1

Working With Testplans

This chapter describes how to use testplans, which are named sequences of tests that are executed as a group to test a specific device or unit under test.

For an overview of testplans, see Chapter 3 in the *Getting Started* book.

A Suggested Process for Creating a Testplan

Although we have no way of knowing about your specific hardware, we recommend that you consider the following process when creating a testplan.

Preparing to Write the Testplan

1. Gather the testing specifications and requirements for the UUT (unit under test).

   You must thoroughly understand the UUT before you can test it effectively. This includes both the physical (such as pinouts) and electrical characteristics of the device.

2. Plan the tests and the sequence in which they will be executed.

   Determine which kinds of tests are needed in your testplan (including tests for failure and exception handling, if desired). Determine the order in which the tests should be executed. Given the above, determine where to use test groups.

   Tip: You may find it useful to draw a worksheet and make copies of it to write on when planning tests. For example, the worksheet might briefly describe the test, list the hardware resources needed, the test limits, any setup or cleanup requirements, timing constraints, a list of input and

output pins, and such. An example of a typical worksheet is shown below.

TEST NAME Volt2DMM

Measurement(s): (measure, limits)	10 volts Limits: 9.9 - 10.1
Preconditions:	
UUT Setup	None
Connections	V Src hi to DMM hi, V src lo to DMM lo
Power (volts, amps, pin)	N/A
Load (value, power, pin)	N/A
Constraints:	
Timing	N/A
Test Sequencing	N/A
Test Description:	Output 10 volts w/voltage source & measure w/DMM
Reuse:	
Test Templates	Volt2DMM
Actions	Switching, Configure V source, Measure DMM
Instruments: (name, settings)	V source 10 volts DMM volts

3. Plan the system resources for each pin on the UUT.

   Using the information from the previous step, be sure your test system has the hardware resources needed to do the tests. For example, do you have enough power supplies, signal sources, and signal detectors? If not, you must add hardware or find a way to simplify the tests.

4. Plan and build the fixture or other means of connecting the test system's hardware with the UUT.

   Pins on the UUT must be connected to the test system's power supplies, signal sources, and signal detectors. If you test various kinds of UUTs on a single test system, you may want to use an interchangeable fixture to make the connections. Or, you need some type of cabling to make the necessary connections.

If you are using programmable switches, such as switching cards, to make connections between resources and the UUT and you have hardware handler software for those switches, you probably will want to use the Switching Topology Editor to define your topology so you can use switching actions in your tests.

Writing the Testplan

1. Add tests and test groups to your testplan.

2. Copy and customize existing tests from libraries where possible. Where needed, add the tests to test groups. If there is no existing test to reuse, create new tests from existing actions in libraries where possible. If no suitable actions exist from which you can create a new test, create new actions, add them to an action library, and then create a new test from them.

3. Tune the tests for performance and reliability.

   This process can be as flexible as you like. For example, you might begin by creating actions, using them to create tests, and then using the tests to create a testplan. But if it is more convenient—for example, if different people are developing the actions and the testplan—you may want to begin with an empty testplan and then expand it by adding tests as the actions needed to create the tests become available.

   For more information about tuning tests, including how to use HP TestExec SL's built-in profiler, see "Optimizing the Throughput of Testplans."

To Create a Testplan

Use the Test Executive's graphical tools to create a testplan.

1. Click  in the toolbar or choose File | New in the menu bar.
2. Choose Testplan as the type of document.
3. Choose the OK button.
4. Add one or more tests or test groups to the list shown in the left pane of the Testplan Editor.

   For information about adding tests and test groups, see "Using Tests & Test Groups in Testplans."
5. Click  in the toolbar or choose File | Save in the menu bar.
6. Enter a name for the testplan.
7. Choose the Save button.

To Specify Switching Topology Layers for a Testplan

The switching topology information for a specific testplan resides in three files whose extensions are ".ust". These files contain information about the system, fixture, and UUT layers of switching topology. Given that one test system can use many testplans, you must specify which switching topology files to use for a given testplan.

Each test system has one system layer defined for it, and the name and location of the file containing the system layer resides in HP TestExec SL's initialization file. This is described under "System Setup" in Chapter 6.

Although you can locate the remaining two files, which contain the fixture and UUT layers, wherever you like, it usually makes sense to put them with other files used with the testplan. Then you must associate these two topology files with the testplan.

Do the following to associate the files for the fixture and UUT layers with the testplan:

1. Load the testplan.

2. Choose Options | Switching Topology Files in the menu bar.

3. Specify the locations of the files for the fixture and UUT layers.

For an overview of switching topology, see "About Switching Topology" in Chapter 3 of the *Getting Started* book. For detailed information, see Chapter 4 in this book.

Using Tests & Test Groups in Testplans

Note

The Testplan Editor window supports the various mechanisms that Microsoft Windows provides to select multiple items; i.e., holding the Ctrl key as you click multiple items; pressing and holding the mouse's left button and then dragging across multiple items; and clicking the first item in a desired list, simultaneously pressing and holding the Ctrl and Shift keys, and clicking the last item in the list. This means that many of the tasks described for individual tests or test groups also can apply to multiple tests or test groups. For example, if you select multiple tests or test groups, you can copy or delete them as you would a single test or test group.

To Add a New Test/Test Group

1. Click the desired insertion point in a testplan shown in the left pane of the Testplan Editor window.

   The test or test group will be inserted immediately before the line selected as the insertion point.

2. Do one of the following:

   - *To insert a test*, click  in the toolbar or choose Insert | Test in the menu bar.

     - or -

   - *To insert a test group*, click  in the toolbar or choose Insert | Test Group in the menu bar.

3. Do the following in the right pane of the Testplan Editor window:

   a. Specify a name for the test or test group.

If you are using datalogging, be aware of the following restrictions on the names of tests or test groups:

- If your log data is processed by HP Pushbutton Q-STATS, you must not use slashes (/ or \) in test names.
- If your log data is processed by Q-STATS II, only the first forty letters of the test name are significant.

b. Add any desired actions to the test or test group.

See “To Add an Action to a Test/Test Group” in Chapter 2 for more information.

c. If you wish to use variants to provide multiple versions of the parameters and limits, specify them.

See “To Add a Variant to a Testplan” for more information.

To Add an Existing Test

The easiest way to create a test is to reuse a similar test from a test library.

Note

Be sure the search paths for test libraries are set up correctly or you may not be able to find the test you want; see “Specifying the Search Path for Libraries” in Chapter 5.

1. With a testplan loaded, choose an insertion point in the left pane of the Testplan Editor window.

   The test will be inserted immediately before the line selected as the insertion point.

2. Click  in the toolbar or choose Insert | Saved Test in the menu bar.

3. When the Test Libraries box appears, use it to find an existing test similar to the one you need.

For more information about using the Test Libraries box's search features, see "Searching for Items in a Library" in Chapter 5.

4. Make a copy of the test under a new, unique name.

5. Modify the existing actions as needed.

   For more information, see "To Specify Parameters for Actions in a Test/Test Group" and "To Specify Limits for Actions in a Test/Test Group" in Chapter 2.

6. Modify the existing parameters as needed.

   For more information, see "Specifying Parameters for a Test/Test Group" in Chapter 2.

To Examine or Modify a Test/Test Group

1. Click a test or test group shown in the left pane of the Testplan Editor window.

2. Use the right pane of the Testplan Editor window to examine or modify the contents of the test or test group.

See Chapter 2 for information about specifying the contents of tests and test groups.

