Qualification Verifier Exerciser Tool (Q-VET)

User's Guide

Part Number: EK-QVEUN-UG. A01

May 1998

This manual discusses how to use the Qualification Verifier Exerciser Tool (Q-VET) to exercise DIGITAL hardware and the DIGITAL UNIX, OpenVMS and Windows NT operating systems.

Revision/Update Information: This is a new manual.

Operating System and Version: DIGITAL UNIX, OpenVMS, Windows NT

Digital Equipment Corporation
Maynard, Massachusetts

May 1998

Table of Contents

Figures

Tables

Preface

The *Qualification Verifier Exerciser Tool* (*Q-VET) User's Guide* discusses how to use Q-VET for system and option testing and quality assurance. on systems running three different operating systems:

- DIGITAL UNIX using the DECwindows Motif and CDE interface
- OpenVMS
- Windows NT

The Q-VET interfaces for the DIGITAL UNIX and OpenVMS operating systems have been combined in this manual due to their great similarities. The interface to Windows NT is depicted in its own chapter.

This is the first release of combined documentation for Q-VET; it is a general overview and guide to Q-VET, and does not include extensive details about individual tests.

Intended Audience

This guide is intended for who use Q-VET for any of the following:

- System and options qualification
- Verifying installation and system acceptance
- Maintaining and troubleshooting installed systems
- Testing system integration
- Performing quality assurance
- Testing device drivers
- Performing load and stress testing
- Certifying hardware during the manufacturing process

Document Structure

The *Qualification Verifier Exerciser Tool* (*Q-VET) User's Guide* consists of four chapters, six appendices, and glossary organized as follows:

- Chapter 1 is an overview of Q-VET and the standard exerciser tests that it contains to facilitate system maintenance and quality assurance.
- Chapter 2 shows how to use Q-VET with the DIGITAL UNIX or OpenVMS operating systems.
- Chapter 3 shows how to use Q-VET with the Windows NT operating system.
- Chapter 4 shows how to use Q-VET with the Command Interface.
- Appendix A describes the types of messages Q-VET displays and lists and explains frequently displayed messages.
- Appendix B lists and explains option information that you can supply to generic Q-VET exercisers and discusses the detail of error reports and the summary information that each exerciser displays.
- Appendix C provides a comparison of equivalent menu entries and commands.
- Appendix D Describes the available device tests.
- Appendix E Describes the jacketed test device and it’s use in running tests foriegn to Q-VET.
- Appendix F describes the data patterns for various tests.
- The Glossary lists and defines specific Q-VET terms.

Conventions

The following conventions are used in this manual:

Convention Example	**Description**
MBR, MBL	MBR refers to the rightmost button on the mouse; MBL refers to the leftmost button.
`command input`	This bold typeface is used to indicate Q-VET commands as they should be entered by the user.
`system output`	This typeface is used to indicate system output or the exact name of a command, option, partition, path name, directory, or file.

Italic typeface	This typeface is used to indicate variables that must be defined by the user (i.e. path name, directory, file name). Also, titles of information sources are in italic, and occasionally italic is used for emphasis in the text. Italics such as *n* or *x* are used to indicate numeric variables.
Braces **{ }**	Surround a list of arguments from which you may choose one.
Brackets **[ ]**	Surround optional arguments.
[Ctrl/X]	A key sequence such as Ctrl/X indicates that you must hold down the key labeled Ctrl while you press another key or a pointing device button.
[Enter]	Indicates a key on your keyboard.

1 Qualification Verifier Exerciser Tool (Q-VET) Overview

Q-VET Overview

The Qualification Verifier Exerciser Tool (Q-VET) evolved from the DIGITAL DEC-VET project. Q-VET is a state-of-the-art tool suite developed and supported primarily for use in system and option Qualifications, ECO Verifications and operating system/driver Exercising in engineering and manufacturing environments.

The complete kit incorprates both internal and external test tools which are controlled by the Q-VET test manager. Q-VET runs at a much higher stress level than the original DEC-VET product. It is not intended for use by end-user customers as an installation verifier.

Q-VET provides a consistent human interface or GUI which allows the user to exercise the hardware and software in the same way on every system. It can also be controlled via the serial console on DIGITAL UNIX and OpenVMS systems.

Q-VET supports various exerciser configurations, ranging from a single device to full system loading in a distributed testing environment. Exercisers can run either sequentially or concurrently.

Q-VET supplies a set of standard qualification scripts and menus for ease of use and repeatability. Custom scripts can be written by any user.

This chapter discusses:

- New Q-VET Features
- The Q-VET Functionality
- Interfacing with Q-VET
- Generic Exercisers

New Q-VET Features

The following are some of the features in Q-VET (which were not available in DEC-VET).

- Qual testing scripts selectable from the GUI or from serial console.
- Automatic configuration of testing scripts, at Q-VET start time (drive/partition select, etc.)
- DumpMedia utility for examining data directly on drives (for error checking)
- Automatic re-format/re-disklabel after completing drive write testing.
- Automatic system tuning during DIGITAL UNIX and OpenVMS installation.
- Does NOT prompt for verification of 'write enable' before write testing!
- 21 additional memory, disk, file and network data patterns.
- Extended coverage (blocks, packets, etc) of hard drives, memory, network and file tests.
- Multi-screen/head video support.
- Defaults to testing 100% of physical memory amount. (DEC-VET is 5% of the VA swap space)
- 4 additional memory tests
- 5 (6 for NT) additional CPU tests including various floating point screens.
- 2 additional disk tests on DIGITAL UNIX and Windows NT, 1 additional on OpenVMS
- 2 additional tape tests on DIGITAL UNIX (tapex and tapex-compression)
- 1 additional video test on DIGITAL UNIX and OpenVMS (mpc)
- 1 additional terminal test (scrolling 'H' used for FCC/VDE) (DIGITAL UNIX and OpenVMS)
- Q-VET runs x11perf, ico and mpc as a graphics and system loads (DIGITAL UNIX and OpenVMS)
- 'crash_test' system crash dump check/test.
- 'boot_loop' system cycle test
- Support for Intel Windows NT
- 64 bit and 32 bit memory data tests (32 bit for NT, 64 bit when Windows NT supports it)

- Automatic summary report generation, which includes a scan of the system error/event logs during the test (shows all tests and errors)
- For NT, some of the Microsoft HCTs are launched as additional tests within the various exercises i.e. HCT CoProcessor test will be run as test 10 of the CPU exerciser, the HCT Virtual Memory test will be run as test 5 of the Q-VET memory exerciser.
- "jacketed_test" device for running most any external test from Q-VET

Q-VET Functionality

Q-VET can be used to:

- Qualify systems and options
- Test system integration
- Perform quality assurance
- Test device drivers
- Perform load and stress tests on single devices, systems, and networks
- Certify hardware during the manufacturing process

Interfacing With Q-VET

Q-VET provides a consistent user interface to all exercisers through either the command line or DECwindows Motif interface for DIGITAL UNIX, or through Windows NT. Q-VET controls all exerciser runs, allowing several exercisers to run at once on the same or separate devices. It also controls exercisers running on other nodes, when those nodes are connected through Q-VET to the controlling node.

Q-VET allows the user to specify the following for each run:

- Devices to be tested and nodes on which they reside
- Number of exercisers run on each device and specific exerciser options
- Length of time exercisers are run
- Stopping or looping when an error is encountered
- Level of detail in error reports

Working from a single node, the user can start, suspend, resume, or terminate exercisers and obtain information for each exerciser, regardless of the node on which it is running.

Generic Exercisers

Q-VET comes with a standard set of exercisers, as follows (see Appendix B for additional information on these exercisers):

- CPU Exerciser
  Tests typical system processor functions including binary operations, integer computations, floating-point computations, and data conversion.
- Memory Exerciser
  Tests dynamic allocation and deallocation of virtual memory and verifies test patterns written.
- Disk Exerciser
  Tests logical and physical disk I/O by performing read and write operations and verifies test patterns written.
- File Exerciser
  Tests reading and writing to ordinary disk files and verifies test patterns written.
- Tape Exerciser
  Tests reading and writing to tape device files (including file mark detection, spacing, rewind, end-of-tape detection) and verifies test patterns written.
- Network Exerciser
  Tests underlying protocol (including caches, buffers, and queues), physical network adapters, local and remote networks, destination adapters, network services, and echo daemons for TCP/IP and/or DECnet.
- Printer Exerciser
  Prints out a file containing a test pattern of all the ascii characters from " " (blank space) through "~". This pattern is shifted one place each time on each subsequent line. Two pages are printed out, totaling more lines than the 95 available patterns. Thus it verifies that all ascii characters can be printed at each position. It also offers the option to print out a special postscript file displaying a vertical shaded bar graph, as well as the ability to print out user specified ascii and postscript files.
- Terminal Exercisers (Two Tests)

  The first terminal exerciser test displays to the terminal screen a file containing a test pattern of all the ascii characters from " " (blank space) through "~". This pattern is shifted one place each time on each subsequent line. Thus it verifies that all ascii characters can be printed at each position during its 95 lines of patterns.

  The second terminal exerciser test utilizes scrolling “H’s” for FCC testing.
- Video Exerciser
  Displays various video test patterns and graphics. These verify the console's ability to

accurately display graphics, text, and shades of color. The user is required to indicate any failures if the screen display does not appear to look as it should.

- Boot Loop Exerciser
  Tests system booting capabilities and configurations by automatically rebooting the system.
- Crash Test Exerciser
  Verifies system panic dumping of memory. Also used for memory testing (addressing) and drive testing.
- X11exer Exerciser[1]
  Performs three Xwindows-based video tests; promarily used for video DMA loading.
- Jacketed Device Exerciser
  A general purpose test control psuedo-device which can be set up to start/stop, and error-check almost any type of program or test. See Appendix E.
- Do_it Exerciser[2]
  This is a DMA burst generator which uses disk and video in short bursts. It is used in system and memory testing.

[1] Available for DIGITAL UNIX and OpenVMS only.

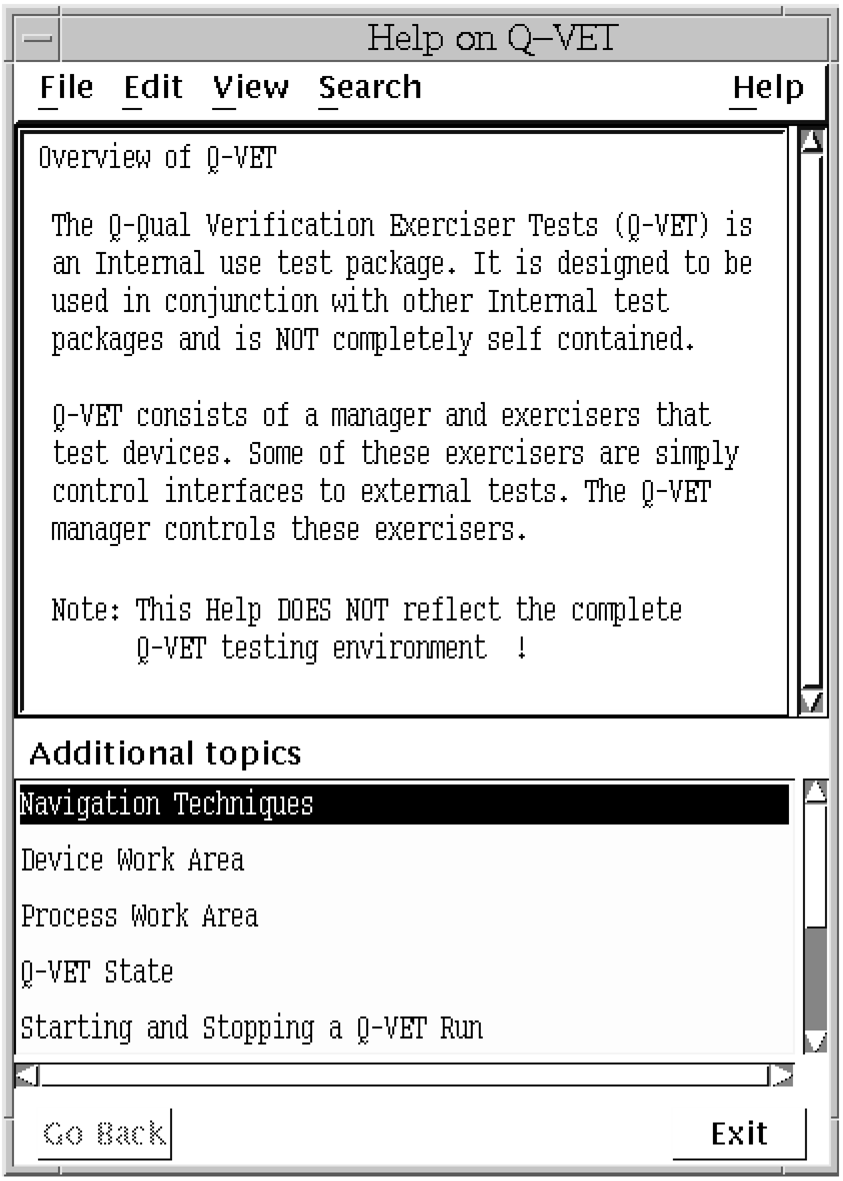
[2] Available for DIGITAL UNIX and OpenVMS only.

2 DIGITAL UNIX and OpenVMS Q-VET

This chapter gets you started with using Q-VET on the DIGITAL UNIX and OpenVMS operating systems by providing easy-to-follow examples of its use. (If you are using the Windows NT operating system, see Chapter 3.)

Online Help

Almost all Q-VET windows and dialog boxes provide a Help button. You can click on any Help button at any time to view a Help display corresponding to that window or box, or you can click on Help on the Main Window menu bar. Most Help window topics also have subtopics that you can view by double-clicking on the subtopic name, as shown in the following example.



Overview of Q-VET

The Q-Qual Verification Exerciser Tests (Q-VET) is an Internal use test package. It is designed to be used in conjunction with other Internal test packages and is NOT completely self contained.

Q-VET consists of a manager and exercisers that test devices. Some of these exercisers are simply control interfaces to external tests. The Q-VET manager controls these exercisers.

Note: This Help DOES NOT reflect the complete
      Q-VET testing environment !

