. To list the public functions that can be used as a test, right-click the DLL and select **Options**.
4. A window appears that lists the assembly name and the classes in that assembly in a tree format. Expand the tree until all the public functions are displayed.
5. Double-click a function name to make that function a test that can be dragged and dropped in the tree. The selected functions appear in the Tests column of the window, as shown below.

4.3 Add scripts to the tree registry

QSPR can use any script that can run under a shell (.exe, .bat, Perl, etc.). To add scripts to the tree registry:

1. In the Tree Registry window, right-click **This PC** and click **Add Script**.
2. Select a DLL and click **Open**.
3. For Perl, Python, or other scripts that need to look up the default handler for the script type, click the Properties tab and specify a program to run your script using the Verb argument in the Script Settings grouping.
4. To open a script, right-click the script file in the registry and click **Open**. This will either run the script or open it in an editor depending on whether a default program was chosen to open the file.

4.3.1 Modify or add script parameters

The arguments for running a script can be input or output parameters. To modify any of these parameters:

1. Locate the script that was previously added and right-click the file path (in this example, C:\QSPR\Samples\Tests\ScriptTest\bin\Debug\ ScriptTest.exe).
2. Click **Options**.
3. Within the window that appears, change values as necessary.
4. To add a new parameter to the script, click anywhere within the last grayed-out row and type the new parameter's name.
5. Fill in the other fields, including selecting Input or Output in the IO Type field to determine the parameter type.
6. Click **OK** to save script parameter changes.

Copy Options and **Paste Options** can be used to copy and paste script options between script tests. Click **Delete Row** to remove the selected parameter row.

4.3.2 Populate output parameters from a script

After a script executes, QSPR can populate output values using the values from the standard output of the script. It parses the contents of the standard output for a string that matches in the format `.<name of the output parameter>.<value>.` For example, to populate values for a script output parameter "Good Power Level", the script should output the following text in standard output:

This is a test. **Good Power Level. 10** is the value.

In this example, 10 is taken as the value for the parameter Good Power Level when the script is executed.

4.4 Adding SUITE subsystem DLL

To add a SUITE subsystem DLL to the tree registry:

1. In the Tree Registry window, right-click **This PC** and select **Add SUITE Subsystem DLL**.
2. Select a Suite Subsystem DLL and click **Open**. The DLL is loaded in the registry and the test that is exposed by the DLL displays below the test name.

4.5 DLL search path(s)

Users can define a set of directories from where the test DLL can be loaded, if the DLL is not available in the directory where the Tree Registry is pointing. This allows users to run the same test tree from two different computers, even if the folder where the test DLL is present is different in these two computers.

To set DLL paths:

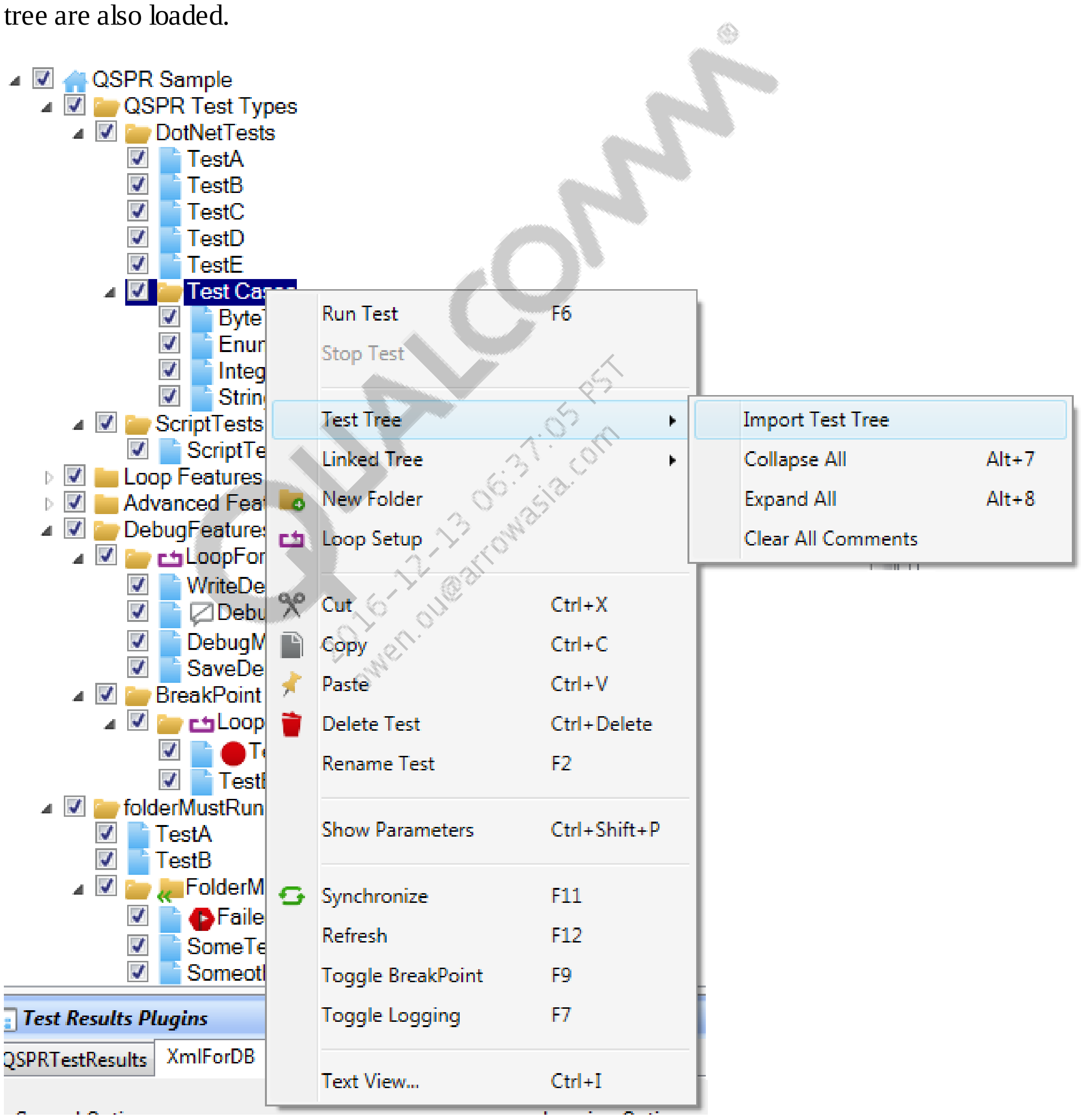
1. In the Tree Registry window, right-click **This PC** and select **DLL Path(s)**.
2. In the DLL Search Path(s) window, edit the list of directories for which you want QSPR to search.
3. Select **Use DLL Search Path(s)**. QSPR will now search for the paths provided for all test trees. DLL Path is a machine-level setup, not a test tree-level setup. The search paths must be defined for each computer. QSPR looks for the path of the test DLL to load in the following order:
   a. Paths specified in the DLL Search Path(s), if enabled
   b. Path specified in the test tree
   c. Path from which the application QSPR.EXE is installed