To Move a Test/Test Group

1. Click a test or test group shown in the left pane of the Testplan Editor window.

2. Choose  in the toolbar or Edit | Cut in the menu bar.

3. Click the desired new location for the test or test group.

   If you click an existing line, the test or test group will be inserted before that line.

4. Choose  in the toolbar or Edit | Paste in the menu bar.

Note

If desired, you can move a test or test group from one testplan to another. Follow the procedure described above, but run two instances of HP TestExec SL. Cut the test or test group from a testplan in one instance and paste it to a testplan in the other instance.

To Copy a Test/Test Group

1. Click a test or test group shown in the left pane of the Testplan Editor window.

2. Choose  in the toolbar or Edit | Copy in the menu bar.

3. Click the desired new location for the test or test group.

   If you click an existing line, the test or test group will be inserted before that line.

4. Choose  in the toolbar or Edit | Paste in the menu bar.

Note

If desired, you can copy a test or test group from one testplan to another. Follow the procedure described above, but run two instances of HP TestExec SL. Copy the test or test group from a testplan in one instance and paste it to a testplan in the other instance.

To Delete a Test/Test Group

1. Click a test or test group shown in the left pane of the Testplan Editor window.

2. Choose Edit | Delete in the menu bar.

Controlling the Flow of Testing

Using Flow Control Statements

Note

Because you specify flow control statements in predefined, “fill in the blanks” dialog boxes, you do not need a detailed understanding of their syntax. If you make an error in entering the syntax, you will be prompted to correct it.

Which Flow Control Statements are Available?

HP TestExec SL supports the following statements that let you control the flow of testing in a testplan.

if...then...else Conditionally executes one or more statements in the testplan, depending upon the value of an expression.

```
if Expression then
  [statements]
[else
  [statements]]
end if
```

Example:

```
if System.RunCount = 0 then
  test Test1
else
  test Test2
end if
```

for...next Repeats one or more statements in the testplan a specified number of times. A negative value for `Step` causes the counter to decrement.

```
for Variable = Start to End step Step
  [statements]
next
```

Example:

```
for Counter = 1 to 5 step 1
  test Test1
next
```

for...in

Repeats one or more statements in the testplan for each value in a list of arguments.

```
for Variable in Group
  [statements]
next
```

Example:

```
A = 4
B = 2
C = 9
for SequenceLocals.MyVariable in C,A,B
   ! Assume that SequenceLocals.MyVariable
   ! is passed as a parameter to Test1
   test Test1
next
```

loop

Repeats one or more statements in the testplan until a condition specified in an expression is satisfied.

```
loop
   [statements]
   exit if Expression
end loop
```

Example:

```
loop
   test Test1
   test Test2
   exit if SequenceLocals.MyVariable = 3
end loop
```

It also supports the miscellaneous syntax elements listed below, which you can use with the flow control statements.

= (assignment operator)	Sets a variable to a value. *Variable = Value* Example: X = 2 SequenceLocals.MyVariable = 7
comment	Non-executing line used to document a testplan. Example: ! This is a comment
else, **end if**, **next**, **end loop**, **exit if**	Syntax elements used with the flow control statements. Some of these are required and others optionally extend the functionality of the flow control statements.

What Are the Rules for Using Flow Control Statements?

Keep the following in mind when using flow control statements:

- Variable names can be either the name of the symbol by itself, such as “A” or “MySymbol”, or include the name of an internal or external symbol table, such as “SequenceLocals.MySymbol”.

  Note: In most cases, variables in flow control statements should use symbols in global symbol tables, such as SequenceLocals or System, instead of using a symbol table whose scope is more restricted, such as TestStepLocals or TestStepParms. This helps keep the symbol in scope even if you reorganize the testplan.

- If you use a variable in a flow control statement but do not specify a symbol table as part of the variable’s declaration, HP TestExec SL looks for an existing symbol with the same name in the SequenceLocals symbol table. If there is no existing symbol, one is automatically created in SequenceLocals.

To Insert a Flow Control Statement into a Testplan

1. In the left pane of the Testplan Editor window, choose the desired insertion point in your testplan.

   You can insert a statement on a blank line or into existing tests or statements. If you click to highlight an existing test or statement, the new statement will be inserted immediately preceding it.

2. Choose Insert | Other Statements in the menu bar and select the desired kind of flow control statement.

3. Use the right pane of the Testplan Editor window to enter any declarations required for the specific kind of flow control statement you chose.

Interacting with Flow Control Statements

Note

The syntax for accessing a symbol in a symbol table from a flow control statement is <*symbol table***.** *symbol*>. If you do not specify <*symbol table*>, its value defaults to SequenceLocals.

If desired, you can directly manipulate the value of a variable in a flow control statement or use the variable's value to control some aspect of testing. Then, examining or modifying the value of the symbol is the same as examining or modifying the value of the variable in the testplan.

How is this useful? Suppose you were testing a module whose stimulus—an input voltage, perhaps—needed to vary within predefined limits until the module either passed or failed. You could:

1. Execute the test for that module in a "for...next" loop, such as:

```
for Voltage = 9.9 to 10.1 step 0.1
  ModuleTest
next
```

2. In the test for the module, query the value of the counter variable and use it to vary the stimulus.

```
ModuleTest
   ...Get value of Voltage from symbol table
   ...Use value of Voltage to increment input voltage
```

Other examples of using flow control statements with symbols include:

- Branching on passing or failing tests, which are described under “To Branch on a Passing Test” and “To Branch on a Failing Test”

- Executing a test or test group only once per run of the testplan, which is described under “To Execute a Test/Test Group Once Per Testplan Run”

Using Arithmetic Operators in Flow Control Statements

If desired, you can use the arithmetic operators for addition (+), subtraction (-), multiplication (*), and division (/) in flow control statements. Also, you can use parentheses to force the order of execution of those arithmetic operators. Shown below is an example of a testplan that contains arithmetic operators in flow control statements.

```
A = (2 + 3) * 4
B = A / 5
if A - B = System.RunCount then
   test Test1
end if
```

To Branch on a Passing Test

You can use an “if...then” statement to examine the predefined TestStatus symbol in the System symbol table and programmatically implement an “on pass branch to” feature based on the results of a test; e.g.,

```
test Test1
if System.TestStatus = 1 then
   ! If Test1 passed run Test2
   test Test2
end if
test Test3
```

1. In the left pane of the Testplan Editor window, click to select the line that follows the test upon which you wish to branch.

Tip: You can click the line that follows the test even if it is blank.

2. Choose Insert | Other Statements | If...Then in the menu bar.

3. With the “if...then” statement selected in the left pane of the Testplan Editor window, specify “System.TestStatus = 1” for the value of Expression in the right pane.

4. Place any tests, test groups, or statements you wish to have executed as “branch on pass” within the boundaries of the “if...then” statement.

To Branch on a Failing Test

You can use an “if...then” statement to examine the predefined TestStatus symbol in the System symbol table and programmatically implement an “on fail branch to” feature based on the results of a test; e.g.,

```
test Test1
if System.TestStatus = 0 then
   ! If Test1 failed run Test2
   test Test2
end if
test Test3
```

Or, you can use the graphical On Fail Branch To feature that is built into each test.

Do either of the following:

1. In the left pane of the Testplan Editor window, click to select the line that follows the test upon which you wish to branch.

   Tip: You can click the line that follows the test even if it is blank.