Additional topics

Navigation Techniques
Device Work Area
Process Work Area
Q-VET State
Starting and Stopping a Q-VET Run

Getting Started with the DIGITAL UNIX DECwindows Motif Interface

This section reproduces a Q-VET session using the DECwindows Motif interface on the DIGITAL UNIX and the OpenVMS operating systems. The session has been designed so that you can follow the examples.

The session does not use every available function. To obtain more information on an item, click on the Help button in the menu bar or in the window in which the item appears. A Help window with detailed information is displayed.

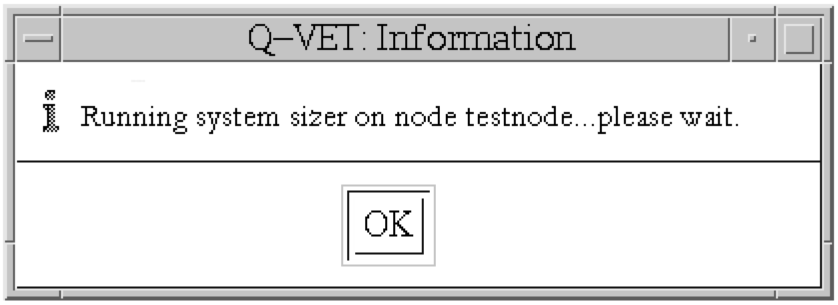
Starting a Q-VET Session

Issue the **vet** command at the system prompt:

```
%  vet &
```

Note: the “&” is recommended, but not a requirement.

A Q-VET information box should be displayed indicating that the system sizer is running.



Q–VET: Information

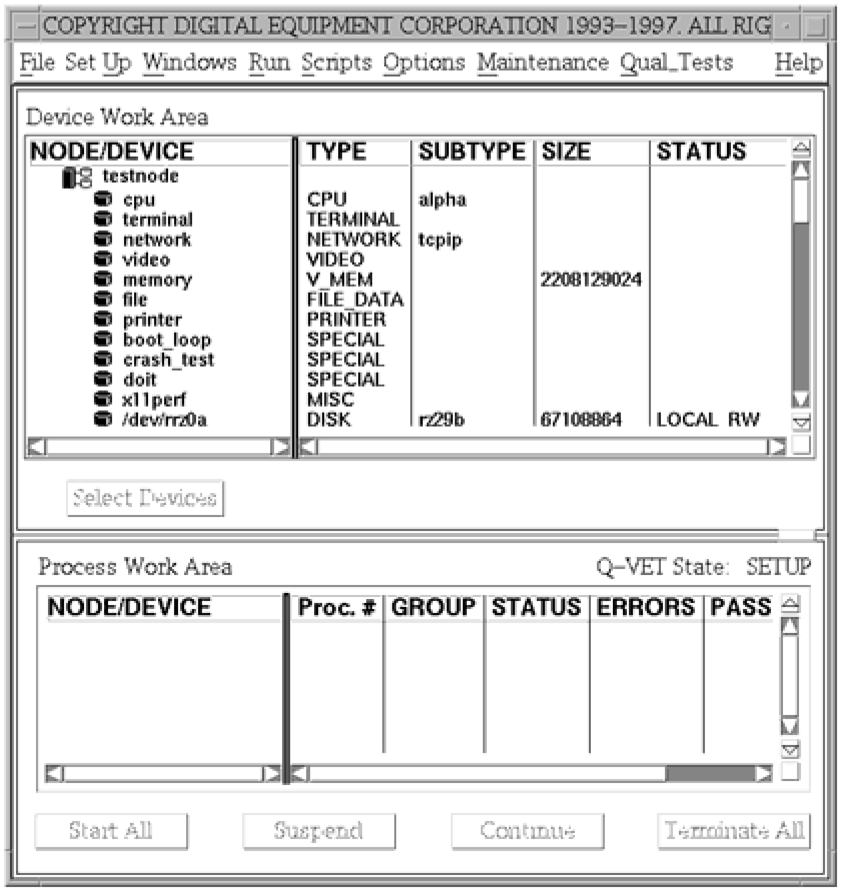
Running system sizer on node testnode...please wait.

This box will disappear once the system sizing process is completed.[3]

Q-VET’s main window is displayed.

[3]The system sizing process finds the devices that are connected to a system and obtains device information needed by Q-VET programs.

Understanding the Main Window


COPYRIGHT DIGITAL EQUIPMENT CORPORATION 1993–1997. ALL RIG
File Set Up Windows Run Scripts Options Maintenance Qual_Tests Help
Device Work Area
Select Devices
Process Work Area
Q–VET State: SETUP
Start All
Suspend
Continue
Terminate All

NODE/DEVICE	TYPE	SUBTYPE	SIZE	STATUS
testnode				
cpu	CPU	alpha		
terminal	TERMINAL			
network	NETWORK	tcpip		
video	VIDEO			
memory	V_MEM		2208129024	
file	FILE_DATA			
printer	PRINTER			
boot_loop	SPECIAL			
crash_test	SPECIAL			
doit	SPECIAL			
x11perf	MISC			
/dev/rrz0a	DISK	rz29b	67108864	LOCAL RW

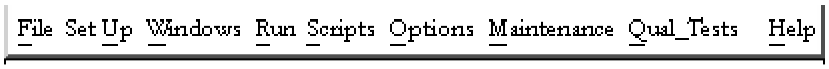
NODE/DEVICE	Proc. #	GROUP	STATUS	ERRORS	PASS

Figure 2-1 Q-VET Main Window - DIGITAL UNIX Systems

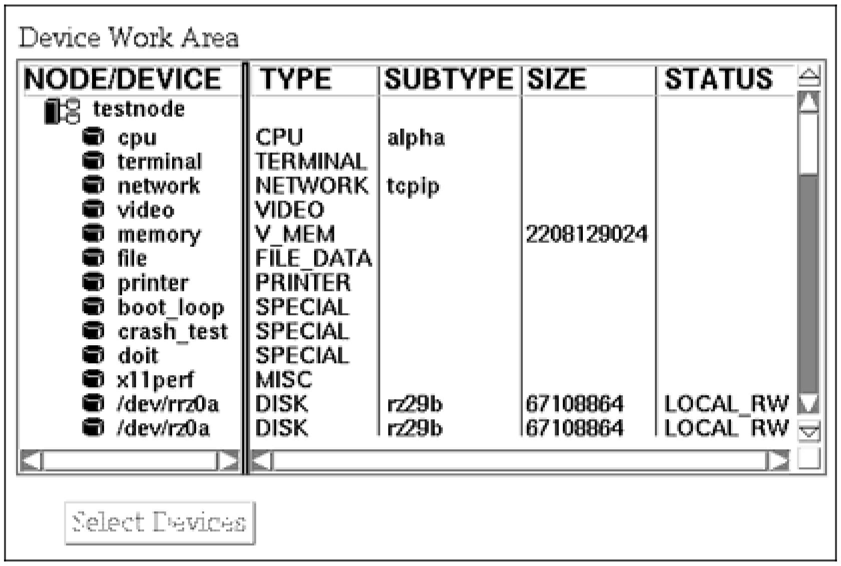
The Q-VET Main Window (Figure 2-1) contains the following areas:

- Title Bar
  The title bar displays the facility name (**Q-VET**) and the window management buttons.

- Menu Bar
  The menu bar lists the menus that you select to perform Q-VET functions.



- Device Work Area Area
  The Device Work Area displays the nodes and devices whose names are contained in Q-VET's configuration database, that is, its database of testable devices.

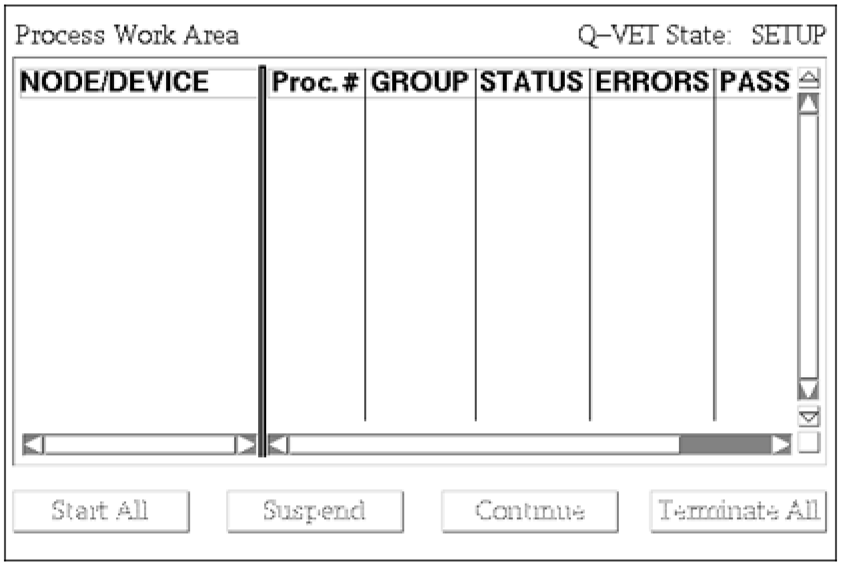

Device Work Area

NODE/DEVICE	TYPE	SUBTYPE	SIZE	STATUS
testnode				
cpu	CPU	alpha		
terminal	TERMINAL			
network	NETWORK	tcpip		
video	VIDEO			
memory	V_MEM		2208129024	
file	FILE_DATA			
printer	PRINTER			
boot_loop	SPECIAL			
crash_test	SPECIAL			
doit	SPECIAL			
x11perf	MISC			
/dev/rrz0a	DISK	rz29b	67108864	LOCAL_RW
/dev/rz0a	DISK	rz29b	67108864	LOCAL_RW

Use the scroll bars to display text that exceeds the size of the display window. The Device Work Area contains the Select Devices button. You click on the Select Devices button to select devices for testing, after highlighting them.

- Pane Separator
  The Pane Separator lets you change the size of the work areas relative to one another by clicking on the rectangle at its right and dragging it up or down.

- Process Work Area
  The Process Work Area displays the nodes and devices you have selected for testing, along with their corresponding process numbers, test groups, status, Errors, Pass #, and Test #."Errors" indicate error count during the run. "Test #" identifies the current test # being performed. "Pass #" identifies the current pass



Use the scroll bars to display text that exceeds the size of the display window.

When you first start a Q-VET session, the Process Work Area is empty.

The Process Work Area provides four buttons, Start All, Suspend, Continue, and Terminate All. Click on these buttons to start, temporarily suspend, continue, and terminate running test processes.

Learning to Use Q-VET

The following sections outline how to perform basic operations in Q-VET

Selecting Devices for a Q-VET Run

The Q-VET run is defined by the nodes or devices placed or "listed" in the Process Area of the Q-VET Main Window. To set up processes for nodes or devices in the Process Area, you must "Select" the nodes or devices. Nodes or devices may be selected in two ways:

1. Loading a predefined "Script" to create processes to test all devices on the system.
2. Choosing them selectively from the Device Area

Loading Predefined Scripts

Using the Q-VET Load option is the simplest way to test your system. The Load option automatically creates processes for most devices on your system.

To use this feature, pull down the Scripts option on the Q-VET Menu Bar, choose Load, and then select any or all of the following scripts.

Script	**Includes:**
CPU Select	All tests for all CPUs on your system
Disk Select	All tests for disks on your system.
Tape Select	All tests for tape drives on your system.
Video Select	All tests for all video options on your system.
Phase 1 Engineering	All tests for all devices on your system.

Processes for the devices you selected appear in the Process Work Area.

Choosing Nodes or Devices From the Device Area.

Q-VET sessions can be customized to allow you to specify which tests you would like to run. Each test can be customized using various options, provided by the Q-VET exercisers. These examples will only show how to change some of the options. To find out about other options, please refer to the online Help.

To select specific nodes or devices listed in the Device Area, you must first highlight them in one of the three ways listed.

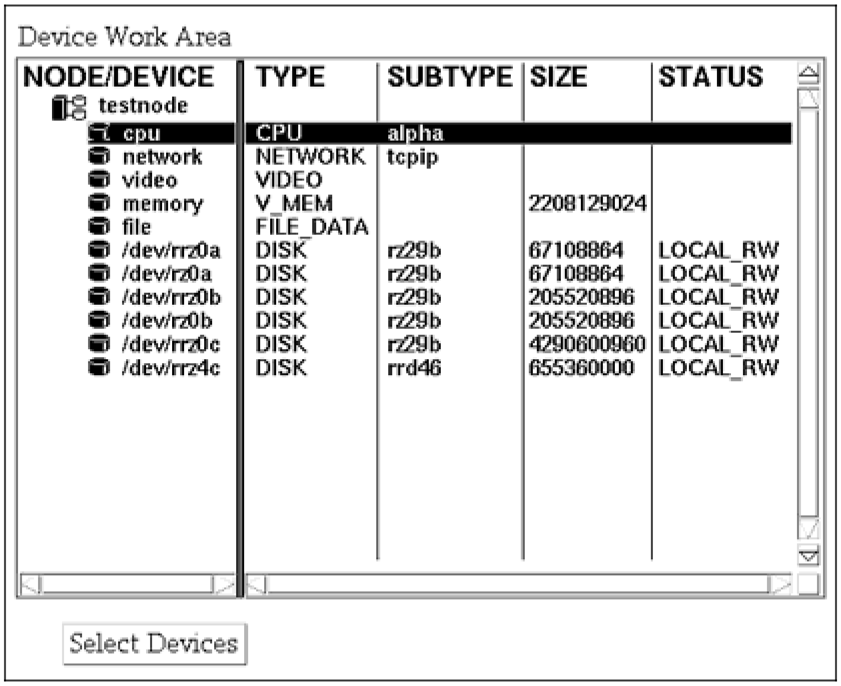
- For a single device or node, point to it and press Mouse Button 1 (MB1), the left hand button on your mouse.
- For multiple contiguous devices, point to the first device, hold down MB1, and drag the pointer down the list of devices; release MB1 while pointing to the last device.

- For multiple noncontiguous devices, highlight the first device or series of devices by pointing or pointing and dragging. Highlight the next device or devices by pointing at them and pressing [Ctrl/MB1]. You can use this technique to highlight as many noncontiguous devices as you want.

After highlighting the devices or nodes, you must select them to be listed in the Process Area by either clicking the “Select Devices” button at the bottom of the Device Area, or placing the cursor on the highlighted item or items, clicking MB3, and choosing “Select Devices”.