The Test Tree Registry displays the name of the DLL in italics if the path is different from the path mentioned in the tree. When the test tree is saved, the original path specified in the test tree is saved.

4.6 Import a test tree/linked tree

4.6.1 Import test tree

To import a test tree, locate the Test Tree option and click **Import Test Tree**, as shown below. This option allows you to browse for another saved test tree file and import it into the current test tree. The tree is imported to wherever the right-click is made. If you right-click in DotNetTests, for example, the entire imported tree is added under that folder. All settings and loops from that tree are also loaded.



4.6.2 Linked tree

Linked tree brings in a selected tree as a read only reference. Tree nodes seen are the same but cannot be edited – They must be edited in the source tree. When the source tree is changed, the linked tree reflects the updates. Once a reference tree has been linked, tests within that tree can be implemented in the newly created tree. This saves the user time by removing the need to recreate tests from an existing tree.

4.7 Organize the test tree

All items in the tree can use the drag-and-drop functionality, except for the root. Dragging a folder takes all items within the folder, wherever it is dropped. The dropped items are placed below the node on which they were dropped. Multiple items may be selected by holding the **Ctrl** key.

4.7.1 Using folders

To create a folder, right-click a node in the test tree and select **New Folder**. Tests and folders can be moved by dragging and dropping them to another location in the tree. Folders adhere to the following rules:

- When a test/folder is dropped in another folder, if the folder is in the Closed state, the dropped test is placed outside the selected folder. If a test in a looped folder is moved outside of that folder, you are prompted to confirm the move. If you confirm, that test is removed from the loop conditions of the folder it was in.
- A test cannot be placed directly under the root node. The first-level child must be a folder.
- If a test is dragged into a folder, it automatically inherits properties from that folder. For example, if the folder is marked as a Tool Test and a test is moved into that folder, it is automatically marked as a Tool Test as well.

4.7.2 Cut/Copy/Paste/Delete test

The Copy command copies the currently selected test or folder to the clipboard. It can then be pasted elsewhere on the tree by using Paste. The Cut and Delete options take the selected folder or test out of the tree.

4.7.3 Search

QSPR has a search feature that enables queries for specific test names, parameter names, etc. within a test tree. To find specific information in the test tree:

4. Select a test in the Tree View window, so that the Edit menu appears.
5. On the Edit menu, click Search Settings (**Ctrl+F**).

In Look in, you can choose what to look for. QSPR can search for these items:

- Test Name
- Real Name
- Branch On Error
- Breakpoint
- BookKeeping Test
- Parameter Name
- Retry Count
- Loop
- Stop On Error

- Pre Condition
- Post Condition
- Test Error Code
- Comments
- Registry

6. In the Find what box, type the text to search for. This box is enabled only for Test Name, Real Name, Parameter Name, Test Error Code, and Comments.
7. Click **Find Next**.
8. To go to the next occurrence, click **Edit→Find Next**, or **F3**.

If the Match case checkbox is selected, QSPR searches only for text that matches the case of the text that you typed in the Find what box.

Text typed in the Find what box also displays in the QSPR_SEARCH global variable. In this way, you can easily change the text that you want to search for by modifying the text in the QSPR_SEARCH global variable.

4.7.4 Synchronize

When working with the test tree, adding scripts and DLLs, and updating parameters, occasionally parameters are changed in the test DLL and not in the test tree. Select Synchronize to automatically synchronize changes across the test tree and your tree registry.

If new parameters are added or deleted in the test DLL, Synchronize updates the list of parameters in the test in the tree. For all the parameters that were present when the tree was first created, the edited values and description are not overwritten when the test is synchronized.

5 Setting properties, parameters, and global variables

5.1 Set test properties

The Properties tab displays information about the currently selected node. If a test is selected, information about that test is displayed. This information can be sorted by category, alphabetical order, or property pages in order from left to right.

All types of tests display the Attributes, Settings, and Target properties. However, the root node, .Net tests, and scripts have additional custom settings. Edit properties by typing a value, selecting the drop-down arrow, or clicking **…** to the right of the respective field.

5.1.1 Common properties

The attributes, debug properties, and settings are common to every type of test in QSPR.