2. Choose Insert | Other Statements | If...Then in the menu bar.

3. With the “if...then” statement selected in the left pane of the Testplan Editor window, specify “System.TestStatus = 0” for the value of Expression in the right pane.

4. Place any tests, test groups, or statements you wish to have executed as "branch on fail" within the boundaries of the "if...then" statement.

- or -

1. In either the Main or Exception sequence, click a test in the left pane of the Testplan Editor window.
2. Choose the Options tab in the right pane of the Testplan Editor window.
3. Click the arrow to the right of "On Fail Branch To" to invoke a list of tests to which the current test can branch if a failure occurs.

   The default value of "<Continue>" means that if the current test fails, the next test in the list will be executed; i.e., there is no branching.

4. Click a test in the list to select it as the desired branch.

To Branch on an Exception

1. In the left pane of the Testplan Editor window, click the arrow to the right of "Testplan Sequence".
2. Choose "Exception" in the list.
3. Add one or more tests to the list of tests for the Exception sequence.

   This list of tests will be executed if an exception occurs when executing the testplan.

4. Click the arrow to the right of "Testplan Sequence".
5. Choose "Main" in the list to return to the Main—i.e., non-exception—sequence of tests.

To Execute a Test/Test Group Once Per Testplan Run

You can use an “if...then” statement to examine the predefined RunCount symbol in the System symbol table and have specific tests, test groups, or statements executed only once each time the testplan runs; e.g.,

```
test Test1
if System.RunCount = 1 then
   ! Execute Test2 the first time the testplan is run
   test Test2
end if
test Test3
```

1. In the left pane of the Testplan Editor window, click to select a line where you wish to insert an “if...then” statement to bound one or more tests, test groups, or statements to be executed only once per testplan run.

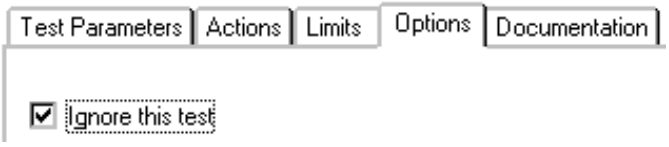
2. Choose Insert | Other Statements | If...Then in the menu bar.

3. With the “if...then” statement selected in the left pane of the Testplan Editor window, specify “System.RunCount = 1” for the value of Expression in the right pane.

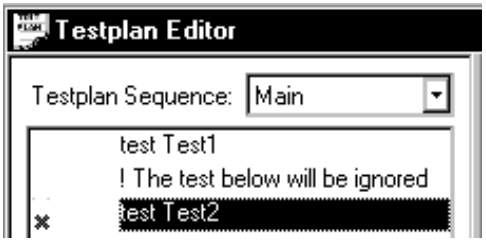
4. Place the desired tests, test groups, or statements within the boundaries of the “if...then” statement.

To Ignore a Test

If desired, you can use the “Ignore this test” feature to ignore a test when the testplan is run. Because no integrity checking is done on ignored tests, they are useful when you wish to insert non-working tests during testplan development and finish them later. Also, you can use ignored tests in conjunction with variants so that one variant of a testplan executes different tests than does another variant.



As shown below, an ignored test has a small cross beside it in the sequence of tests.



1. With a testplan loaded, in the left pane of the Testplan Editor window click to select the test to be ignored.

Note

If you are using variants, specify which variant to use before telling the Test Executive to ignore a test. For more information about variants, see "Testplan Variants" in Chapter 3 of the *Using HP TestExec SL* book.

2. Choose the Options tab in the right pane of the Testplan Editor window.

3. Check the box labeled "Ignore this test".

Running a Testplan

To Load a Testplan

Load a testplan so you can examine, modify, or run it.

1. Click  in the toolbar or choose File | Open in the menu bar.
2. Type the name of an existing testplan file (.tpa) or use the graphical browser to find an existing testplan.
3. Choose the Open button.

To Run a Testplan

Run a testplan to execute the tests in it.

1. Load the testplan, if needed.
2. (*optional*) If you wish to use a testplan variant other than the default, Normal, do the following:
   a. Click  in the toolbar or choose Options | Testplan Options in the menu bar.
   b. On the Execution tab in the right pane of the Testplan Editor window, choose the desired variant from the list under Testplan Variant.

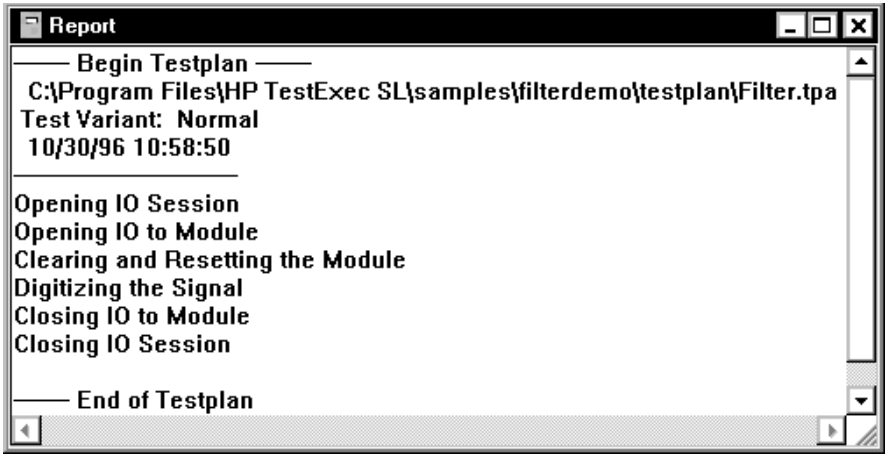
      Tip: The current variant is shown toward the right side of the status bar at the bottom of the Test Executive environment.

   c. Choose the OK button.
3. Choose  in the toolbar or choose Debug | Go in the menu bar.

Viewing What Happens as a Testplan Runs

Using the Report Window to Monitor Results

As shown below, the Report window lets you monitor the results as a testplan runs.



Tip: You may want to minimize the Report window if you wish to examine a report later but do not want the Report window appearing all the time.

To Enable/Disable the Report Window

- With a testplan loaded, click  in the toolbar or choose Window | Report in the menu bar.

A check mark appears to the left of Report in the upper region of the Window menu when the Report window is enabled.

To Specify What Appears in the Report Window

1. With a testplan loaded, click  in the toolbar or choose View | Testplan Options in the menu bar.

2. When the Options box appear, choose its Reporting tab.

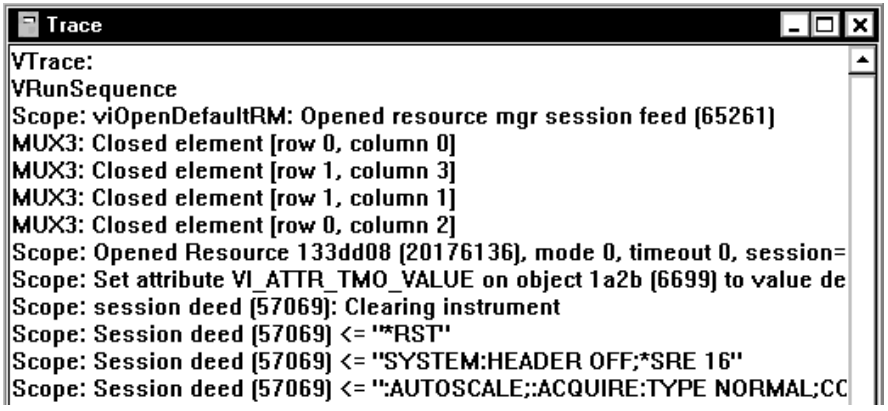
3. Enable/disable any or all of the following check boxes under Report.