To select devices from the Device Work Area for testing:

1. Highlight the names of the nodes or devices that you want to select.
   For this example, the cpu device has been highlighted.



Device Work Area

NODE/DEVICE	TYPE	SUBTYPE	SIZE	STATUS
testnode				
cpu	CPU	alpha		
network	NETWORK	tcpip		
video	VIDEO			
memory	V_MEM		2208129024	
file	FILE_DATA			
/dev/rrz0a	DISK	rz29b	67108864	LOCAL_RW
/dev/rz0a	DISK	rz29b	67108864	LOCAL_RW
/dev/rrz0b	DISK	rz29b	205520896	LOCAL_RW
/dev/rz0b	DISK	rz29b	205520896	LOCAL_RW
/dev/rrz0c	DISK	rz29b	4290600960	LOCAL_RW
/dev/rrz4c	DISK	rrd46	655360000	LOCAL_RW

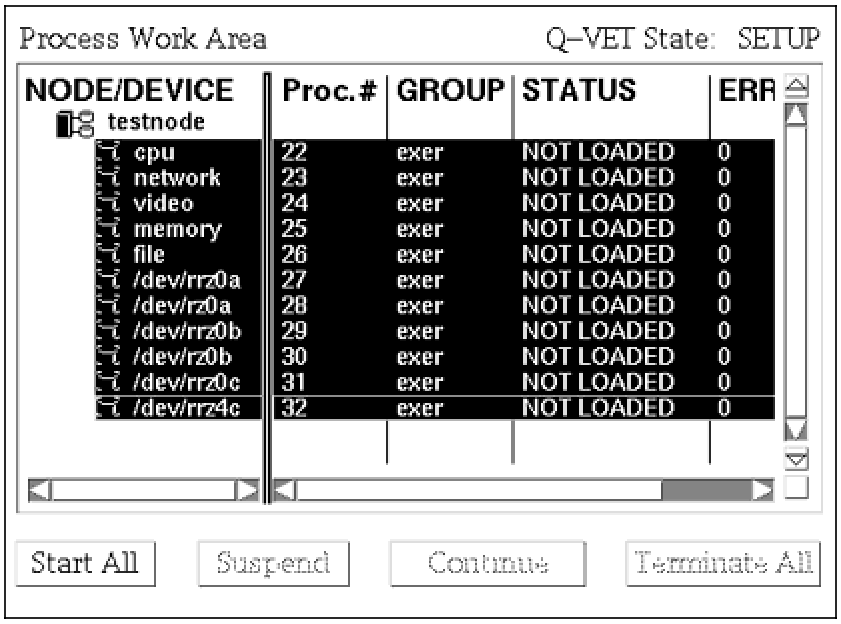
2. Click on the Select Devices button below the Device Work Area or click on Setup on the Menu Bar and choose the Select Devices. The process definitions are displayed in the Process Work Area.
   You can also keep the pointer in the Device Work Area and press MB3. A popup menu displays valid operations. Point to the Select Devices entry and release MB3.

The devices you selected appear in the Process Work Area.

Deselecting Processes

To deselect processes and remove them from a Q-VET run:

1. In the Process Work Area, highlight the names of the nodes or devices that you want to remove by clicking or clicking and dragging.
2. Choose the Set Up menu's Deselect entry.
3. The highlighted entries disappear from the Process Work Area.
4. You can also keep the pointer in the Process Work Area and press MB3.
   A popup menu displays valid operations. Point to the Deselect entry and release MB3.


Process Work Area   Q–VET State: SETUP

NODE/DEVICE	Proc.#	GROUP	STATUS	ERR
testnode				
cpu	22	exer	NOT LOADED	0
network	23	exer	NOT LOADED	0
video	24	exer	NOT LOADED	0
memory	25	exer	NOT LOADED	0
file	26	exer	NOT LOADED	0
/dev/rrz0a	27	exer	NOT LOADED	0
/dev/rz0a	28	exer	NOT LOADED	0
/dev/rrz0b	29	exer	NOT LOADED	0
/dev/rz0b	30	exer	NOT LOADED	0
/dev/rrz0c	31	exer	NOT LOADED	0
/dev/rrz4c	32	exer	NOT LOADED	0

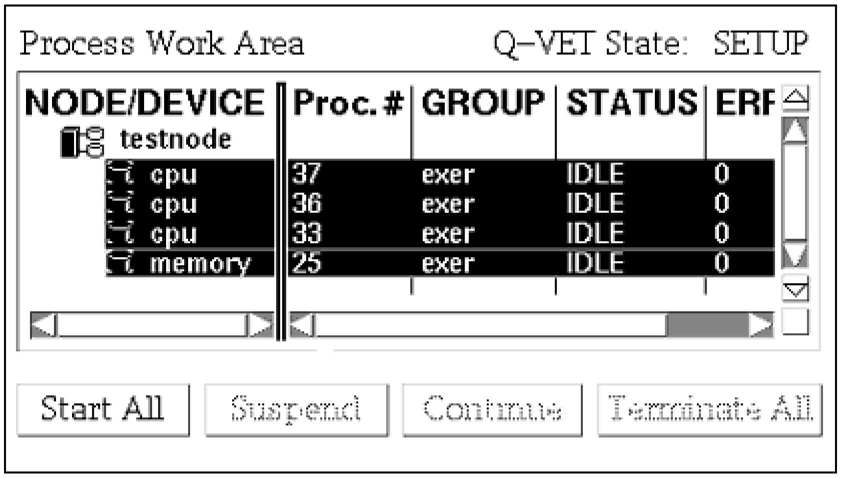
Adding and Dropping Processes

You can add processes to a Q-VET run or drop processes from it. This allows you to increase or decrease the system load gradually.

To add or drop processes:

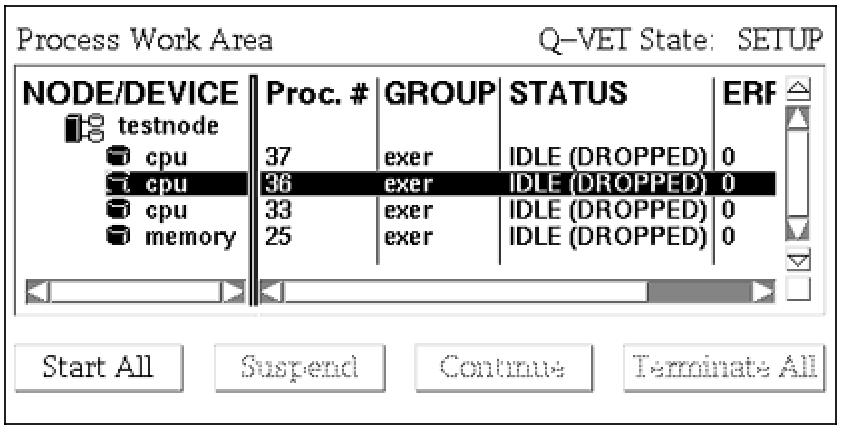
1. In the Process Work Area, highlight the processes that you want to add to a run or drop from it.
2. Click on the Set Up menu.
3. Click on either Add or Drop.
4. You can also keep the pointer in the Process Work Area and press MB3.
   A popup menu displays valid operations. Point to the Add or Drop entry and release MB3. Examples of a Drop and an Add are shown below.

   Highlighted for Drop:



NODE/DEVICE	Proc. #	GROUP	STATUS	ERF
testnode				
cpu	37	exer	IDLE	0
cpu	36	exer	IDLE	0
cpu	33	exer	IDLE	0
memory	25	exer	IDLE	0

Previously dropped process highlighted to be added:


Process Work Area   Q–VET State: SETUP

NODE/DEVICE	Proc. #	GROUP	STATUS	ERF
testnode				
cpu	37	exer	IDLE (DROPPED)	0
cpu	36	exer	IDLE (DROPPED)	0
cpu	33	exer	IDLE (DROPPED)	0
memory	25	exer	IDLE (DROPPED)	0

All processes contained in the Process Work Area before you click start a Q-VET run are automatically added. If you include additional processes *during* a Q-VET run, these processes are automatically dropped. You must add them before they begin executing.

Duplicating a Process

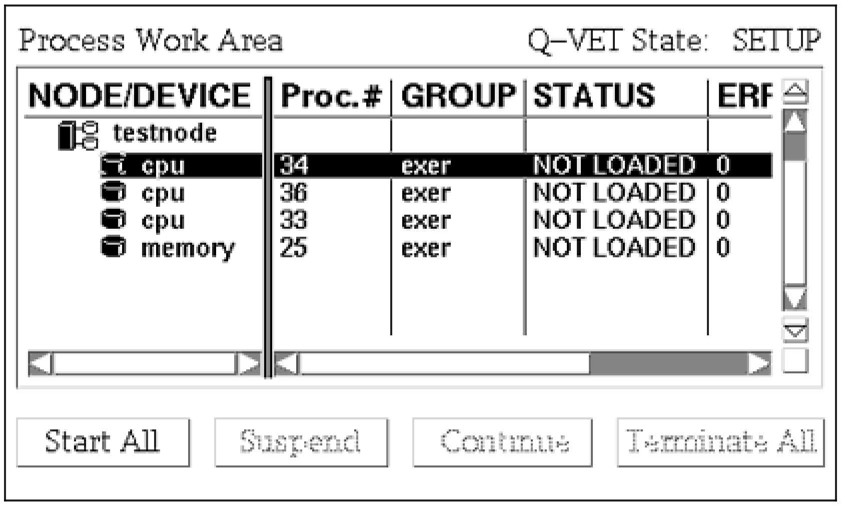
To make one or more copies of an existing process, highlight the processes you wish to duplicate in the Process Work Area, and perform one of the following:

- Choose the Set Up menu's Duplicate entry.
- Keep the pointer in the Process Work Area and press MB3.
  A popup menu displays valid operations. Point to the Duplicate entry and release MB3.

Note

Some processes can only be run one at a time; you will receive a message indicating this at start time.

The highlighted entries are duplicated in the Process Work Area.



NODE/DEVICE	Proc.#	GROUP	STATUS	ERF
testnode				
cpu	34	exer	NOT LOADED	0
cpu	36	exer	NOT LOADED	0
cpu	33	exer	NOT LOADED	0
memory	25	exer	NOT LOADED	0

Repeat the above once for each duplicate you want to make.

Modifying Process Parameters

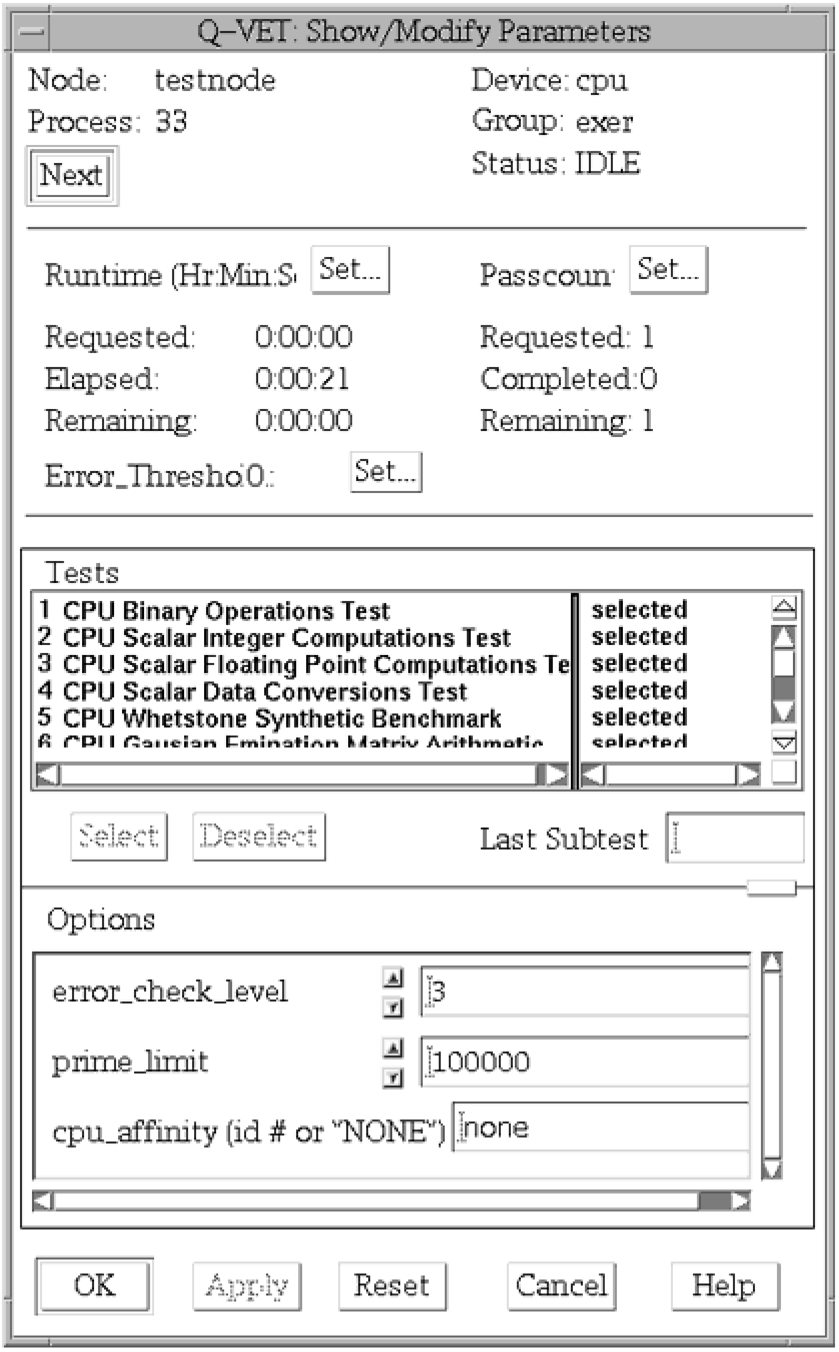
To modify a process definition's parameters:

1. In the Process Work Area, highlight the process.
2. Click on the Set Up menu.
3. Click on the menu's Show/Modify Parameters... entry.
4. You can also keep the pointer in the Process Work Area and press MB3.
   A popup menu displays valid operations . Point to the Show/Modify entry and release MB3.

   The Show/Modify Parameters dialog box is displayed.