5.1.1.1 Attributes

Attributes are:

- BranchOnError – Specifies the test to run when an error occurs
- DataLoggingOnlyOnFailure – If true, data is only collected when a test fails
- FolderMustRun – Test runs if any test within its parent folder runs
- IgnoreEmptyStringParameters – If true, empty string parameters are not logged in data collection
- StopOnError – Tells QSPR what to do if a test fails
- TestResultBasedOnLimits – If true, the pass result is verified with the upper and lower limits
- ToolTest – If true, the test does not run unless it is specifically branched to
- TreeMustRun – Test runs if any test within the tree runs

5.1.1.2 Debug

Debug options are:

- Breakpoint – Enables/disables a breakpoint that causes QSPR to temporarily pause at the start of a test tree. To resume testing, click **Resume Tree**. In Operator mode, even if a tree has breakpoints, it does not pause. For registered trees, enabling and disabling breakpoints does not change the registration.
- LoggingEnabled – Enables/disables the logging of debug messages for the selected test. Logging is enabled by default. The LOGGING_ENABLED global variable contains test names that have logging enabled. For registered trees, enable and disable logging does not change the registration, i.e., if users want to log debug messages for Test A and Test B only, uncheck Show Debug Messages to disable logging for all tests and then select Test A and Test B manually to change the LoggingEnabled property for these two tests to True. When running the test tree, only debug messages from these two tests are logged in the debug window. After QSPR finishes running the test tree, the value of the global variable LOGGING_ENABLED is updated with Test A, Test B.

5.1.1.3 Settings

- BookKeeping – If the property of a test is set to BookKeeping, the result of that test is always true, even if the test fails. The test result of a bookkeeping test does not affect the overall test result of the tree. When a booking test fails, the test result icon is shown with a checkmark and a cross. If it passes, a regular checkmark is shown.
- Description – Additional description about the test
- LoopSetup – Specifies looping setup
- PostConditionAction – Specifies postconditions, see Section 7.2Set a postcondition
- PreConditionEnabled – Specifies preconditions, see Section 7.1
- RealName – The original name of the test when it was created
- RetryCount – QSPR runs the test until either the test passes or the number of iterations reaches the retry count property. The results from retrying the test are not stored. Only the result from the last try of the test is stored. The number of tries before the test passes or a retry count is reached is stored as a loop condition RetryCount for the test.
- TestErrorCode – The error code string is set when the test fails. The GoNoGo result reflects the first error code set during GoNoGo.
- TestName – The test name used for data collection and display
- TestTimeOut – Specifies the length of time in seconds before the test times out and is stopped; a zero value indicates that the test will not time out
- TimeOutAction – Specifies the action to take when the test times out
- TargetName – Target name and location for the test

5.1.2 .Net test properties

The following settings are specific to .Net tests:

- CreateNewInstance – If true, creates a new instance of the class before the test is executed
- FailString – Specifies a return value for the test failure as a string

5.1.3 Script settings

The following settings are specific to scripts:

- CreateNoWindow – If false, the script creates a window to run the application
- DebugWindowName – Name of the script debug window
- RedirectStandardOutput – Set to true to get standard output from the script
- UseShellExecute – Set to true for running scripts; there will be no redirected output
- Verb – Action setup in Folder Options\File Types in Windows Explorer
- WorkingDirectory – Sets and gets the working directory of the script

5.2 Set folder properties

5.2.1 Attributes

Attributes are:

- DataLoggingOnlyOnFailure – If true, data will only be collected when the test fails
- FolderMustRun – Test runs if any test in its parent folder runs
- IgnoreEmptyStringParameters – If true, empty string parameters are not logged in the data collection
- RefreshLoopValuesAtLoopStart – Refreshes loop values from a file just before looping starts
- StopOnError – Tells QSPR what to do if a test fails
- TestResultBasedOnLimits – If true, the pass result is verified with the upper and lower limits for all tests within this folder
- ToolTest – If true, the folder does not run unless it is specifically branched to
- TreeMustRun – The folder runs if any test within the tree runs

5.2.2 Debug

Debug options for folders are the same as those for tests described in Section 5.1.1.2.

5.2.3 Settings

Debug options for folders are the same as those for tests described in Section 5.1.1.3.

5.2.4 Setup and cleanup

Setup and cleanup options are:

- IsCleanupFolder – If true, the tests within this folder will be run after GoNoGo finishes. The tests in this folder will not be logged. There can only be one Cleanup folder and it must be the last item under the Root folder.
- IsSetupFolder – If true, the tests within this folder are set as Tool Tests and these tests run before GoNoGo starts.
- WaitForLogTimeout – This sets the maximum amount of time in seconds the cleanup folder will wait for the XmlForDB plug-in logs to be written. This requires the IsCleanupFolder to be set to true.

5.3 Set root properties

5.3.1 Attributes

Attributes are:

- DataLoggingOnlyOnFailure – If true, data is only collected when the test fails
- IgnoreEmptyStringParameters – If true, empty string parameters are not logged in data collection
- RefreshLoopValuesAtLoopStart – Refreshes the loop values from a file just before starting the looping
- TestResultsBasedOnLimits – If true, the pass result is verified with the upper and lower limits

5.3.2 Debug

Debug options for the root are the same as those for tests described in Section 5.1.1.2.

5.3.3 Miscellaneous

TreeStringID – Shows the TreeStringID for the registered tree; shows -1 if tree is not registered

5.3.4 Registration

Registration options are:

- Database – Specifies the database where the tree is registered
- Enable Database – Warns the user of a nonregistered tree
- ForceRegistrationOnPlugInChange – If true, in GoNoGo tree type, registration is required if there are changes in the plug-in configurations
- RegisteredMachine – Machine name where the test tree was registered
- Server – Specifies the location of the tree registration server
- TreeID – Specifies the TreeID for the registered tree

5.3.5 Security

Security options are:

- EditQGroups – Specifies which QGroups have privilege to edit the test tree; anyone outside the listed Qgroups will only be able to uncheck tests and run the test tree. They will not be able to rename/move/add/delete tests.