Passed tests	If enabled, information about tests that pass appears in the Report window.
Failed tests	If enabled, information about tests that fail appears in the Report window.
Exceptions	If enabled, information about exceptions that occur while executing the testplan appears in the Report window.

4. Choose the OK button.

Using the Trace Window to Monitor I/O Operations

As shown below, the Trace window lets you dynamically monitor I/O operations with hardware, such as instruments and switching modules, in a test system as a testplan runs. Options associated with it let you specify when to trace tests and how much information to gather during tracing.



Trace information appears in named “streams” of information that identify the information’s source. The name of the stream is followed by a semicolon and the status message for that stream. In the example above, MUX3 is the name of a trace stream whose source is a hardware handler that controls a switching module whose logical name is “MUX3”. Status information from MUX3, such as “Closed element [row 0, column 0]”, describes what is

happening at MUX3 as the testplan runs. “Scope” is another stream in the example.

Using the Trace window is a three-step process. You must:

1. Enable the Trace window
2. Specify which tests to trace
3. Specify what kind of trace information to display for each traced test

To Enable/Disable the Trace Window

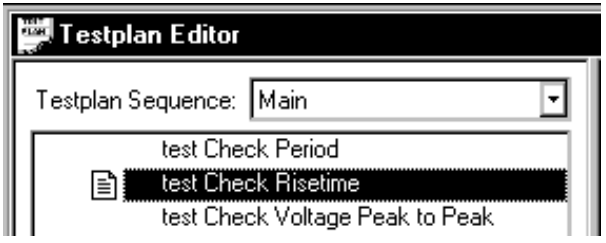
- With a testplan loaded, choose Window | Trace in the menu bar.

As shown below, a check mark appears to the left of Trace in the upper region of the Window menu when the Trace window is enabled.

To Specify Which Tests are Traced

1. With a testplan loaded, in the left pane of the Testplan Editor window choose one or more tests to be traced.
2. Choose Debug | Set Trace in the menu bar.

As shown below, a trace icon appears to the left of traced tests.



Tip: A quick way to select all tests for tracing is to choose a test in the left pane of the Testplan Editor window, type Ctrl-a or choose Edit | Select All in the menu bar, and then choose Debug | Set Trace in the menu bar.

To Specify What Appears When Tests are Traced

1. With a testplan loaded, choose Debug | Trace Settings in the menu bar.

2. Enable/disable any or all of the following items under Trace Settings. Each corresponds to a named stream of trace information.

User Trace	If enabled, user-defined trace information appears for actions in traced tests as the testplan runs. This is the default stream for trace information sent from actions. "User-defined trace information" means information programmatically sent to the Trace window from actions via API functions such as `UtaTrace()`. See the *Reference* book for more information about APIs used for tracing.
Test	If enabled, test names appear for traced tests in the Trace window as the testplan runs.
Test Details	If enabled, detailed information about traced tests appears in the Trace window as the testplan runs.
other	Some actions, hardware handlers, or instrument drivers add other stream names to the Trace settings menu. API functions such as `UtaTraceEx()` and `UtaHwModTraceEx()` let you send trace information in named streams from actions and hardware handlers, respectively. See the *Reference* book for more information about APIs used for tracing.

As shown below, a check mark appears next to the names of streams selected for tracing.

To Stop a Testplan

When you stop a testplan, execution halts when the current operation—such as executing an action—has finished.

- Choose Debug | Stop or in the menu bar or  in the toolbar.

Note If you need to halt a testplan immediately, use the Abort command instead.

To Abort a Testplan

When you abort a testplan, execution halts immediately regardless of what the testplan is doing.

- Choose Debug | Abort or in the menu bar or  in the toolbar.

Note If you wish to complete the current operation in progress—such as executing an action—before halting, use the Stop command instead.

Other Tasks Associated with Testplans

Using Global Variables in Testplans

Global variables let actions share data across tests in a testplan. The scope of a global variable can be:

- The entire testplan, which means the symbol is stored in an external symbol table or in the System symbol table.
- Restricted to a single sequence in a testplan, which means the symbol is stored in the SequenceLocals symbol table.

For detailed information about using symbols tables, see “Using Symbol Tables” in Chapter 5.

Note

By default, HP TestExec SL stores some global information in predefined symbols in the System symbol table; see “Predefined Symbols” in Chapter 5.

To Use a Global Variable Whose Scope is the Testplan

1. With a testplan loaded, use the Symbol Tables box (View | Symbol Tables) to declare a new variable in an external symbol table.

   Note: If there is no existing external symbol table to hold your global variable, use File | New and choose Symbol Table to create a new one. Then choose the Add External Symbol Table button in the Symbol Tables box to make the externally stored symbol visible to your testplan.

2. Choose the Actions tab in the right pane of the Testplan Editor window.

3. In the list of actions, choose an action that has a parameter you wish to associate with the global variable.

   Example: Name of parameter is “dutvoltage” and Value is “5”.

4. Double-click the Name column in the row that contains the parameter of interest.

5. When the Edit Symbol box appears, enable Reference a Symbol if it is not already enabled.

6. Select the desired external symbol table from the Search list.

7. Use the Reference list to select the name of the global variable.

8. Choose the OK button.

Example: Value of parameter “dutvoltage” now is “@ExtSymTable.dutvoltage”; i.e., the value of the parameter is determined by the value of variable “dutvoltage” in the ExtSymTable symbol table.

To Use a Global Variable Whose Scope is a Sequence

1. With a testplan loaded, in the left pane of the Testplan Editor window choose a sequence—Main or Exception—in which to use the global variable.

2. Choose the Actions tab in the right pane of the Testplan Editor window.

3. In the list of actions, choose an action that has a parameter you wish to associate with the global variable.

   Example: Name of parameter is “dutvoltage” and Value is “5”.

4. Double-click the Name of the parameter.

5. When the Edit Symbol box appears, enable Reference a Symbol if it is not already enabled.

6. Select the SequenceLocals symbol table from the Search list.

7. Use the Reference list to select the name of the global variable.

8. Choose the OK button.

Example: Value of parameter “dutvoltage” now is “@SequenceLocals.dutvoltage”; i.e., the value of the parameter is determined by the value of variable “dutvoltage” in the SequenceLocals symbol table.

To Specify the Global Options for a Testplan

1. With a testplan loaded, click  in the toolbar or choose Options | Testplan Options in the menu bar.

2. Use the features on the various tabs in the Testplan Options box to specify the global options for the current testplan.

To Specify Which Topology Files to Use

1. With a testplan loaded, choose Options | Switching Topology Files in the menu bar.

2. Type the name of a topology file for the fixture layer or click the associated Browse button and use the graphical browser to choose a file.

3. Type the name of a topology file for the UUT layer or click the associated Browse button and use the graphical browser to choose a file.

4. Choose the OK button.

Note Topology files have a “.ust” extension; e.g., “fixture1.ust”.

Using Testplans & UUTs with an Operator Interface

To Register a Testplan for an Operator Interface

A typical operator interface lets production operators choose from a list of testplans to run. You must manually edit file “tstexcsl.ini” to specify which testplans appear in the list, which variant is chosen by default, and a brief description of what the testplan does.