Note

The Show/Modify Parameters dialog box and it's dialog windows will differ from the example according to the process you have chosen to modify.



Q–VET: Show/Modify Parameters

Node: testnode
Process: 33

Device: cpu
Group: exer
Status: IDLE

Next

Runtime (Hr:Min:S Set...

Passcoun Set...

Requested:	0:00:00	Requested:	1
Elapsed:	0:00:21	Completed:	0
Remaining:	0:00:00	Remaining:	1

Error_Thresho 0: Set...

Tests

1 CPU Binary Operations Test	selected
2 CPU Scalar Integer Computations Test	selected
3 CPU Scalar Floating Point Computations Te	selected
4 CPU Scalar Data Conversions Test	selected
5 CPU Whetstone Synthetic Benchmark	selected
6 CPU Gausian Emination Matrix Arithmetic	selected

Select Deselect Last Subtest

Options

error_check_level	3
prime_limit	100000
cpu_affinity (id # or "NONE")	none

OK Apply Reset Cancel Help

5. Use the dialog box to view and modify process parameters and options that apply *only to this process*. You can:
   - Set a run time and/or pass count
   - Select or deselect specific tests within the selected test group
   - Modify various process options using click boxes and fill-in dialog windows

If you select several processes in the Process Work Area, you can use the Next button in the Show/Modify Parameters dialog box to examine and change process parameters one process at a time.

1. Click on the Set Up menu.
2. Click on the menu's Show/Modify Parameters... entry; the Show/Modify Parameters screen appears.
3. Read the contents of the displayed dialog box.
4. Click on Next to view the dialog box associated with the next process.
5. Click on Cancel.

Note

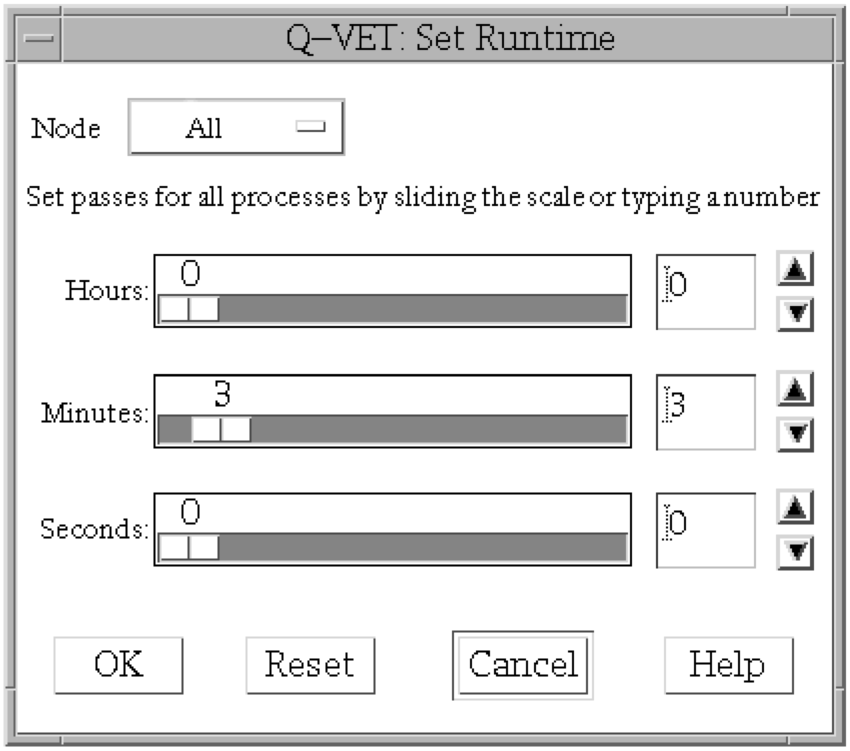
A list of available tests for each device is presented in Appendix D.

Setting the Run Time for All Processes

To set the run time for all processes:[4]

1. In the main window's Option menu choose Set Runtime...
   The Set Runtime dialog box is displayed.
2. Set the run time by sliding the slide bars, clicking on the stepper buttons, or typing numbers in the text fields.
3. Click on the OK button to activate your settings. (Click on the Reset button to restore the current run time.)

[4]A process's run time is the total time for which it is allowed to execute.


Q–VET: Set Runtime

Node: All

Set passes for all processes by sliding the scale or typing a number

Hours: 0

Minutes: 3

Seconds: 0

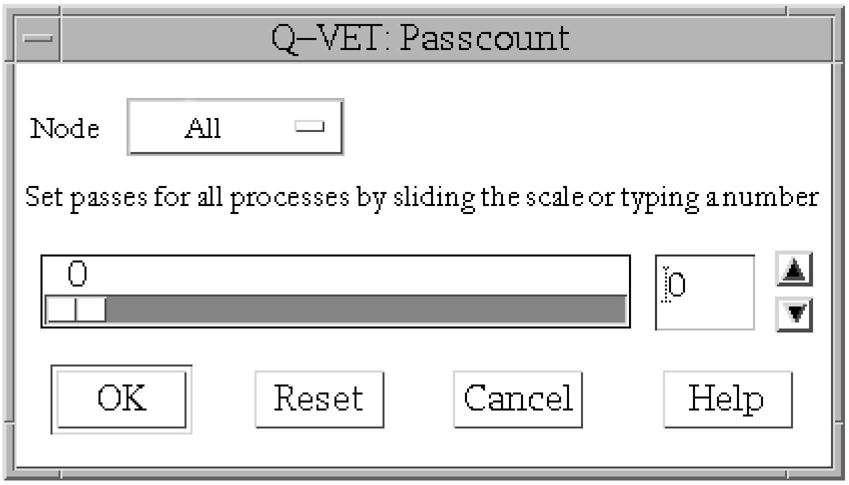
Setting the Pass Count for All Processes

To set the pass count for all processes:

1. In the main window's Options menu, choose Set Passcount.
   The Set Passcount dialog box is displayed.
2. Set the pass count by sliding the slide bar, clicking on the stepper buttons, or typing numbers in the text field. If you set the pass count at 0; Q-VETwill ignore the pass count and use only the run-time value to end the run.
3. Click on the OK button to activate your settings. (Click on the Reset button to restore the current pass count.)

Note

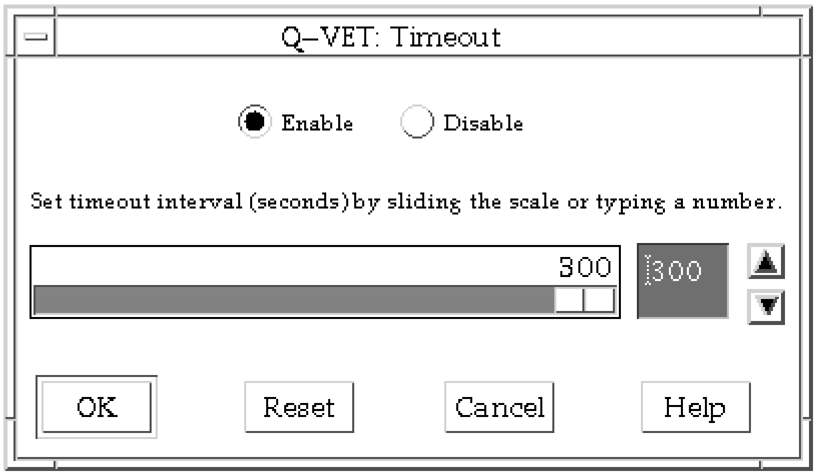
The execution time of one pass is not the same for every exerciser.


Q–VET: Passcount
Node
All
Set passes for all processes by sliding the scale or typing a number
0
0
OK
Reset
Cancel
Help

Setting Timeout for All Processes

To set the timeout interval for all processes:

1. In the main window's Options menu, choose Set Timeout...
   The Timeout dialog box is displayed.


Q–VET: Timeout
Enable
Disable
Set timeout interval (seconds) by sliding the scale or typing a number.
300
300
OK
Reset
Cancel
Help

2. Click on the Enable or Disable button.

3. If you choose to enable timeout, set the timeout interval you wish in seconds by either sliding the horizontal scale and/or typing a new upper value in the right hand box.

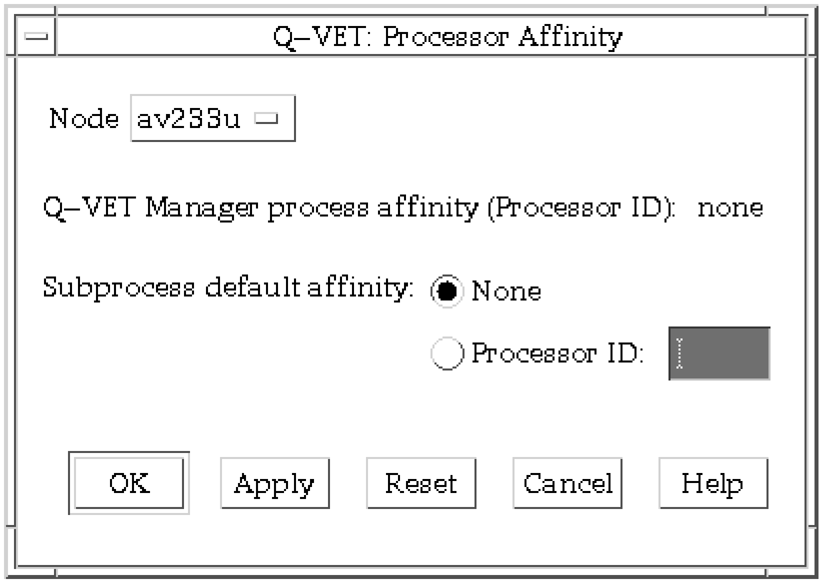
4. Click on the OK button to activate your settings. (Click on the Reset button to restore the current timeout setting.)

Setting the CPU Affinity for All Processes

CPU affinity refers to which CPU the process is bound to (set to run on).

To set the cpu affinity for all processes:

1. In the main window's Options menu, choose Set Affinity...
   The Set Affinity dialog box is displayed.



Q–VET Manager process affinity (Processor ID): none

Subprocess default affinity: [✓ None] Processor ID:

2. Choose the node if there are more than one.

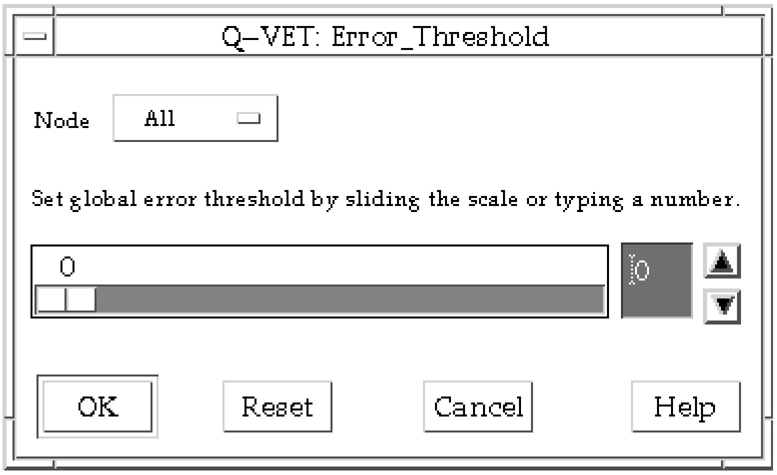
3. Set the subprocess default affinity processor id by clicking on the empty box, then entering the appropriate id number.

4. Click on the OK button to activate your settings. (Click on the Reset button to restore the current affinity id.)

Setting the Error Threshold for All Processes

To set the error threshold for all processes:

1. In the main window's Options menu, choose Set Error_Threshold...
   The Error Threshold dialog box is displayed.



Set global error threshold by sliding the scale or typing a number.

0

2. Click on the appropriate Node.
3. Set the error threshold you wish by either sliding the horizontal scale and/or typing a new upper value in the right hand box.
4. Click on the OK button to activate your settings. (Click on the Reset button to restore the current error threshold setting.)

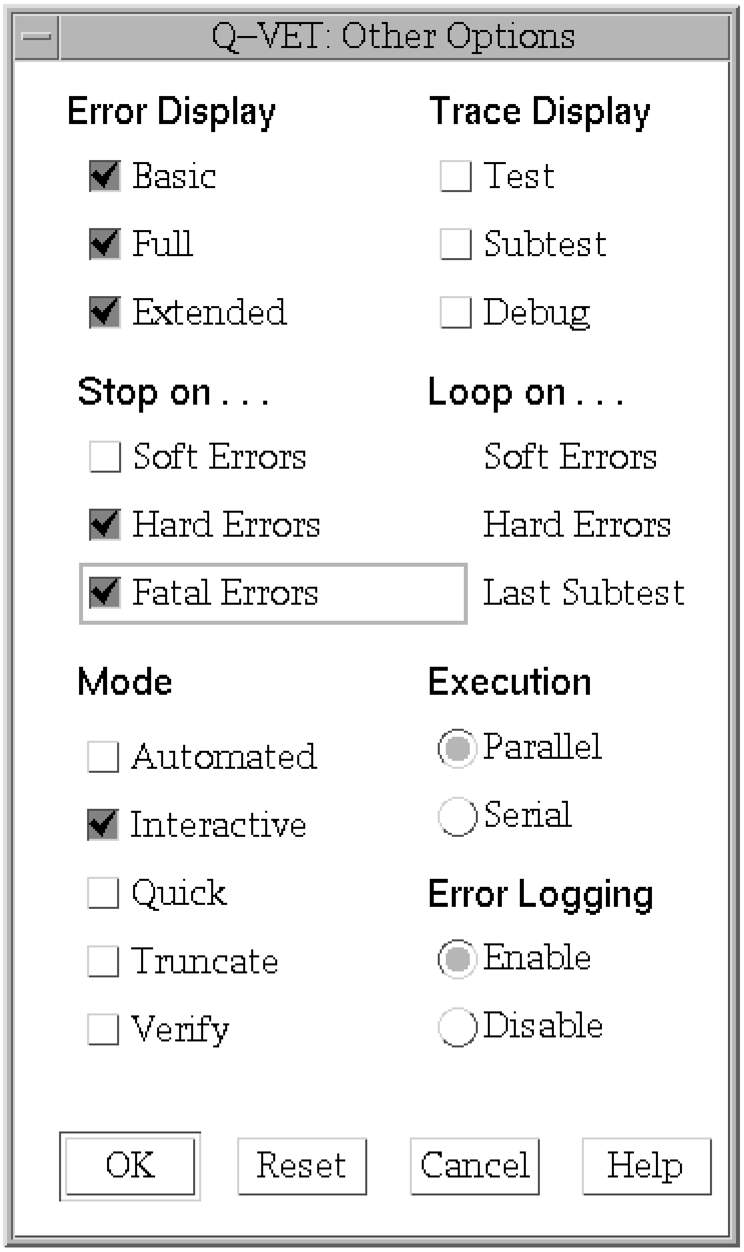
Modifying Other Options

If you choose the Options menu's Other... entry, a dialog box is displayed that lets you modify several Q-VET run-time options. Click on the dialog box's Help button to read about the options.