5.3.6 Settings

Settings are:

- AlwaysPerformMustRunTests – If true, always execute tests that are marked as TreeMustRun, even if a tree is aborted or stopped
- BookKeeping – See Section 5.1.1.3
- Description – Additional description about the tree
- PauseOnTestFailure – In GoNoGo mode, if any test fails, testing is paused.
- LoopSetup – See Chapter 6
- RetryCount – See Section 5.1.1.3
- StopOnFirstTestFailure – In GoNoGo mode, if any test fails, testing is stopped.
- TestErrorCode –The error code string is set when the test fails. If the failed test does not have an error code, then the error code in the Root is set. The GoNoGo result reflects the first error code set during GoNoGo.
- TestErrorCodeOnAbort – When a test throws an exception, the error code string for that test is set. If that test does not have an error code, the TestErrorCodeOnAbort string is set. Similarly, the error code string TestErrorCodeOnAbort is also set if the user aborted a GoNoGo tree. The GoNoGo result reflects the first error code set during GoNoGo.
- TestName – Specifies the name of the tree for data collection and display
- TimeOutAction – See Section 5.1.1.3
- TreeTimeOut – See TestTimeOut in Section 5.1.1.3
- TreeType – Specifies the type of testing to be done; the TreeType can be GoNoGo or DVT.
    - GoNoGo – All database-related restrictions are applied. Editing on the tree (such as checking/unchecking tests, rename of test, change parameters) results in the registration being reset to 0. When data is stored in the database, the tree type information is also stored. Qualcomm recommends that you set the tree to this type when the tree is ready for production.
    - DVT – This tree type is used when the user is still modifying the tree. When a registered tree is edited, deleting or unselecting tests do not change the registration. Optional to have configuration file set up.
    - TestPlan – Reserved

5.3.7 Tester

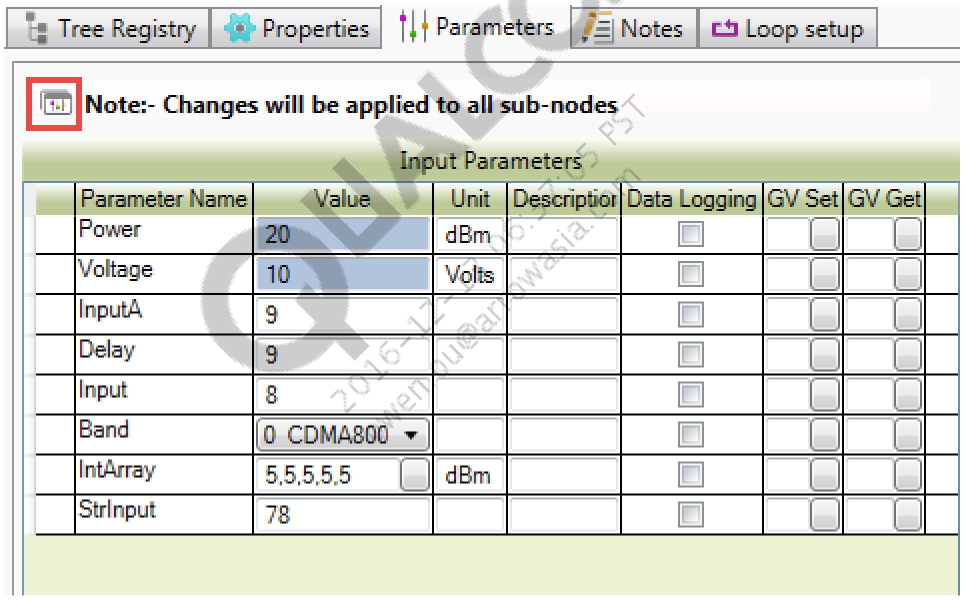
Tester options are:

- ConfigFileName – Specifies the location of the Tester Configuration file
- RefreshConfig – Specifies whether to reload the Tester Configuration file each time GoNoGo runs

5.4 Configure parameters for a test or folder

The Parameters tab is used to edit parameters related to the currently selected test/folder in the tree view.

1. Select the test/folder for which to configure parameters.
2. Select the **Parameters** tab. If a folder was selected in step 1, the tab will appear empty. Click **Show unique parameters for all tests** (highlighted below) to display parameter details.



Note:- Changes will be applied to all sub-nodes

Input Parameters

Parameter Name	Value	Unit	Description	Data Logging	GV Set	GV Get
Power	20	dBm		☐		
Voltage	10	Volts		☐		
InputA	9			☐		
Delay	9			☐		
Input	8			☐		
Band	0 CDMA800			☐		
IntArray	5,5,5,5,5	dBm		☐		
StrInput	78			☐		

3. Edit the settings of each parameter as necessary (see subsequent sections for details). The following rules apply:
   - When a folder is selected, some parameter fields may be highlighted blue. This indicates that the parameter is shared between multiple tests in the folder and that the parameter's value differs across tests. Editing highlighted fields will apply changes to all tests within the folder.
   - To edit an array value, click the button that appears to the right of the text in the value cell. This opens a window that enables editing for individual values and data logging options for each array value. For .Net tests, it is possible to insert and delete array items. This overrides the default array size specified in the .Net test code.

5.4.1 Input parameters

The upper portion of the Parameters view shows input parameters:

- Parameter Name – Name of the parameter; read-only
- Value – Value of the parameter; when entering the new value, the application validates the value based on the specified data type before the new value is displayed
- Data Logging – If checked, data is logged when the tree is running
- Unit – Unit of measurement
- Description – Description of the parameter
- GV Set – QSPR takes the value of the parameter and sets the global variable to the parameter's value (see Section 5.4.4)
- GV Get – QSPR takes the value from an existing global variable and assigns that value to the parameter (see Section 5.4.4)

5.4.2 Output parameters

The upper portion of the Parameters view shows input parameters:

- Parameter Name – Name of the parameter; read-only
- LL – Lower Limit; when entering a new value, QSPR validates the value based on the specified data type before the new value is displayed
- Value – Value of the parameter; read-only
- UL – Upper Limit; when entering a new value, QSPR validates the value based on the specified data type before the new value is displayed
- Data Logging – If checked, data is logged when the tree is running
- Logging Condition – When a condition is set, data is logged only if the condition is satisfied; a logging condition can only be set if Data Logging is checked
- Charting Enabled – If checked, data is used to plot charts in the Chart Mantra plug-in
- Unit – Unit of measurement
- Description – Description of the parameter
- GV Get – QSPR takes the value from an existing global variable and assigns the value to the parameter (see Section 5.4.4)
- GV Set – QSPR takes the value of the parameter and sets the global variable to the parameter's value (see Section 5.4.4)
- GV LL Get – QSPR takes the value from an existing global variable and assigns the value to the parameter's lower limit
- GV LL Set – QSPR takes the lower limit value of the parameter and sets the global variable to the parameter's lower limit
- GV UL Get – QSPR takes the value from an existing global variable and assigns the value to the parameter's upper limit

- GV UL Set – QSPR takes the upper limit value of the parameter and sets the global variable to the parameter's upper limit

5.4.3 Parameter logging

To collect data logs for a parameter, select the checkbox for the parameter in the Data Logging column.