1. Open file “tstexcsl.ini” (in directory “<*HP TestExec SL home*>\bin”) with a text editor, such as WordPad in its text mode.

2. Add entries for one or more testplans to the [Testplan Reg] section of the file.

Note The file contains descriptive comments about the formats of these entries.

3. Save the updated file and exit the editor.

To Register a UUT for an Operator Interface

Some operator interfaces let production operators use a bar code reader to scan the information for a UUT, and then parse the bar code to automatically load the appropriate testplan. If your operator interface supports this feature, you must manually edit file “tstexcsl.ini” to specify the association between UUTs and testplans.

1. Open file “tstexcsl.ini” (in directory “<*HP TestExec SL home*>\bin”) with a text editor, such as WordPad in its text mode.

2. Add entries for one or more UUTs to the [UUT Reg] section of the file.

Note The file contains descriptive comments about the formats of these entries.

3. Save the updated file and exit the editor.

Using Variants in Testplans

Variants let you create named variations on the contents of a testplan. After you create a testplan’s variants, you can specify the parameters and limits for the tests and test groups in each variant. Because they let you use *one* testplan with *n* different sets of test limits and parameters, variants are useful where one UUT is very similar to another except for slightly different values for its test limits or parameters.

To Add a Variant to a Testplan

1. With a testplan loaded, choose Options | Variants in the menu bar.
2. When the Test Variants box appears, choose the Add button.
3. In the Add Variant box, type a name for the new variant in the field under New Variant.
4. Choose a template for the new variant from the list of existing variants shown under Based On.

   Tip: Base the new variant on whichever existing variant is most like the new one.
5. Choose the OK button.

For information about specifying the contents of variants after you have created them, see “Specifying Variations on Tests/Test Groups When Using Variants” in Chapter 2.

To Rename a Variant in a Testplan

1. With a testplan loaded, choose Options | Variants in the menu bar.
2. When the Test Variants box appears, click the name of an existing variant in the list under Current Variants.
3. Choose the Rename button.
4. In the Rename Variant box, choose the name of an existing variant from the list shown under Variant Name.
5. Type a new name for the variant in the field under New Name.
6. Choose the OK button.

To Delete a Variant from a Testplan

1. With a testplan loaded, choose Options | Variants in the menu bar.

2. When the Test Variants box appears, click the name of an existing variant in the list under Current Variants.

   Note: You cannot delete Normal, which is the default variant.

3. Choose the Delete button.

4. Choose the OK button.

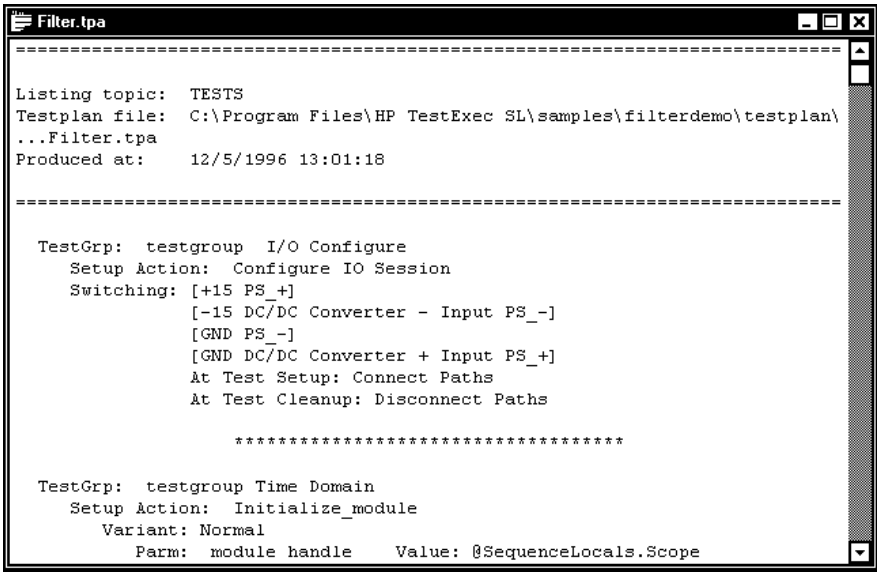
To Examine All the Variants for a Testplan

You can examine all the variants of a testplan while globally viewing or modifying the test limits; see “To View the Limits for Tests in a Testplan” in Chapter 2.

Examining Testplans & System Information

Overview

The Listing window lets you view or print information about various aspects of your testplans and hardware controlled by your test system. The example below shows how you can view a descriptive listing of the tests in a testplan.



Which Kinds of Information Can I Examine?

The categories of information you can examine or print in the Listing window include:

Actions	Lists detailed information about actions in the current testplan, including action names, source file names, and routine names
Symbol tables	Lists the symbols used in symbol tables in the current testplan.

Testplan Audit	Lists auditing information for the current testplan
Testplan	Lists detailed information about the current testplan, including test groups, tests, actions, variants, and run options.
Tests	Lists detailed information about tests in the current testplan, including test names, actions, variants, source files names, and routine names.
Adjacencies	Lists all topology adjacencies—i.e., nodes separated by a switching element—for the current testplan, including preferred node names, adjacency names, module names, and switching elements and their positions.
Node Labels	Lists all node labels for the current testplan, including label names, preferred node names that are aliased, descriptions, and keywords.
Instruments	Lists information about instruments controlled by the current testplan.
Switches	Lists information about switching hardware controlled by the current testplan.
Fixture Layer	Lists topology information about connections on the fixture topology layer, which includes aliases, wires, and modules.
System Layer	Lists topology information about connections on the system topology layer, which includes aliases, wires, and modules.
UUT Layer	Lists topology information about connections on the UUT topology layer, which includes aliases, wires, and modules.

To List Testplans & System Information

1. Choose View | Listing in the menu bar.

2. Choose which type of listing to view.

To Print Listings of Testplans & System Information

1. Choose View | Listing in the menu bar.
2. Choose which type of listing to view.
3. Click  in the toolbar or choose File | Print in the menu bar.
4. Set the printing options as desired.
5. Choose the OK button.

Tip: You can use File | Print Preview in the menu bar to see how a listing will look before printing it.

To Find Specific Text in Testplans & Listings

If desired, you can search testplans or any of the various listings of system information for a specific word or phrase.

1. Do either of the following:
   - If you wish to search a testplan, with a testplan loaded click in the left pane of the Testplan Editor window.
   - If you wish to search a listing, generate the listing as described earlier in “To List Testplans & System Information.”
2. Click  in the toolbar or choose Edit | Find in the menu bar.
3. In the “Find what” field, specify the text you wish to search for.

   Tip: Check the “Match case” box if you wish to search for exactly the same pattern of upper and lowercase characters specified in the “Find what” field.
4. Choose the Find Next button.

Debugging Testplans

As you develop testplans and their components you need to verify their operation and any fix problems that arise. HP TestExec SL's debug features let you interact with testplans and their components as they execute.

If you are using C/C++ to develop actions, also see "Debugging C/C++ Actions" in Chapter 3.