To modify Other options:

1. Click on the Options menu
2. Click on the Other... entry

The Q-VET Other Options Dialogue Window is displayed.


Q–VET: Other Options

Error Display
[✓ Basic]
[✓ Full]
[✓ Extended]

Trace Display
Test
Subtest
Debug

Stop on . . .
Soft Errors
[✓ Hard Errors]
[✓ Fatal Errors]

Loop on . . .
Soft Errors
Hard Errors
Last Subtest

Mode
Automated
[✓ Interactive]
Quick
Truncate
Verify

Execution
[✓ Parallel]
Serial

Error Logging
[✓ Enable]
Disable

To set Other options, do the following.

1. Click on the desired option button(s).

2. Click on the OK button.

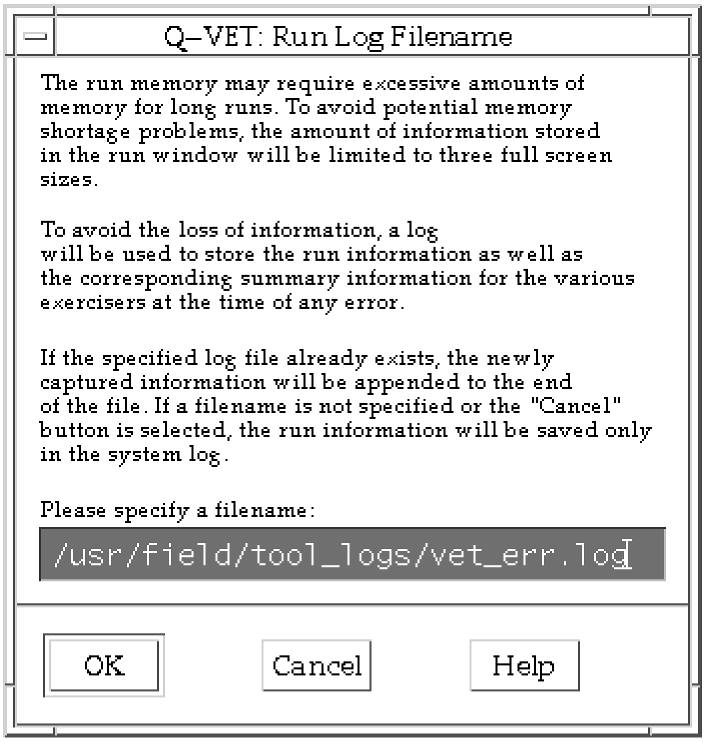
For this example, we enabled stopping on hard and fatal errors.

Starting a Q-VET Run

To start a Q-VET run, click on the Start All button in the Process Work Area. This starts the run that was set up by the Select or Load options.,

If this is your first run in this Q-VET session,, a Log File Screen will appear. This screen shows a default location where a log of all Q-VET activity will be posted. If you wish to change the default location, you may place your cursor in the edit box and type a new location.

Click OK.


Q–VET: Run Log Filename

The run memory may require excessive amounts of memory for long runs. To avoid potential memory shortage problems, the amount of information stored in the run window will be limited to three full screen sizes.

To avoid the loss of information, a log will be used to store the run information as well as the corresponding summary information for the various exercisers at the time of any error.

If the specified log file already exists, the newly captured information will be appended to the end of the file. If a filename is not specified or the "Cancel" button is selected, the run information will be saved only in the system log.

Please specify a filename:

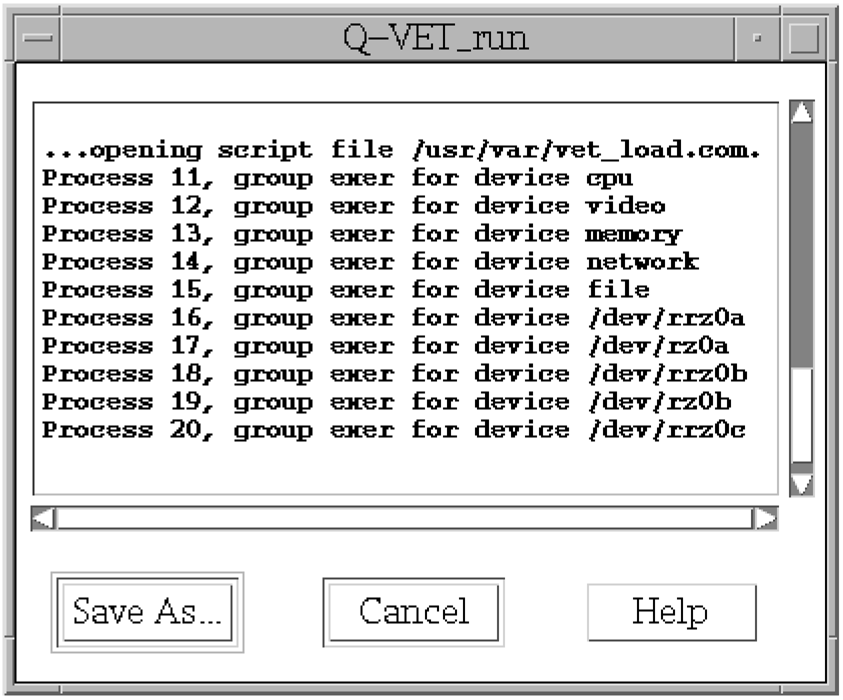
/usr/field/tool_logs/vet_err.log

Note

If you have previously started a test run during this session, Q-VET skips the Log File Screen. (If you wish to change the log file location, you may exit Q-VET and start a new session to obtain the Log File Screen).

If a run was started using the Qual Menu (see Running Q-VET Qual Tests), the run log filename will automatically be chosen as `/usr/field/tool_logs/vet_err_nnnnnn-mm-dd-yyyy.log` where nnnnnn is the nodename of the system under test, and mm-dd-yyyy is the date when the test was started.

The Q-VET_run window is displayed. It shows the events that occur during the run---process start, error reports, and process completion.



```
...opening script file /usr/var/vet_load.com.
Process 11, group exer for device cpu
Process 12, group exer for device video
Process 13, group exer for device memory
Process 14, group exer for device network
Process 15, group exer for device file
Process 16, group exer for device /dev/rrz0a
Process 17, group exer for device /dev/rz0a
Process 18, group exer for device /dev/rrz0b
Process 19, group exer for device /dev/rz0b
Process 20, group exer for device /dev/rrz0c
```

Wait for the run to complete, or suspend and terminate the test when you desire.

Viewing Test Results

Test results can appear in many places. Error messages are displayed in the Run Display window. The Summary window contains brief error messages and run statistics. The System Error Log contains errors that were detected by the system during the run.

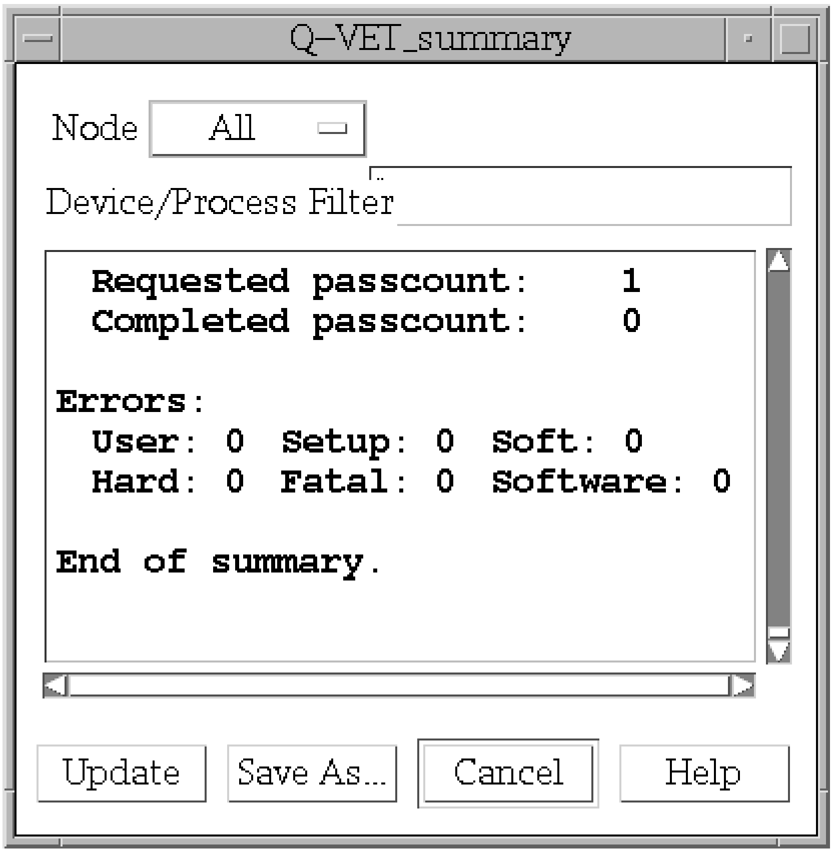
Note

When the session is terminated, a summary file containing test summary information and any system errors which may have occured during the test run is generated. This summary file can be found in the \dec\vet\tool_logs directory in DIGITAL UNIX, and its equivalent in Open VMS, sys$specific:[sysmgr.tool_logs].
This file will always be named `vet_sum_nnnnnn_mm-dd-yyyy.log`.

Viewing a Run Summary

To view a run summary during or after a run:

1. Choose the Summary... entry from the Windows menu.
   A summary window is displayed. It shows the state of the Q-VET run at the time you invoked the window.



```
  Requested passcount:    1
  Completed passcount:    0

Errors:
  User: 0  Setup: 0  Soft: 0
  Hard: 0  Fatal: 0  Software: 0

End of summary.
```

2. During a run you can:

   b) Click on the Update button to update the display.

   c) Enter specific devices or processes in the Summary window's Device/Process Filter field. This causes Q-VET to display summaries for only these devices or processes the next time Update is selected.

   d) Click on the Save As... button to save the summary in a file.

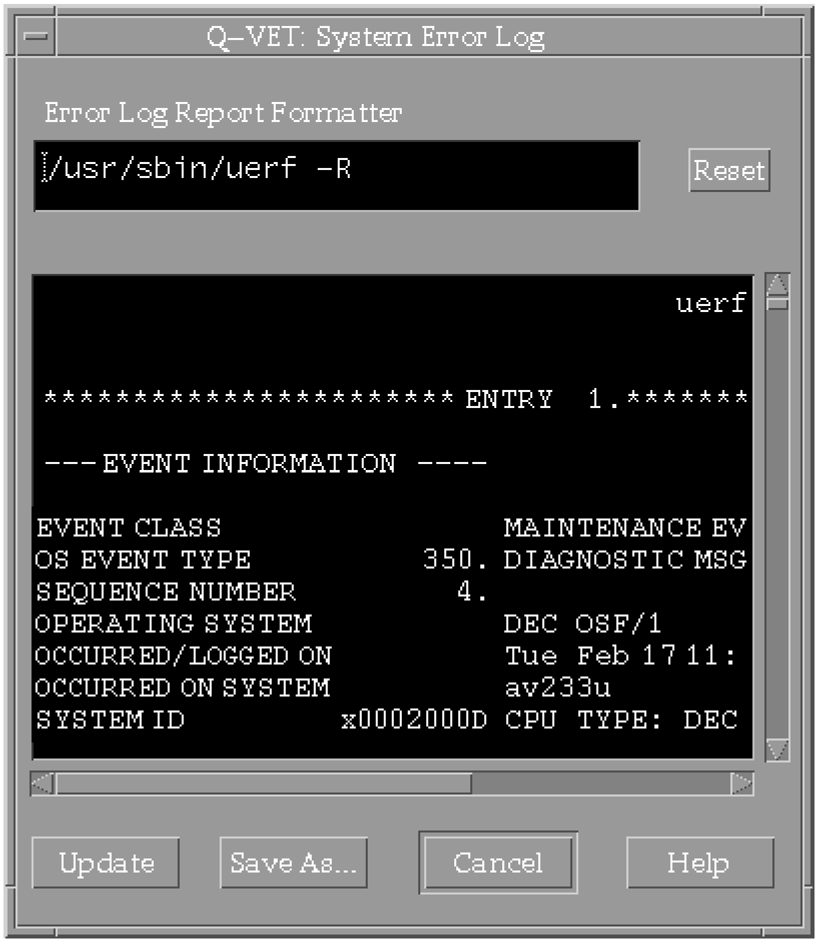
Note

When the session is terminated, a summary file containing test summary information and any system errors which may have occured during the test run is generated. This summary file can be found in the usr/field/tool_logs directory.

Viewing the System Error Log

To view the System Error Log:

1. Click on Window in the Main Menu and choose System Error Log.



```
/usr/sbin/uerf -R
```

```
                                         uerf


 ************************ ENTRY   1.*******

 --- EVENT INFORMATION ----

EVENT CLASS                        MAINTENANCE EV
OS EVENT TYPE             350.  DIAGNOSTIC MSG
SEQUENCE NUMBER             4.
OPERATING SYSTEM                   DEC OSF/1
OCCURRED/LOGGED ON                 Tue Feb 17 11:
OCCURRED ON SYSTEM                 av233u
SYSTEM ID            x0002000D  CPU TYPE: DEC
```

2. The default (or most recent) report format is shown in the Error Log Report Formatter box. You can reformat the report by entering new parameters and clicking Reset if you wish.
3. Scroll down the display window to view the errrors in the format you have chosen.
4. When you are done, exit the window by clicking File and choosing Exit.