To exclude a parameter from the logs, right-click the name of a parameter and click **Exclude Parameter Logging (Global)**. To include a parameter that has been previously excluded, right-click the name of a parameter and click **Include Parameter Logging (Global)**. Exclude/include settings apply to the entire test tree.

To set a condition for data logging for an output parameter, select the **Logging Condition** checkbox. Setting the data logging condition is similar to setting a precondition (see Section 7.1).

5.4.4 Global variables

Global variables are a way to share data between tests. They can be used to:

- Save the output of a test for use in another test
- Provide input data to another test or loop
- Make decisions based on the results of a previous test using pre and postconditions
- Save an error code from a log

In the context of this section, global variables can be used to set an input parameter and save the data from an output parameter.

5.4.4.1 Global variables – Get and set

In the parameter window, global variables can be created or modified based on test results. Two columns are available, GV Set and GV Get. To set values for either of these, click the empty box that appears in the cell to open the Global Variable Manager.

When setting a global variable as Set, QSPR takes the value of the linked parameter and sets the global variable to that value. When setting it as Get, QSPR gets a value from an existing global variable and uses it as the value of the linked parameter.

All global variables from the selected tree appear within this window. Also, any data cell can be modified by clicking it and retyping the value. If you would like to add a new variable completely, click **Add** to create a new row where new values can be entered. To remove global variables, select the row and click **Remove**.

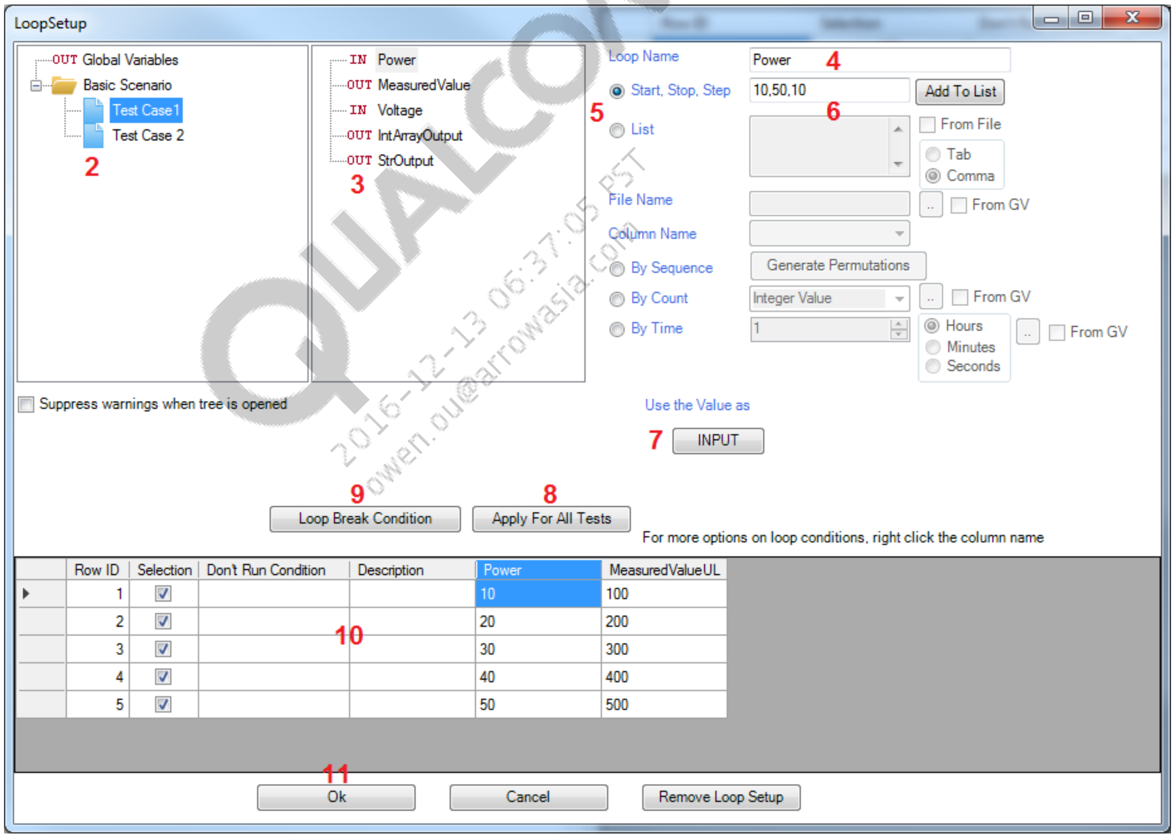
5.4.4.2 Embedding global variables in an input parameter string value

In addition to the GV Get and Set columns, global variables can also be embedded inside input string parameters using the convention "{GV_NAME}". For example, to concatenate two or more global values in a single string parameter, set the input parameter value as "{MONTH}/{DAY}/{YEAR}". Assuming the global variables have the values MONTH=10, DAY=31, and YEAR=2013, this input parameter value will be replaced at runtime with the string 10/31/2013. Global variables can also be dragged into a parameter value from the Global Variable view in QSPR. If a value that already contains a string is dragged, it will concatenate the global name and enclose it in curly brackets to signify it is an embedded global that should be replaced at runtime.