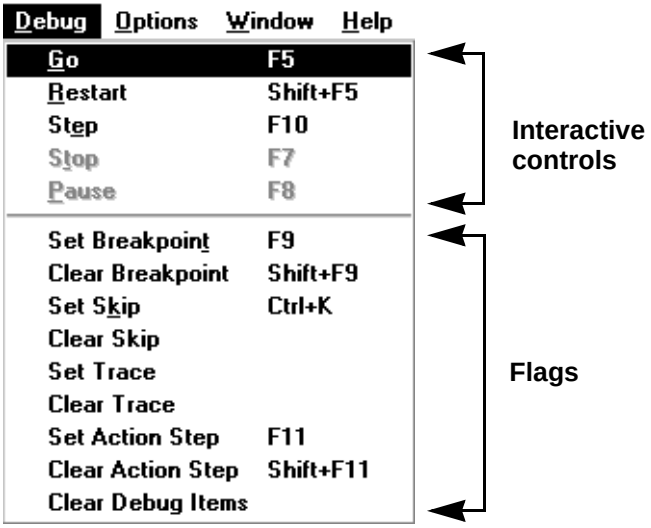
Using Interactive Controls & Flags

Once started, a testplan normally runs from beginning to end, executing tests in the order in which they appear in it. However, the Test Executive provides several features you can use to modify the running of a testplan. These features can be particularly useful when you are debugging a testplan or test, or when you need to stop or pause the testplan at a specific place while troubleshooting a UUT.

There are two main kinds of features you can use to control testplans:

Interactive Controls	These are features such as Stop/Continue, Restart, Step, Stop, and Pause. They are interactive insofar as using them causes an immediate response.
Flags	You can set "flags"—i.e., markers—in the testplan. A flag is acted upon if it is encountered as the testplan runs. You can set a flag that marks a test to be stopped upon, skipped, traced, or have its actions single-stepped. Also, you can clear an individual flag or clear all flags for selected tests.

As shown below, these features appear as options under the Debug menu in the menu bar.



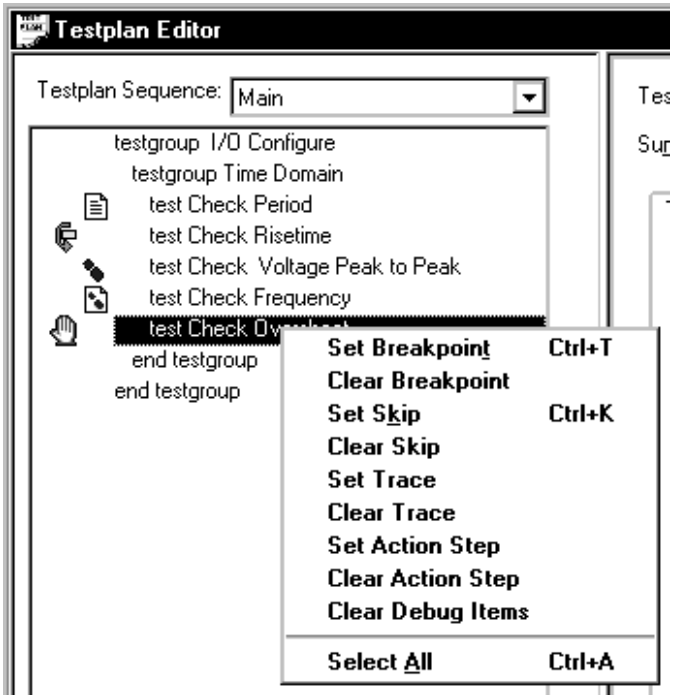
When you use the Debug menu's options to set a flag for a test in a testplan, one of the icons shown below appears to the left of the test.

This icon...	Means that...
	A **breakpoint** has been set for the test, which means the testplan will execute until the breakpoint is encountered, and then stop executing immediately before the marked test.
	Items marked in the testplan will be **skip**ped; i.e., the testplan will not execute the marked items. Be aware that skipping a test is not the same as ignoring it (see "Ignoring a Test" earlier in this chapter); the overall integrity of skipped tests is checked, but that of ignored tests is not.
	The test will be **trace**d, which means that status information will appear in the Trace window as the test executes.

Actions in the marked test will be **single-step**ped. The testplan will pause at the first action in the test, and you can use either the Step command in the Debug menu or the  icon in the toolbar to execute the test's actions one at a time.

A combination of the **trace** *and* **single-step** icons; i.e., the marked test will be traced as you single-step through it.

As a shortcut when setting flags, you can select a test in the left pane of the Testplan Editor window and then right-click to invoke the menu shown below.



Tip: If desired, you can select multiple tests in a testplan and simultaneously set or clear all of their flags.

Caution

If you add flags and then save a testplan, the flags are saved with it. Be sure to remove flags from testplans before releasing them to production. For

example, a breakpoint flag can cause the testplan to stop executing prematurely and leave the operator interface “hung.”

Single-Stepping in a Testplan

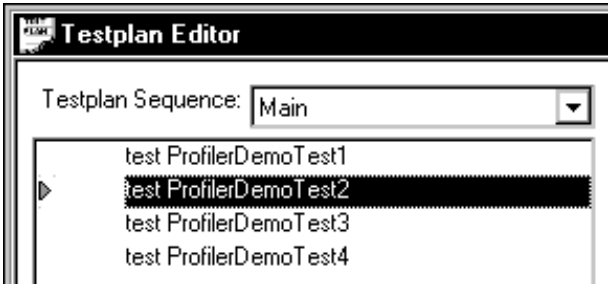
Single-stepping in a testplan lets you pause as needed to verify that tests and actions are working correctly.

Single-Stepping Through Tests

Overview

If desired, you can single-step through the tests in a testplan. Each time you single-step, the testplan executes one test, halts, and then displays a pointer icon that identifies the next test to be executed.

In the example below, test ProfilerDemoTest1 has been executed and the testplan has halted pending execution of test ProfilerDemoTest2.



To Single-Step Through the Tests in a Testplan

- With a testplan loaded, click  in the toolbar or choose Debug | Step Test in the menu bar.

To Cancel Single-Stepping Through the Tests in a Testplan

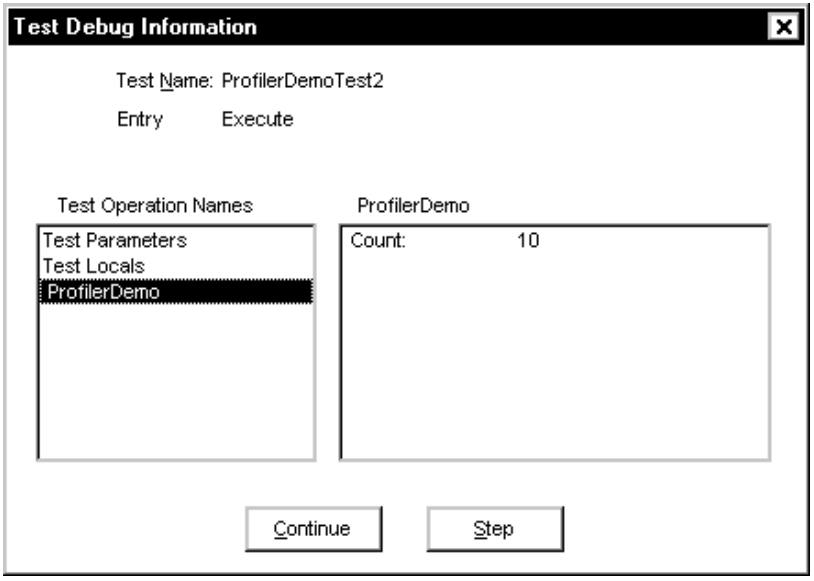
- While single-stepping through a testplan, click  in the toolbar or choose Debug | Stop in the menu bar.