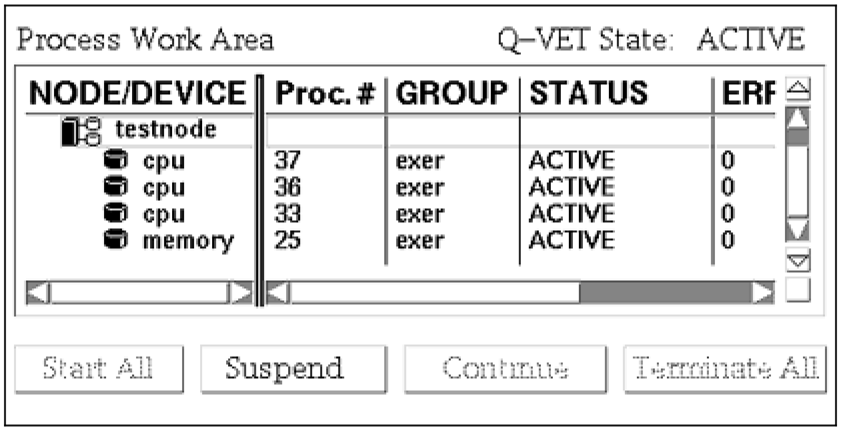
Suspending, Continuing, and Terminating a Q-VET Run or Process

You can suspend, continue, or terminate an entire Q-VET run, or individual processes.

The Run Menu is divided into upper and lower sections. The upper section lists Start All, Suspend, Continue All, and Terminate All; these choices affect the entire Q-VET run, i.e. all processes whether highlighted or not. The lower section of the Run Menu lists Suspend, Continue and Terminate; these choices affect individual processes which have been highlighted.

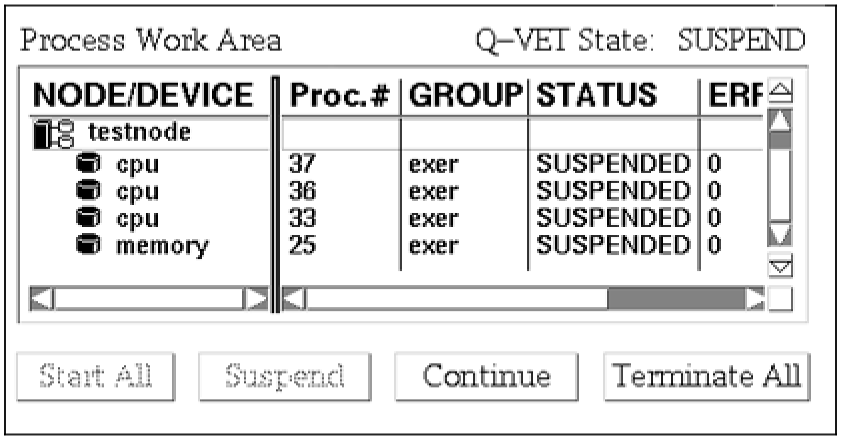
Suspending, Continuing, and Terminating a Q-VET Run

Use the buttons on the bottom of the Process Work Area to suspend a Q-VET run, continue a suspended run, or terminate a run.



NODE/DEVICE	Proc. #	GROUP	STATUS	ERF
testnode				
cpu	37	exer	ACTIVE	0
cpu	36	exer	ACTIVE	0
cpu	33	exer	ACTIVE	0
memory	25	exer	ACTIVE	0

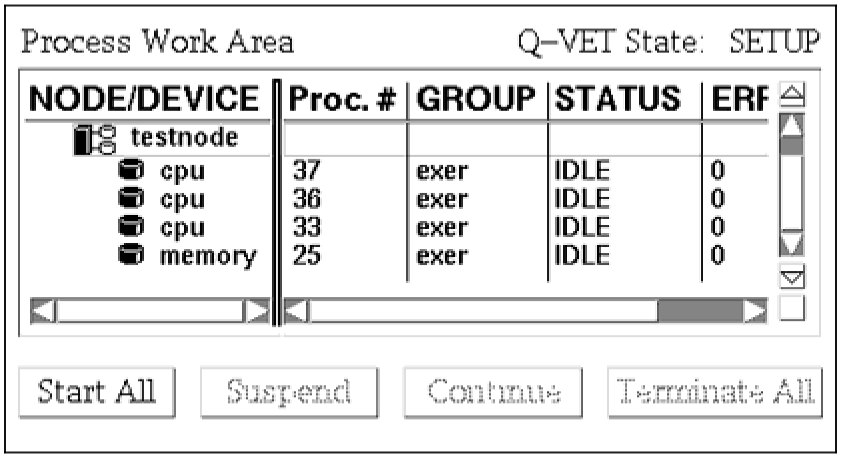
Clicking the Suspend button suspends all Q-VET internal tests. (Note that it does not suspend external tests started by Q-VET processes.) Each one's status is listed as suspended in the Process Work Area.


Process Work Area Q–VET State: SUSPEND

NODE/DEVICE	Proc.#	GROUP	STATUS	ERF
testnode				
cpu	37	exer	SUSPENDED	0
cpu	36	exer	SUSPENDED	0
cpu	33	exer	SUSPENDED	0
memory	25	exer	SUSPENDED	0

Clicking the Continue button causes all processes resume executing. Each one's status is listed as active in the Process Work Area.

To terminate a Q-VET run while it is running, click on the Suspend button then click on the Terminate All button. This terminates the execution of all processes whether or not they have completed. Each one's status is listed as idle in the Process Work Area.


Process Work Area Q–VET State: SETUP

NODE/DEVICE	Proc.#	GROUP	STATUS	ERF
testnode				
cpu	37	exer	IDLE	0
cpu	36	exer	IDLE	0
cpu	33	exer	IDLE	0
memory	25	exer	IDLE	0

Suspending, Continuing, and Terminating Individual Processes

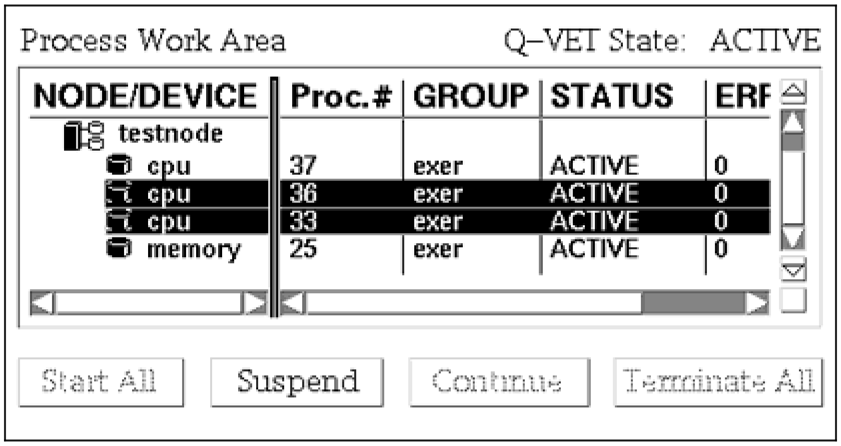
After you have started a run, you can suspend, continue, or terminate individual processes by using Run menu entries.

The Run menu's Start All, Suspend, Continue, and Terminate All entries are identical to the buttons under the Process Work Area. They affect all processes.

To suspend, continue, or terminate individual processes:

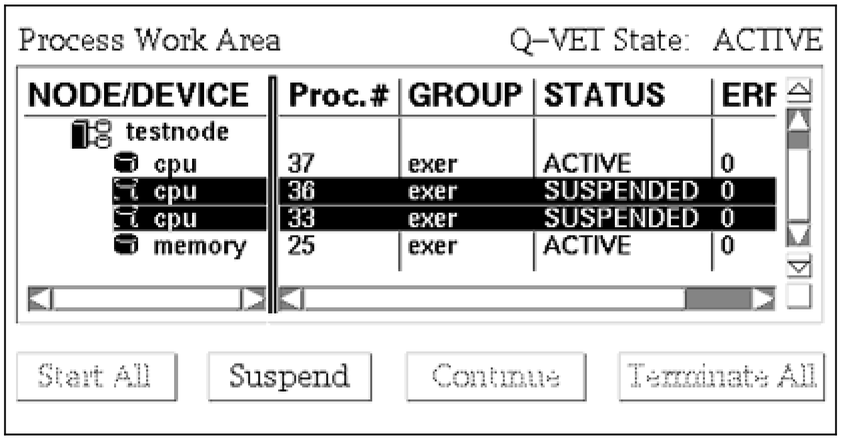
1. In the Process Work Area, highlight the names of the processes that you want to suspend, continue, or terminate.
2. Click on the Run menu.
3. Click on Suspend, Continue, or Terminate.

In this example, two cpu processes have been highlighted.


Process Work Area   Q–VET State: ACTIVE

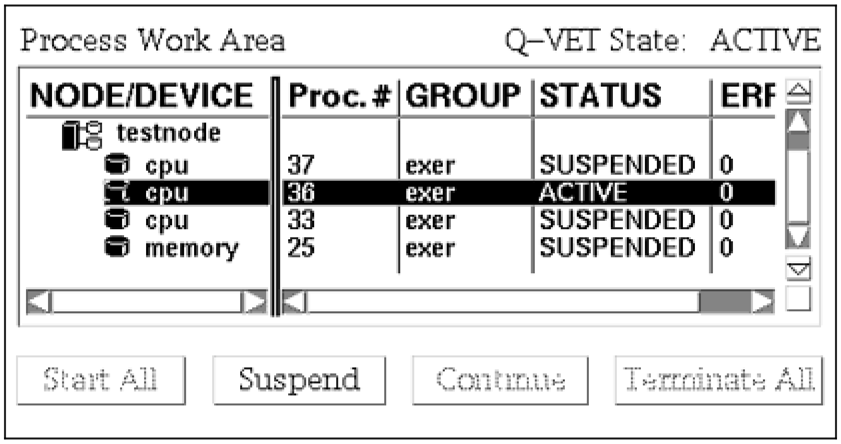
NODE/DEVICE	Proc.#	GROUP	STATUS	ERI
testnode				
cpu	37	exer	ACTIVE	0
cpu	36	exer	ACTIVE	0
cpu	33	exer	ACTIVE	0
memory	25	exer	ACTIVE	0

The Suspend option in the Run menu is chosen to stop the two cpu processes. Q-VET stops the two processes.


Process Work Area   Q–VET State: ACTIVE

NODE/DEVICE	Proc.#	GROUP	STATUS	ERI
testnode				
cpu	37	exer	ACTIVE	0
cpu	36	exer	SUSPENDED	0
cpu	33	exer	SUSPENDED	0
memory	25	exer	ACTIVE	0

In this example, all processes have been suspended; and one cpu process has been highlighted and continued by selecting the Continue option in the Run menu.

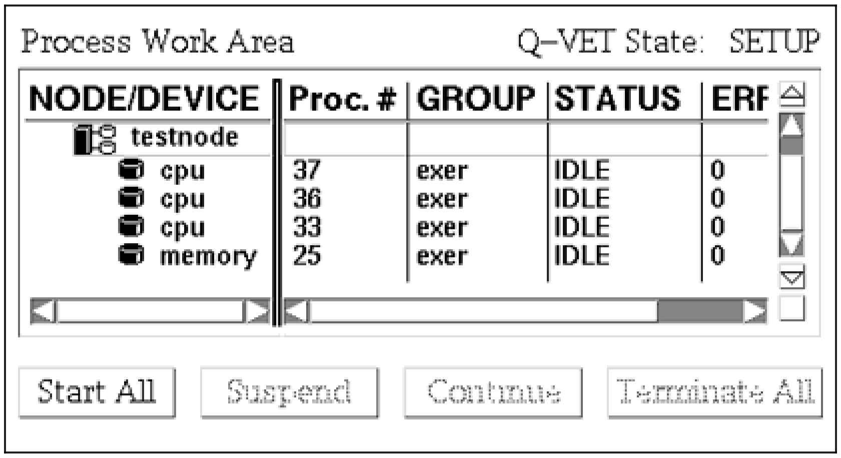

Process Work Area   Q–VET State: ACTIVE

NODE/DEVICE	Proc. #	GROUP	STATUS	ERF
testnode				
cpu	37	exer	SUSPENDED	0
cpu	36	exer	ACTIVE	0
cpu	33	exer	SUSPENDED	0
memory	25	exer	SUSPENDED	0

Note

If tests were started using the Qual Menu, they can only be terminated by clicking the Terminate button in the Qual Status window.

In this example, all processes have been highlighted and terminated by selecting the Terminate All option from the Run Menu.


Process Work Area   Q–VET State: SETUP

NODE/DEVICE	Proc. #	GROUP	STATUS	ERF
testnode				
cpu	37	exer	IDLE	0
cpu	36	exer	IDLE	0
cpu	33	exer	IDLE	0
memory	25	exer	IDLE	0

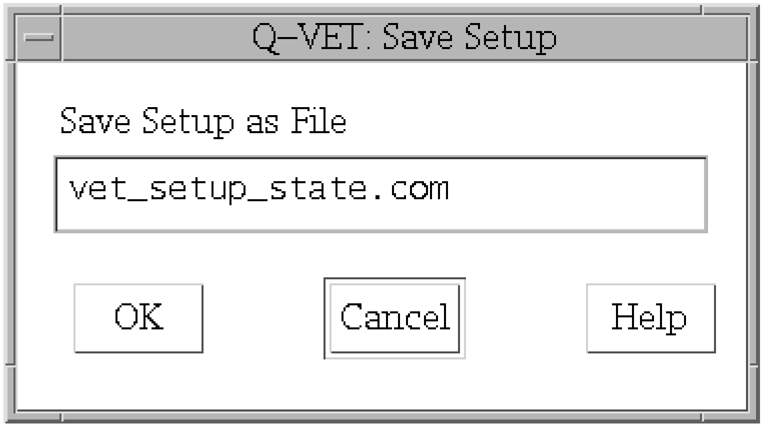
Saving and Restoring Setup

You can save or clear the current configuration displayed in the Device Work and Process Work Areas, current process parameters, and Q-VET options, and you can recall this saved state later during the current Q-VET session or in future ones.

Saving the Setup State

To save the current setup state, do the following:

1. Click on the Set Up menu.
2. Click on the Set Up menu's Save Setup... entry.
   The Save Setup dialog box is displayed.