6 Creating loops

QSPR’s loop functionality allows users to repeat tests with varied conditions and gather data for future review. The user can customize testing by setting which values to hold constant and which values to change and by how much. To create a loop:

1. Right-click a folder or node and select **Loop Setup**. Any folder with tests may be looped except folders that are directly under the root.



Loop Name: Power

[✓ Start, Stop, Step]: 10,50,10

By Time: 1

Use the Value as: INPUT

	Row ID	Selection	Don't Run Condition	Description	Power	MeasuredValueUL
▶	1	✓			10	100
	2	✓			20	200
	3	✓			30	300
	4	✓			40	400
	5	✓			50	500

2. Select a test from the list or select Global Variables.

3. Select an input or output parameter from the list. If Global Variables was selected in step 2, select a global variable from the list. If creating a loop for a folder, no selection in this area is necessary.

4. Enter a unique name for the loop. Loop names can be changed later by right-clicking the loop name when it is added to the lower grid and selecting **Change Loop Name**.

5. Select the type of loop:
   - Start, stop, step – See section 6.1.1
   - List – See Section 6.1.2
   - Count – See Section 6.2.1
   - Time – See Section 6.2.2
   - Sequence – See Section 6.2.3
6. Fill in the values for the new loop.
7. Select how to use the value:
   - For input parameters, there is only one choice – Click **Input**.
   - For output parameters, click either **UL** or **LL** to set the entered values as the upper or lower limits for the parameter, respectively.
8. To apply parameter changes to all tests in the loop folder that share the selected parameter, click **Apply For All Tests**.
9. If necessary, click **Loop Break Condition**. A loop break condition is a way to terminate a loop if specified conditions are met. Loop break conditions are evaluated after each iteration of the loop. To set a loop break condition, use the process described in Section 7.1.
10. For each value that is looped, a row is created in the grid. To edit the loop information, select a cell. See Section 6.3 for more information on editing the grid.
11. Click **OK** to save the loop. The icon will appear next to the test the loop applies to in the test tree.

To edit or remove a loop from a folder, select the folder and click **Remove Loop Setup**.

6.1 Configure parameter loops

6.1.1 Start, Stop, Step

The Start, Stop, Step loop type takes a value in the format X,Y,Z, where X is the number the loop starts on, Y is the value the loop stops at, and Z is the increment between each loop. For example, an entry of 3,10,1 starts the loop on 3 and increments by 1 until it gets to 10.

Optionally, click **Add to List** to expand the Start, Stop, Step into the List loop type. The example Stop, Start, Step loop of 3,10,1 is equivalent to a List of 3,4,5,6,7,8,9,10.

6.1.2 List

The List loop type takes a comma-delimited list of values. For example, a List loop with the entry 3,4,5,6,7,8,9,10 will loop sequentially on those values. This is equivalent to a Start, Stop, Step of 3,10,1; The List loop type exists to add more flexibility to the values the user can loop.

As an alternative to manually typing the values for your loop in the List field, an external .csv or .txt file can be used to load values. To load a file:

1. Click **List**
2. Click **From File**.

3. Click **…** and select the file.
4. Select **Tab** or **Comma** to indicate if the values in the file are comma- or tab-delimited.
5. In the Column Name field, select either **X Values** or **Y Values** to indicate which column of values to use in the file.

The filename can also be retrieved from a global variable:

1. Click **List**.
2. Click **From File**.
3. Click **From GV**.
4. Click **…** to open the Global Variable Manager.
5. Click the checkbox associated with the global variable. Only variables that contain information in the Value column are valid selections.
6. Click **OK**.
7. Select **Tab** or **Comma** to indicate if the values in the file are comma- or tab-delimited.
8. In the Column Name field, select either **X Values** or **Y Values** to indicate which column of values to use in the file.

The following is an example that could be used for a List loop:

```
X axis, Y axis
1, 3
2, 4
3, 7
4, 12
```

6.2 Configure folder loops

6.2.1 By Count

The By Count loop type takes a numerical value and loops on a folder that many times. Loop Forever can be also selected to run a loop until a specified condition is met.

The By Count value can also be retrieved from a Global Variable:

1. Click **By Count**.
2. Click **From GV.**
3. Click **…** to open the Global Variable Manager.
4. Select the checkbox associated with the global variable. Only variables that contain information in the Value column are valid selections.
5. Click **OK**.

The By Count loop type cannot be used in combination with List, By Time, or Start, Stop, Step loops.

6.2.2 By Time

The By Time loop type loops on a folder for a specified length of time. To configure a By Time loop:

1. Click **By Time**.
2. Enter a value in the By Time field.
3. Select **Hours**, **Minutes**, or **Seconds** to indicate the unit of measurement.

The By Time value can also be retrieved from a Global Variable:

1. Click **By Time**.
2. Click **From GV**
3. Click **…** to open the Global Variable Manager.
4. Select the checkbox associated with the global variable. Only variables that contain information in the Value column are valid selections.
5. Click **OK**.
6. Select **Hours**, **Minutes**, or **Seconds** to indicate the unit of measurement.

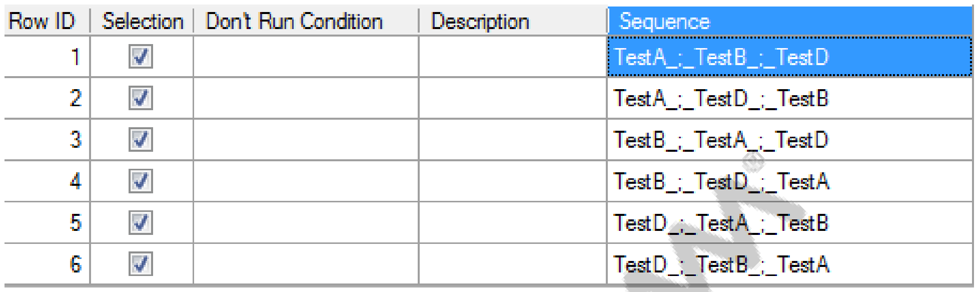
The By Time loop type cannot be used in combination with List, By Count, or Start, Stop, Step loops.