Single-Stepping Through Actions

Overview

Each test in a testplan contains one or more actions. If desired, you can single-step through the actions. This can be useful if you wish to verify the results of each action as a test executes. For example, you could connect test equipment to the UUT, pause on a specific action, and verify that the action is interacting correctly with the UUT.

When the testplan is paused while single-stepping through actions, the Test Debug Information box shown below appears.



Here, the test's name is ProfileDemoTest2 and it contains an execute action named ProfilerDemo that uses a parameter named Count whose value is 10. The test is paused on ProfilerDemo.

To Single-Step Through Actions

1. With a testplan loaded, in the left pane of the Testplan Editor window click a test whose actions you wish to step through one at a time.

2. Choose Debug | Set Action Step in the menu bar or right-click and choose Set Action Setup from the menu that appears.

3. Run the testplan as usual.

4. When the test pauses on an action and the Test Debug Information box appears, make debugging measurements or select an item in the list under Test Operation Names and examine its characteristics.

5. Do one of the following:

   - To single-step to the next action in the test (if the test contains more than one action), choose the Step button.

     - or -

   - To proceed to the next test without single-stepping through any more actions in the current test, choose the Continue button.

     - or -

   - To stop after executing the current test, choose  in the toolbar and then choose the Continue button.

6. When you have finished single-stepping, clear the flags used to mark the tests.

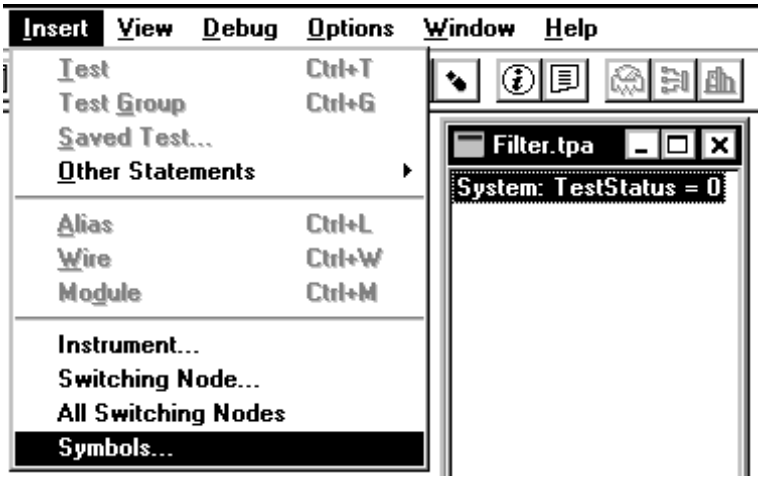
Using the Watch Window to Aid Debugging

Overview

Many programming environments provide a “watch” feature that lets you examine the values of variables and expressions while debugging programs. In a similar fashion, HP TestExec SL lets you specify items such as symbols, instruments[1], or switching paths to be watched when debugging a testplan. You use the Insert menu to place these items into the Watch window, as

1. You can watch instruments only when using specific driver software from Hewlett-Packard.

shown below, and then examine them when the testplan is paused, such as while single-stepping through actions.



The name of the symbol table in which a symbol resides is prefixed to the name of the symbol. In the example above, the symbol named TestStatus appears in the symbol table named System—i.e., System: TestStatus—and its current value is zero.

Note

To ensure that testplans execute rapidly, the Watch window is updated only when testplan execution pauses or stops.

To Insert a Symbol into the Watch Window

1. With a testplan loaded, make sure the Watch window is active; i.e., its border is highlighted.

   If the Watch window is not visible, choose Window | Watch. If the Watch window is visible but inactive, click its border to make it active.

2. Choose Insert | Symbols in the menu bar.

3. When the Select Symbol to Watch box appears, do the following in it:

   a. Choose a symbol table from the list under Available Tables.

b. Choose a symbol from the list under Available Symbols.

c. Choose the OK button.

For more information about symbol tables, see “Using Symbol Tables” in Chapter 5.

To Insert a Switching Node into the Watch Window

1. With a testplan loaded, make sure the Watch window is active; i.e., its border is highlighted.

   If the Watch window is not visible, choose Window | Watch. If the Watch window is visible but inactive, clicks its border to make it active.

2. Choose Insert | Switching Node in the menu bar.

   Tip: As a shortcut when setting watches on all switching nodes, choose Insert | All Switching Nodes.

3. When the Select Switching Node box appears, do the following in it:

   a. Choose a node from the list.

      Tip: If desired, you reduce the number of nodes that appear in the list by choosing a Filter from the list.

      Tip: If desired, you can sort the list of nodes by selecting the Sort Node Names check box.

   b. Choose the OK button.

For more information about switching nodes, see “About Switching Topology” in Chapter 3 of the *Getting Started* book.

To Insert an Instrument into the Watch Window

Note This feature is enabled only when using specific instrument drivers provided by Hewlett-Packard.

1. With a testplan loaded, make sure the Watch window is active; i.e., its border is highlighted.

   If the Watch window is not visible, choose Window | Watch. If the Watch window is visible but inactive, click its border to make it active.

2. Choose Insert | Instrument in the menu bar.

3. When the Select Instrument box appears, do the following in it:

   a. Choose an instrument from the list.

   b. Choose the OK button.

To Remove an Item from the Watch Window

1. In the Watch window, select the item to be removed.

   If the Watch window is not visible, choose Window | Watch. If the Watch window is visible but inactive, clicks its border to make it active.

2. Choose Edit | Delete in the menu bar.

Fine-Tuning Testplans

A testplan is only as good as the tests in it. Good tests are fast, reliable, and accurate. After you have your tests and testplan running, you may want to consider taking the steps described in the following topics to fine-tune your results.

Optimizing the Reliability of Testplans

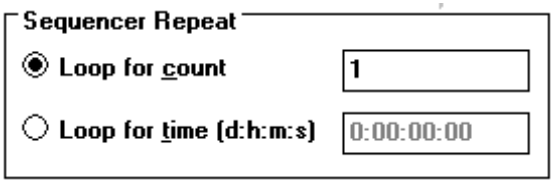
Several ways to improve the reliability of your testplans are:

- Debug known problems in actions and tests as needed.

  For example, you can use the debugging features of the language used to create actions to debug actions. And you can use features in the Test Executive that control the running of testplans to pause on a test, skip a test, and such while debugging tests.

- Run testplans for a prolonged period, such as overnight, to verify the reliability of the tests in them.[1]

  Tip: To run repetitively a testplan, use the “Loop for count” or “Loop for time” options under Sequencer Repeat on the Execution tab in the Testplan Options box (Options | Testplan Options).



- Run testplans with datalogging on and examine the results for consistency.

1. If you do this, you may want to turn off datalogging to prevent log records from potentially filling your hard disk.

For example, you might turn on datalogging and run the testplan to collect data about a single UUT or a group of UUTs. If the data are inconsistent, try to identify which test(s) is the problem and then fix it.

- Deliberately stress your testplan by introducing conditions that can cause exceptions, and add fixes as needed.

  For example, you might see what happens if an instrument “times out” without returning a reading. Or, you might deliberately test UUTs whose performance is grossly outside the normal limits.

Optimizing the Throughput of Testplans

Suggested Ways to Make Testplans Run Faster

Some ways in which you can make your testplans execute faster are:

- Use test groups to do slow actions outside of tests or to eliminate redundant tasks.