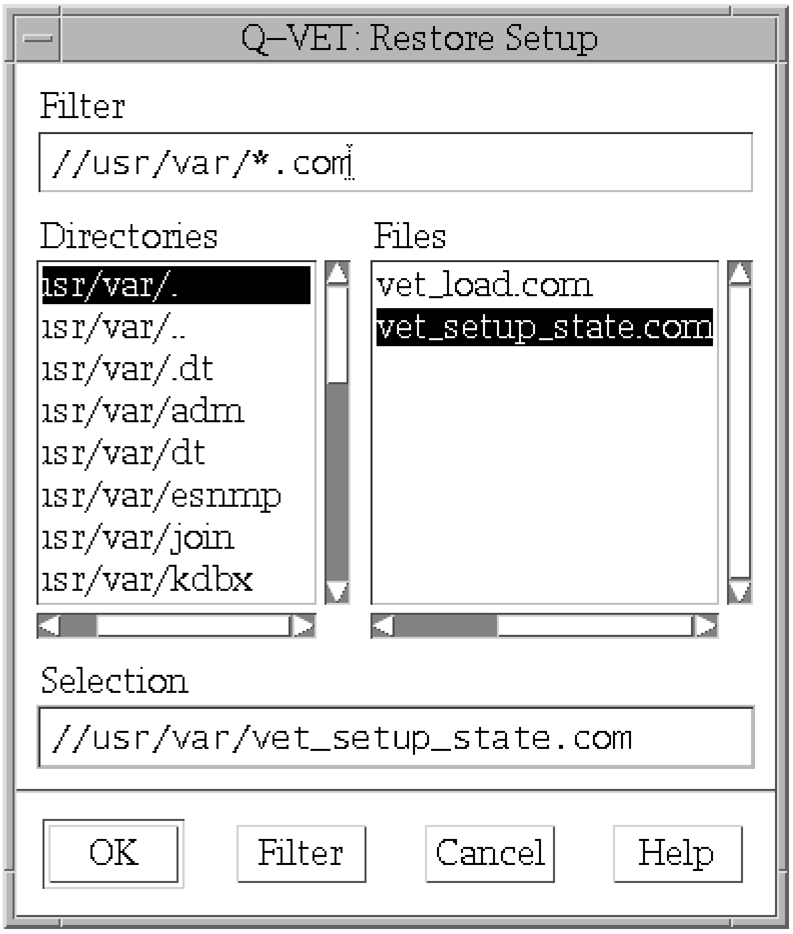
vet_setup_state.com

3. You can save the setup in the default file whose name is provided.
4. Or you can save the setup in a file of another name:
   - Point within the Save Setup as File selection box and click on MB1.
   - Delete the file name that already appears.
   - Enter the name of the file in which you want to save the setup.
   - Click on the OK button to save the state in that file.

Restoring a Setup State

To restore a saved setup state:

1. Click on the Set Up menu.
2. Click on the menu's Restore Setup... entry.
   The Restore Setup dialog box is displayed.



Filter: //usr/var/*.com

Directories:
```
ısr/var/.
ısr/var/..
ısr/var/.dt
ısr/var/adm
ısr/var/dt
ısr/var/esnmp
ısr/var/join
ısr/var/kdbx
```

Files:
```
vet_load.com
vet_setup_state.com
```

Selection: //usr/var/vet_setup_state.com

3. Specify a filename by clicking on a filename in the Files field or by typing a filename in the Selection field.
4. Click on the OK button.
   Q-VET clears the current setup state, restores the state contained in the specified file, and updates the main window.

Editing and Executing Script Files

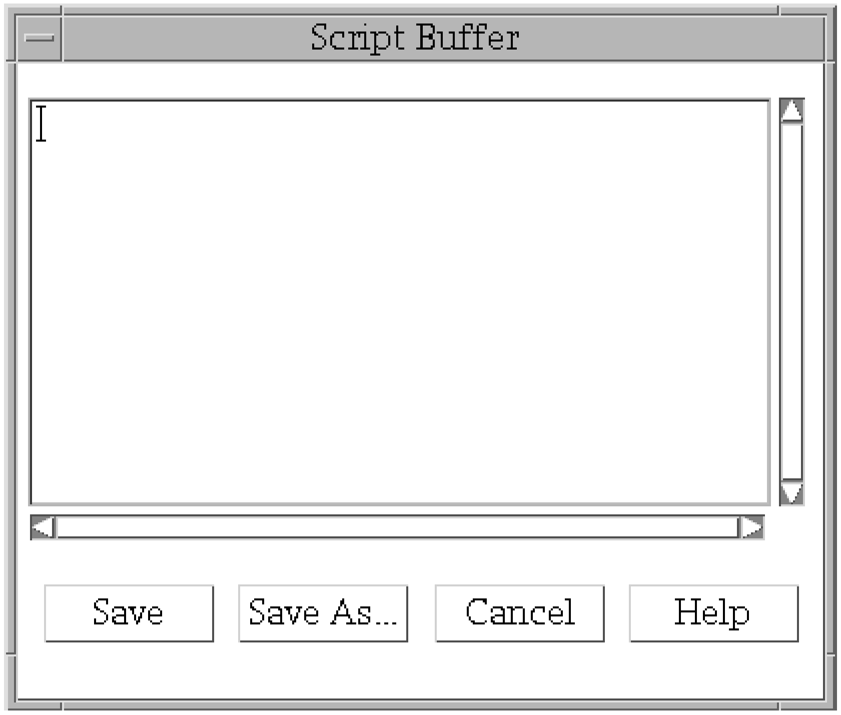
You can edit and execute script files by selecting the Scripts menu. A script or command file is a file containing commands that Q-VET can execute.

Use the script buffer to create and execute temporary scripts. Use script files to save your scripts permanently.

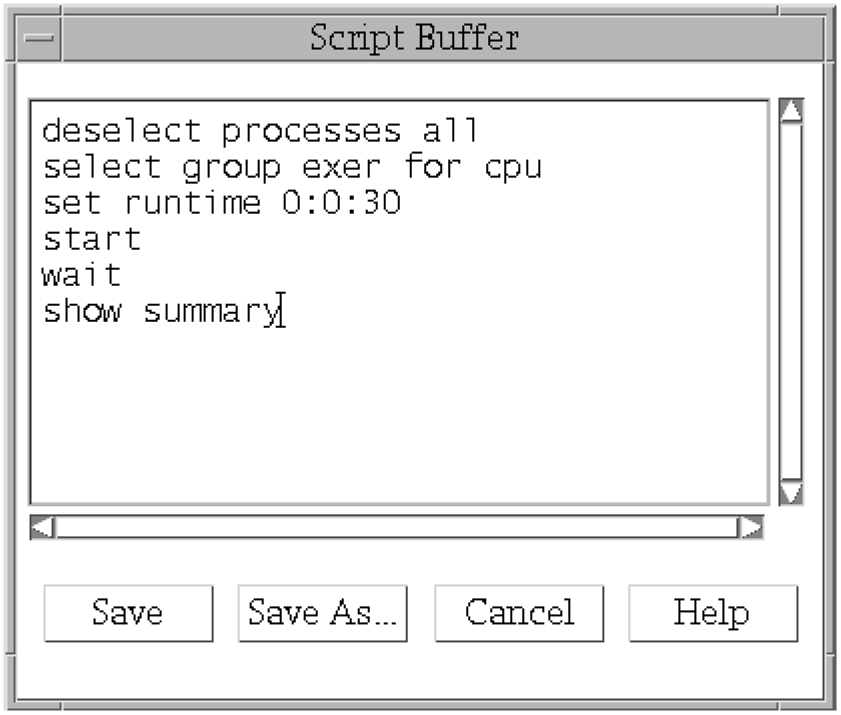
To edit the script buffer:[5]

1. Click on the Scripts menu.
2. Click on the menu's Edit entry.
3. Choose the Script Buffer... entry of the submenu.
   The Script Buffer dialog box is displayed.

[5]The script buffer's contents are not permanent. The first time you issue the **edit** command during a Q-VET session, the script buffer is empty. The commands you enter can be overwritten during the session. The script buffer's contents are lost when you terminate the session.

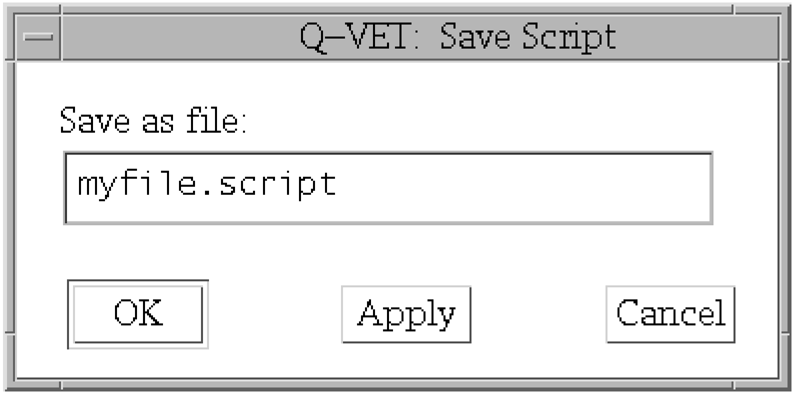


4. Activate the text field by pointing within it and clicking on MB1.
5. Enter one Q-VET command per line in the text field; use lower case letters.



```
deselect processes all
select group exer for cpu
set runtime 0:0:30
start
wait
show summary
```

6. After you have finished entering commands, click on the Save button. (Q-VET does not execute commands, unless you have clicked on Save.)



Save as file: myfile.script

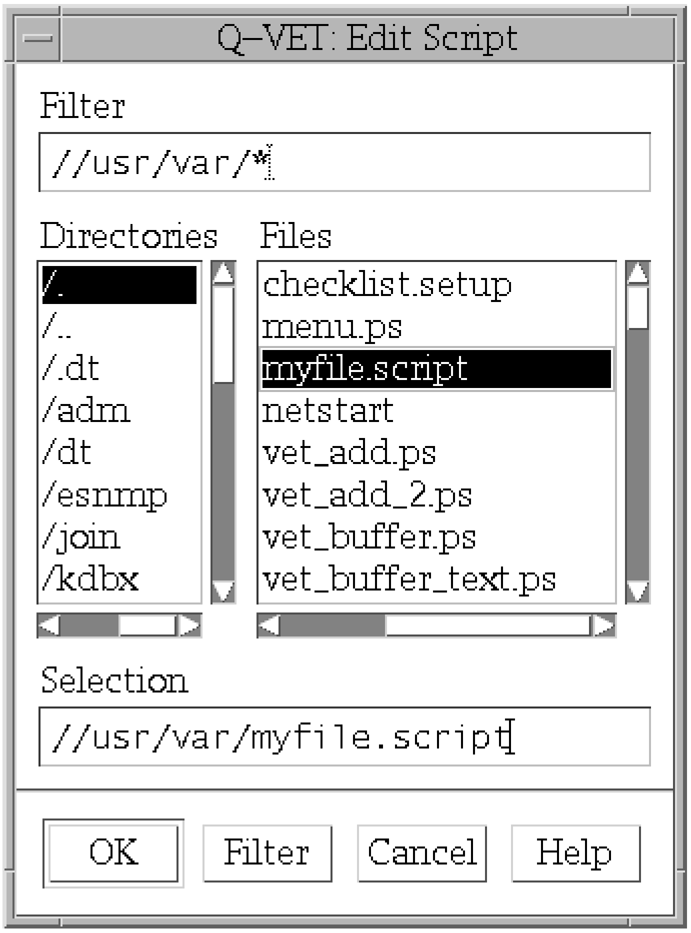
To execute commands in the script buffer:

1. Click on the Scripts menu.
2. Click on the menu's Execute entry.
3. Choose the Script Buffer entry of the submenu. Q-VET will now execute the commands.

To create or edit a permanent script file:

1. Click on the Scripts menu.
2. Choose the menu's Edit entry.
3. Choose the File... entry of the submenu.

The Edit Script dialog box is displayed, so that you can select a file name.



To execute commands in a permanent script file:

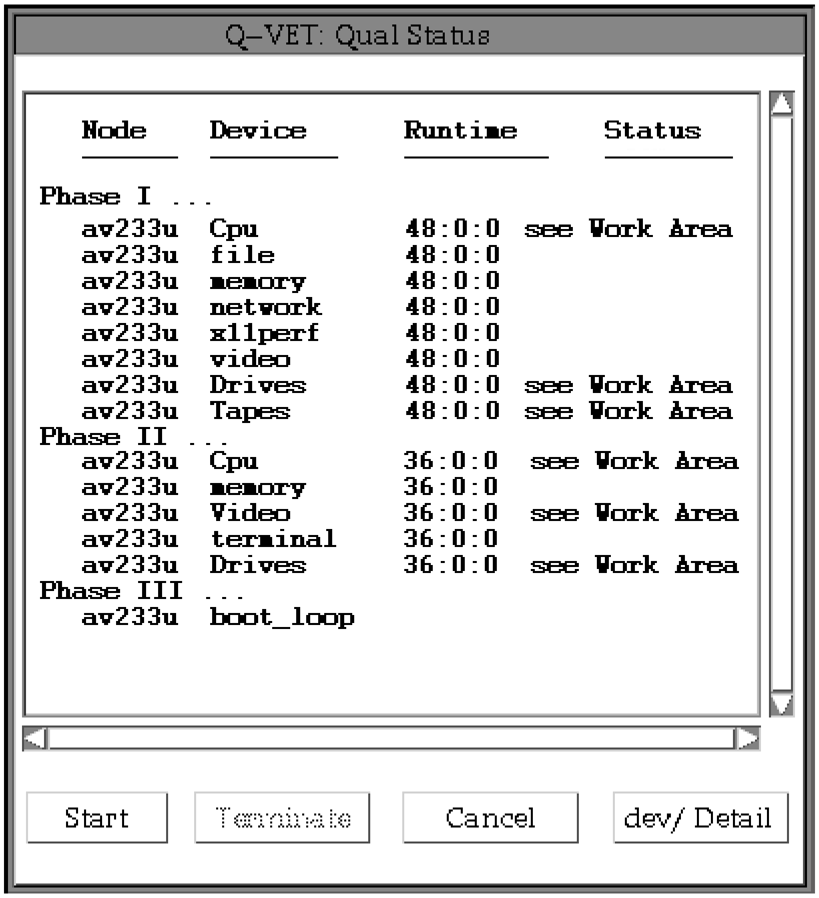
1. Click on the Scripts menu.
2. Choose the menu's Execute entry.
3. Choose the File... entry of the submenu.
   The Execute Script dialog box is displayed, so that you can select a file. (This is essentially the same screen as the Edit Script dialog box.)