6.2.3 By Sequence

The By Sequence loop type runs a set of tests under a folder in all possible permutations. For example, if a folder contains two tests, TestA and TestB, and those tests are looped by sequence, the folder runs TestA followed by TestB the first time through the loop. The second time through the loop, TestB runs followed by TestA. To configure a By Sequence loop:

1. Click **By Sequence**. Checkboxes will appear next to all tests and folders in the test tree view.
2. Select the checkboxes corresponding to the tests to permute. At least two tests must be selected. Selected tests must be consecutive within the same parent folder.
3. Click **Generate Permutations**. Selected tests will be highlighted in red and are no longer selectable since they are already part of a sequence loop.

The loop grid will now contain six loop rows representing each of the six sequences of the selected tests. The loop value cell in each row contains the name of each test separated by the characters “_:_” in the order they run for that particular loop iteration. The following is an example By Sequence loop.



Row ID	Selection	Don't Run Condition	Description	Sequence
1	✓			TestA_:_TestB_:_TestD
2	✓			TestA_:_TestD_:_TestB
3	✓			TestB_:_TestA_:_TestD
4	✓			TestB_:_TestD_:_TestA
5	✓			TestD_:_TestA_:_TestB
6	✓			TestD_:_TestB_:_TestA

When a sequence loop is run, the tests selected as part of the loop creation run in an order based on the loop settings. In the example loop shown above, the folder runs six times. The tests selected run in the order specified in the loop row. Any unselected tests (in this case TestD is unselected) run in the same order as they are shown in the tree. This example loop run results in the following sequences:

- Loop iteration 1 – TestA, TestB, TestC, TestD
- Loop iteration 2 – TestA, TestC, TestB, TestD
- Loop iteration 3 – Test B, TestA, TestC, TestD
- Loop iteration 4 – Test B, TestC, TestA, TestD
- Loop iteration 5 – Test C, TestA, TestB, TestD
- Loop iteration 6 – Test C, TestB, TestA, TestD

6.3 Edit the grid

Using the right-click context menu, rows and columns in the grid can be easily edited. The same menu appears upon right-clicking either a row or column, but not all options can be selected for each. The following options are available:

- Move Right – This moves the currently selected column to the right.
- Move Left – This moves the currently selected column to the left.
- Add Row – This appends a new blank row to the end of the grid.
- Delete Column – After selecting a column, you can right-click it and use Delete Column to remove it.
- Add Row – This adds a row to the currently selected row.
- Delete Row – This deletes the currently selected row or rows. To select more than one row at a time, click the row highlight area and drag the mouse up or down. This selects multiple rows.
- Change Loop Name – After selecting a column and choosing Change Loop Name, a text box appears above the grid where you can edit the loop name. When renaming is complete, left-click the column again to apply the change.

- Editing rows – Copy, cut, and paste allows you to copy a selected row and move it to another selected area in the grid.
- Paste Last – The Paste Last option allows you to copy a selected row and automatically paste to the bottom of the grid.
- Remove Loop Setup – This erases all loops currently in the grid.

Don't run condition

Adding a Don't Run Condition to a selected row or loop will prevent that loop from running f certain conditions are met. Multiple types of conditions can be placed here, including the <=, =<, >=, =<, !=, =, <, >, &, | commands.

For example, if you do not want loop 1 to run when the variable ASPower is greater than 30, you can type ASPower > 30 in the Don't Run Condition box for loop 1. The variable used has to be at least at the parent level to the loop, or above, for this to work. The loop then moves on to the next loop within the loop each successive time this condition occurs.

7 Setting test conditions

All tests in the QSPR can have pre- and/or postconditions applied. These conditions tell QSPR to run only specified folders or tests, depending on the results from other tests. Two types of conditions can be set:

- Precondition – Applied before a test to determine if the test should run. If the precondition indicates that its test should not run, the test is skipped and execution moves to the next test in the Test Tree.
- Postcondition – Evaluated after a test run to determine if a specified post condition action should be taken.

7.1 Set a precondition

To set a precondition:

1. Select a test from the test tree.
2. Click the **Properties** tab.
3. Click the **PreConditionEnabled** line.
4. Click **…** to open the condition editor.
5. Click **New Condition**. The condition is assigned a default name (for example, C1) and appears within the Formula Editor. To change the name from the default, edit the text in the Name field.
6. Select the **Type**:
   - Test Parameters – See Section 7.1.1
   - Test Result – See Section 7.1.2
   - Global Variable – See Section 7.1.3
7. If multiple conditions are created, they can be combined into a compound condition using Boolean logic in the Formula Editor. The following actions can be performed in any order to create a formula:
   - Double-click conditions to add the condition at the cursor's location within the formula field. If text is highlighted in the formula field, double-clicking a condition will replace the highlighted text.
   - Click the operator buttons (AND, OR, and NOT) to add a Boolean operator to the formula at the cursor's location within the formula field. If text is highlighted in the formula field, clicking an operator button will replace the highlighted text.
   - Click **( )** to add parentheses at the cursor's location within the formula field. If text is highlighted in the formula field, parentheses are inserted around the highlighted text. Formulas are evaluated using the standard precedence rules indicated by the parentheses.

When the formula is complete, click **Check Syntax** to validate it. If the returned message indicates that the formula is invalid, continue editing until validation is successful.

8. Select one of the following:
   - Met – The test runs when the condition (or compound condition) is met. A condition is considered to be met if its formula evaluates to true.
   - Not Met – The test when the condition (or compound condition) is not met. A condition is considered to not be met if its formula evaluates to false.
9. By default, new loop break conditions are active upon creation. To make a loop break condition inactive, deselect **Condition Active**.
10. Click **OK** to save the condition. The icon will appear next to the test the condition applies to in the test tree.