  If you have a group of tests whose setup/cleanup needs are alike, insert those tasks once, at the beginning of a test group that includes the group of tests, instead of inside each test. An example of this might be initializing power supplies or setting up instruments that require similar setups for more than one test. If several tests require positive sources, do the tests as a group. Or, if several tests require the same UUT setting, do the tests as a group.

- Use triggers for fast synchronization of tests.

  For example, avoid synchronizing to slow cycle waveforms. Also, avoid controller-induced test delays.

- Find faster ways to do tests.

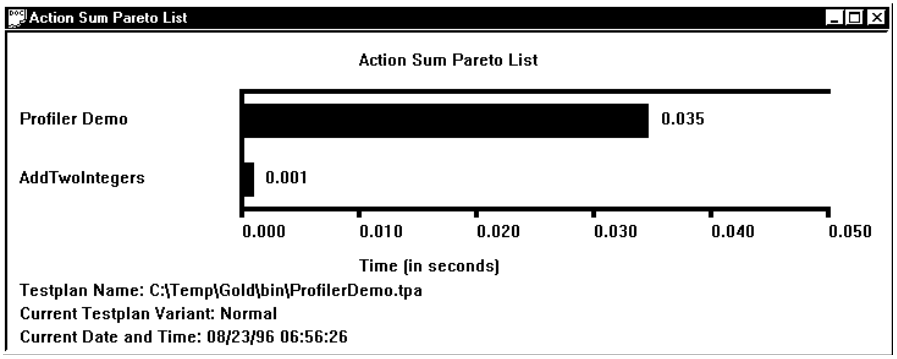
  For example, use a DMM instead of a slower digitizer.

- Use HP TestExec SL’s profiler feature (described below) to optimize the actions inside tests in a testplan.

Using the Profiler to Optimize Testplans

HP TestExec SL includes a profiler you can use to see how long each action or test group in a testplan takes to execute. Once you know how long each action or test group takes to execute, you can decide where to begin the "tuning" process, and monitor any improvements you make.

After enabling the profiler, you run a testplan to collect data, and then either view Pareto charts directly in HP TestExec SL or use a financial spreadsheet program to further analyze the data. As shown below, the profiler display in HP TestExec SL lists actions or test groups in order from slowest to fastest, and shows how long each took to complete.



Each time you run the testplan, profiler data from the previous run is discarded. If a testplan aborts, its profiler data is lost. Also, the profiler is automatically turned off whenever you exit a testplan.

Note

Because the profiler can significantly degrade HP TestExec SL's performance, you probably will not want to run it during production testing.

To Set Up the Profiler

Before you can use the profiler, you must enable it.

1. Choose Options | Testplan Options in the menu bar.
2. In the Testplan Options box, choose the Profiler tab.
3. Enable the Enable Profiler check box.

4. If, besides viewing the profiler data in HP TestExec SL, you want to save the data in a tab-delimited file for subsequent analysis, such as in a spreadsheet, do the following:

   a. Select the Save to File check box.

   b. Either type the name of a file in the data entry field or choose the Browse button and use the graphical browser to specify a name for the file in which the profiling data will be saved.

5. Choose the OK button.

To Run the Profiler

- With the profiler enabled, run the testplan as usual.

  As the testplan runs with the profiler enabled, HP TestExec SL collects data about the testplan.

To View Profiler Results in HP TestExec SL

1. After running the testplan with the profiler enabled to collect data, choose View | Profiler Results in the menu bar.

2. Choose how you would like to see the data displayed.

   Formats for displaying profiler data in Pareto charts include:

Sum of Action Execution Times	Total time that actions in the testplan took to execute. If an action is used more than once, this will be its accumulated time.
Average Action Execution Times	Average time that actions in the testplan took to execute. If an action is used more than once, this will be the arithmetic mean of each execution time.

Sum of Test Execution Times	Total time that tests in the testplan took to execute.
Average of Test Execution Times	Average time that tests in the testplan took to execute.

3. If you wish to limit the amount of data that appears, specify an alternate value for Maximum Number of Items to Display.

4. Choose the OK button.

   Tip: If desired, you can simultaneously view other types of Pareto charts by choosing Profiler Pareto from the menu bar and choosing another type when the viewer is active.

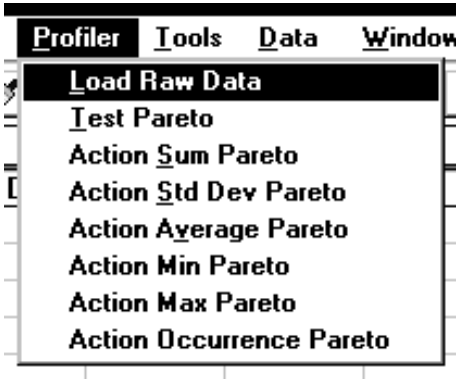
   Tip: If desired, you can use File | Print Graph to print the results when the viewer is active.

To View Profiler Results in a Spreadsheet

When you use the profiler's Write to File option and specify a file name, data is saved in a tab-delimited format suitable for examination with a spreadsheet.

Hewlett-Packard also provides a worksheet ("profile.xls") and an add-in ("profile.xla") you can use with Microsoft Excel as the starting point in examining the data file's contents. These files are located in directory "*<HP TestExec SL home>*\samples\excelmacros". As shown below, loading

either of these files adds a Profiler option and related menu items to Excel's menu bar.



Moving a Testplan

You may want to develop testplans on a central development system that is fully configured even if you intend to use them elsewhere. That way, not every test system needs a full set of hardware resources for compatibility; i.e., each destination system needs only the subset of the development system's resources that are required to run a specific testplan.

Once you have developed and debugged a new testplan on the development system, you probably will want to release it to your production environment. For example, if you intend to run the testplan on more than one test system, you must copy the appropriate files to other systems. Also, you probably will want to make a backup copy of the completed testplan "just in case."

Do the following to move a testplan from your development system to another system:[1]

- Be sure the destination system has all the hardware resources needed to run the testplan.
- Copy the testplan file—i.e., "*testplan_name*.tpa"—to the destination system.
- Be sure all the files used by actions in your testplan exist on the destination system. These include "*.umd" files and executable libraries.

  Tip: You can use View | Listing | Actions to list the contents of actions in a testplan. Or, you can use an audit listing to show all the files used by a testplan.
- Copy the topology files for the fixture and UUT layers ("fixture.ust" and "uut.ust" files or equivalent) to the destination system.
- If external symbol tables are associated with the testplan, copy them ("*.sym" files) to the destination system.

1. The directory structure on the destination system can be different from the directory structure on the development system.

- Verify that the datalogging options are the same across the systems:
  - Be sure the [Data Log] section in the “tstexcsl.ini” file on the destination system identifies the format and definition files you wish to use when datalogging.
  - Be sure the datalogging options for the testplan (Options | Testplan Options | Reporting) reflect the settings you wish to use on the destination system.
  - Be sure the destination system's topology file for the system layer (“system.ust”) is the same as or a superset of the file on the development system.
  - Be sure to remove any flags, such as skipped tests or breakpoints, if you are moving the testplan to a system used for production testing.

Caution

Flags left in the testplan can cause the operator interface to behave incorrectly. For example, a breakpoint flag can cause the testplan to stop executing prematurely and leave the operator interface “hung.”

For more information about flags, see “Using Interactive Controls & Flags.“

For suggestions about setting up library search paths to optimize the portability of testplans, see “Using Search Paths to Improve Testplan Portability” in Chapter 5.