7.1.1 Configure a Test Parameters condition

A Test Parameters condition creates an equation that compares a test parameter to a constant value, the value of a global variable, or one of the parameter's limits. When a Test Parameters condition is applied, a test will only execute if the created equation returns true. To configure a Test Parameters condition:

1. Select **Test Parameters** in the type field.
2. Select a test from the tree at the top of the Loop Condition window. The parameters of the selected test will populate.
3. Select a parameter in the Condition Editor.
4. If the selected parameter has more than one element, choose the index from the drop-down Index menu.
5. Select the mathematical comparison type:
   - Equal
   - Not equal
   - Less than
   - Less than equal
   - Greater than
   - Greater than equal

   The selected parameter is the left-side operand of the equation. For example, selecting less than results in the parameter being less than the right-hand operand:

6. Select the right-side operand of the equation:
   - Value – Compares the value of the selected parameter to a constant value. Enter a value.
   - Global – Compares the value of the selected parameter to the value of a global variable. To select a global variable:
     - i Click **…**
     - ii Select a global variable
     - iii Click **OK**.
   - Lower Limit – Compares the value of the selected parameter to its defined lower limit
   - Upper Limit – Compares the value of the selected parameter to its defined upper limit
7. Click **Save Condition**.

7.1.2 Configure a Test Results condition

A Test Results condition checks the result of a previously run test. When a Test Results condition is applied, a test will only execute if the selected previous test returns the specified value of pass or fail. To configure a Test Results condition:

1. Select **Test Results** in the type field.
2. Select a test from the tree at the top of the Loop Condition window.
3. Select Passed or Failed.
4. Click **Save Condition**.

7.1.3 Configure a Global Variables condition

A Global Variables condition compares a global variable either to a constant value or to another global variable. To configure a Global Variables condition:

1. Select **Global Variables** in the type field.
2. Select a global variable from the list on the left side of the Condition Editor.
3. Select the mathematical comparison type:
   - Equal
   - Not equal
   - Less than
   - Less than equal
   - Greater than
   - Greater than equal

   The selected global variable is the left-side operand of the equation. For example, selecting less than results in the global variable being less than the right-hand operand:
4. Select the right-side operand of the equation:
   - Value – Compares the value of the selected global variable to a constant value. Enter a value.

- Global – Compares the value of the selected global variable to the value of another global variable. To select a global variable:
  - i Click **…**
  - ii Select a global variable
  - iii Click **OK**.

5. Click **Save Condition**.
6. Click **OK**.

7.2 Set a postcondition

To set a precondition:

1. Select a test from the test tree.
2. Click the **Properties** tab.
3. Click the **PostConditionAction** line.
4. Click **…** to open the condition editor.
5. Click **New Condition**. The condition is assigned a default name (for example, C1) and appears within the Formula Editor. To change the name from the default, edit the text in the Name field.
6. Configure the postcondition as described in Section 7.1.
7. Click the **Selection Action** and select one of the following:
   - Skip the folder
   - Skip to next looped iteration
   - Skip to first test after the loop
   - Skip to next category
   - Complete stop
   - Complete stop no retry
   - Ignore session
   - Restart test tree
   - Restart test tree without must runs
   - Branch
   - Pause test tree
   - None
8. If branch was chosen as the selection action, select a test to branch using the tree window. Alternatively, select **Branch on TestName** and enter the name of the tree.
9. Click **Save Condition**.
10. Click **OK**. The icon will appear next to the test the condition applies to in the test tree.

A Configuring the QSPR environment

A.1 Security features

QSPR provides options for setting up security to restrict access depending on the user. There are two types of security in QSPR, Windows account-based and password-based. In both cases, there are at least two levels of security. The highest level of permission gives full access to open, create, edit, save, and run test trees. The other permission levels can only open and run test trees. The default security settings are based on Windows accounts. When password security is enabled, it overrides the windows account security.

A.1.1 Windows account-based security

In Windows account-based security, all administrator accounts on the machine have full access to QSPR. Nonadmin accounts can only open and run trees. An admin user can switch to Password mode by going to the Security configuration tab in the Configuration dialog.

A.1.2 Password-based security

In password-based security, all users are restricted from creating/editing trees until they log in with the password. Once the user has logged in, they are free to change or disable the password. Disabling the password returns QSPR to its default Windows account-based security settings.

A.2 Plug-ins

QSPR for external customers (installed with QDART), comes with several plug-ins pre-installed. To view/edit QSPR plug-ins:

1. Click **Options -> Edit Configurations**.
2. Select the **Workspace Configuration** tab.
3. Make any necessary changes in the plug-in Configuration section.
4. Click **OK**.

B Running QSPR from the command line

QSPR provides command line options for launching QSPR such as opening/running a tree, and closing when finished. To view the current options available, launch QSPR.exe from the command line with the /help option:

```
C:\Windows\system32>”C:\Program Files (x86)\QUALCOMM\QDART\bin\QSPR.exe” /help
```

C Icon glossary

C.1 Tree view icons

🏠	Root node
📁	Collapsed folder
📂	Expanded folder
🔁	Looped folder/test icon
⬇	Test/folder queued to run
📄	Test
🔄	Test/folder is running
✅	Test/folder passed
❌	Test/folder failed
✓✗	Test/folder passed at least once, failed at least once
📁🔒 📄🔒	Folder/Test is from a linked tree and is not editable
✖	Aborted
🛑	Stop on error
⑂	Branch on error
💬	Test has comments
«	Pre-condition
»	Post-condition
↻	Scenario break condition
▶	Folder must run
▶	Test must run
●	Breakpoint enabled
○	Breakpoint disabled
💬	Logging disabled
⏸	Root node's PauseOnTestFailure property is set to True
⏹	Root node's StopOnFirstTestFailure is set to True

C.2 Tree registry

💻	Registryroot folder
✓	Loaded file
⛔	Unloaded file
📄	Test name
{}	Workspace name
■●▲◆	Class name

D References

D.1 Acronyms and terms

Acronym or term	Definition
CTS	Common Test Software
QSPR	Qualcomm Sequence Profiling Resource