Running Q-VET Qual Tests

Q-VET provides several predefined Qual Tests. You can run Q-VET Qual Tests by doing the following:

1. Click on Qual_Tests on the Q-VET Menu Bar.
2. Choose “Cert. Tests” from the pull down menu.
3. Choose “Engineering” from the next pull down menu.

The Q-VET Qual Status screen appears.



```
   Node    Device      Runtime     Status

Phase I ...
   av233u  Cpu         48:0:0  see Work Area
   av233u  file        48:0:0
   av233u  memory      48:0:0
   av233u  network     48:0:0
   av233u  x11perf     48:0:0
   av233u  video       48:0:0
   av233u  Drives      48:0:0  see Work Area
   av233u  Tapes       48:0:0  see Work Area
Phase II ...
   av233u  Cpu         36:0:0  see Work Area
   av233u  memory      36:0:0
   av233u  Video       36:0:0  see Work Area
   av233u  terminal    36:0:0
   av233u  Drives      36:0:0  see Work Area
Phase III ...
   av233u  boot_loop
```

4. Use the Start or Terminate buttons on the bottom of the screen to start or terminate the Qual Tests. The Cancel button lets you exit the screen without taking action, and the dev/Detail button lets you view the qual test details for a particular device.

Terminating a Q-VET Session

To terminate the Q-VET session:

1. Be sure all processes have been terminated.
2. Click on the File menu.
3. Click on the menu's Exit entry.

3

Windows NT Q-VET

This chapter gets you started with using Q-VET on the Windows NT operating system by providing easy-to-follow examples of its use. (If you are using the DIGITAL UNIX or OpenVMS operating systems, see Chapter 2.)

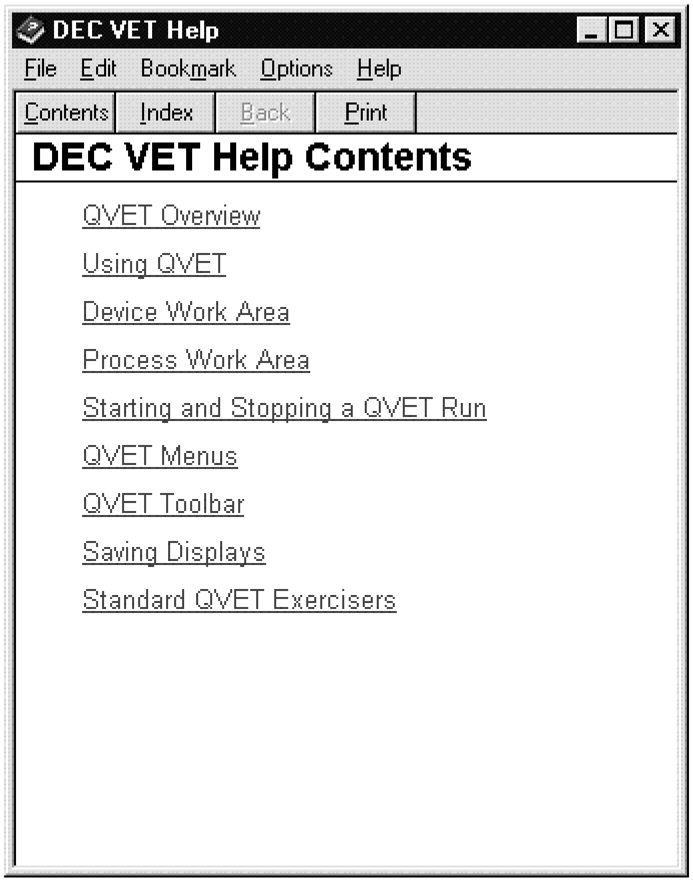
Online Help

Help can be obtained from the Help Menu, or Help buttons in the dialog boxes, which provide a Help display corresponding to that dialog box. Most Help topics have hot spots which allow you to view other topics. The hot spots are underlined and appear in green. When you click on a hot spot, the corresponding help text appears.

The Help Menu provides entries for obtaining the contents for Q-VET help (shown in the following figure), searching for keywords in Q-VET help, and obtaining information on how to use the Help facility.

To obtain context sensitive help on a child window or dialog box, select the window or dialog box and press the F1 key. A help window appears with information on the window or dialog box. In order to get information on the main Q-VET window in this manner, you must close all the child windows.

To obtain context sensitive help on a menu entry, press the Shift and F1 keys at the same time. The cursor changes to an arrow and question mark indicating context sensitive help. To obtain help on an entry, position the cursor over the entry and click MBR. A help window appears displaying information on that menu entry.



DEC VET Help Contents

QVET Overview

Using QVET

Device Work Area

Process Work Area

Starting and Stopping a QVET Run

QVET Menus

QVET Toolbar

Saving Displays

Standard QVET Exercisers

Getting Started With the Windows NT Interface

This section reproduces a Q-VET session using the Windows NT interface.

The section does not describe every available function. To obtain more information on an item, click on the Help button in the menu bar or in the window in which the item appears. A Help window with detailed information is displayed.

Note

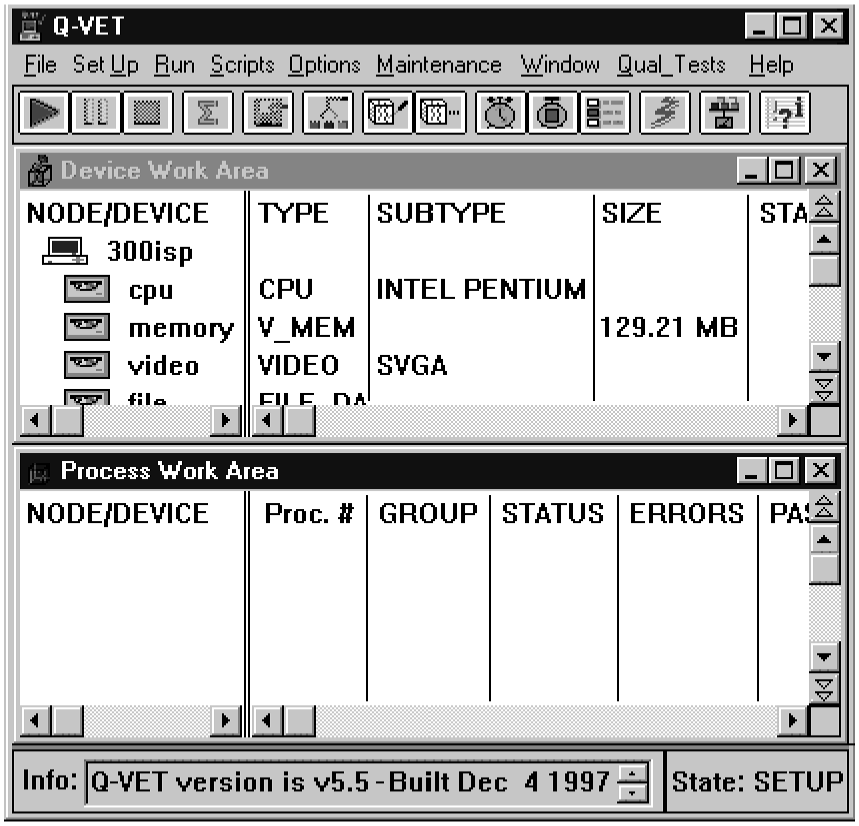
Q-VET for the Windows NT operating system includes a Tool Bar which provides clickable buttons for available functions from the particular point that you are at in Q-VET. Once you are familiar with Q-VET, these buttons may be used as shortcuts to desired screens or functions in lieu of the menu selections or mouse button clicks suggested in the examples below.

Starting a Q-VET Session Windows NT Systems

In order to start Q-VET, issue the **vet** command in the Command Prompt window, or click on the Q-VET icon in the Q-VET program group.

Understanding the Main Window

The Q-VET Main Window is now visible.


Q-VET
File Set Up Run Scripts Options Maintenance Window Qual_Tests Help
Device Work Area
Process Work Area

NODE/DEVICE	TYPE	SUBTYPE	SIZE	STA
300isp				
cpu	CPU	INTEL PENTIUM		
memory	V_MEM		129.21 MB	
video	VIDEO	SVGA		
file	FILE_DA			

NODE/DEVICE	Proc. #	GROUP	STATUS	ERRORS	PA

Info: Q-VET version is v5.5 - Built Dec 4 1997

State: SETUP

Figure 3-1 Q-VET Main Window - Windows NT Systems

This window contains the following areas:

- Title Bar - The title bar displays the facility name (Q-VET) and the window management buttons.
- Menu Bar - The menu bar lists the menus that you select to perform Q-VET functions.
- Tool Bar - The tool bar contains buttons which are equivalent to frequently used menu entries.
- Device Work Area - The Device Work Area displays the nodes and devices whose names are contained in Q-VET's configuration database, that is, its database of testable devices.

- Process Work Area **-** The Process Work Area displays the nodes and devices you have selected for testing, along with their corresponding process numbers, test groups, and status.

  When you first start Q-VET, the Process Work Area is empty.
- Information Display Bar **-** This display bar contains two fields:

  The Info field displays various information, such as completion messages from the Q-VET sizer. Be sure to check the Info field for messages when you invoke Q-VET and when you connect to other nodes.

  The State field changes according to what state the Q-VET run is in. There are four states that Q-VET can operate in: setup, active, suspend, and qual. (The qual state is entered when the qual status window is visible.)

Q-VET sizing information will display briefly in the information display bar while Q-VET is loading.

Learning to Use Q-VET

The following sections outline how to perform basic operations in Q-VET.

Selecting Devices for a Q-VET Run

The Q-VET run is defined by the nodes or devices placed or “listed” in the Process Area of the Q-VET Main Window. To set up processes for nodes or devices in the Process Area, you must “Select” the nodes or devices. Nodes or devices may be selected in two ways:

1. Loading a predefined “Script” to create processes to test all devices on the system.
2. Choosing them selectively from the Device Area

Loading Predefined Scripts

Using the Q-VET Load option is the simplest way to test your system. The Load option automatically creates processes for most devices on your system.

To use this feature, pull down the Scripts option on the Q-VET Menu Bar, choose Load, and then select any or all of the following scripts.

Script	**Includes:**
CPU Select	All tests for your CPU
Disk Select	All tests for disks on your system.
Tape Select	All tests for tape drives on your system.
Video Select	All tests for video options on your system.
Phase 1 Engineering	All tests for all devices on your system.

Processes for the devices you selected appear in the Process Work Area.

Choosing Nodes or Devices From the Device Area

Q-VET sessions can be customized to allow you to specify which tests you would like to run. Each test can be customized using various options, provided by the Q-VET exercisers. These examples will only show how to change some of the options. To find out about other options, please refer to the online Help.

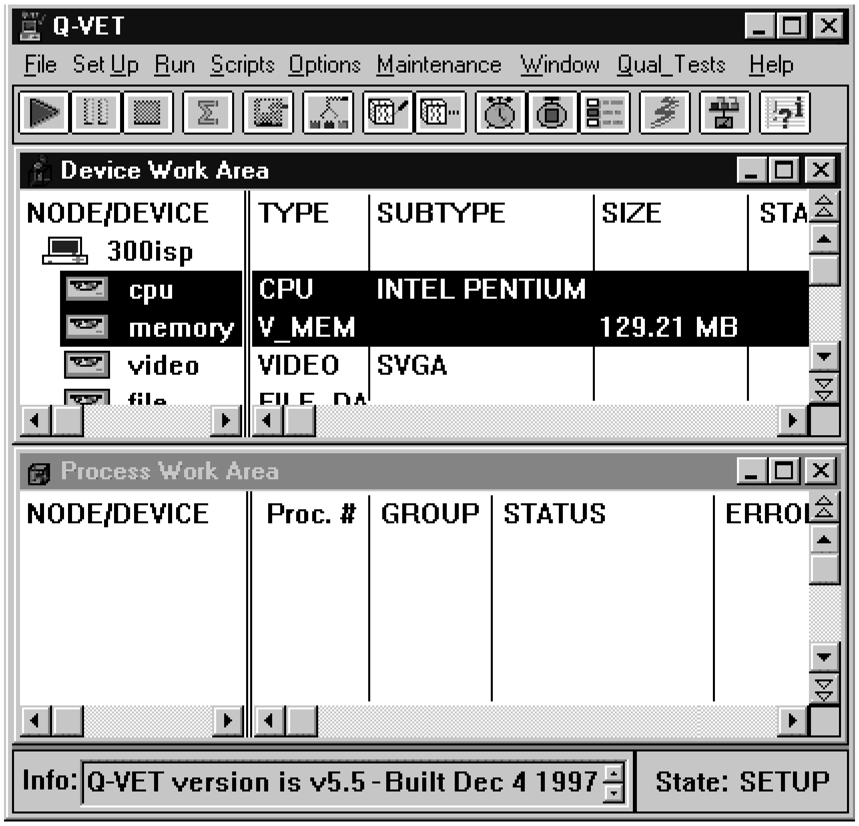
To select specific nodes or devices listed in the Device Area, you must first highlight them in one of the three ways listed.

- For a single device or node, point to it and press Mouse Button 1 (MB1), the left hand button on your mouse.
- For multiple contiguous devices, point to the first device, hold down MB1, and drag the pointer down the list of devices; release MB1 while pointing to the last device.
- For multiple noncontiguous devices, highlight the first device or series of devices by pointing or pointing and dragging. Highlight the next device or devices by pointing at them and pressing the Ctrl key on your keyboard and MB1 simultaneously [Ctrl/MB1]. You can use this technique to highlight as many noncontiguous devices as you want.

After highlighting the devices or nodes, you must select them to be listed in the Process Area by either clicking Set Up in the Menu Bar and choosing Select Devices, or placing the cursor on the highlighted item or items, clicking MB3, and choosing “Select Devices”.

To select devices from the Device Work Area for testing:

1. Highlight the names of the nodes or devices that you want to select.
   For this example, the cpu and memory devices have been highlighted.


Q-VET
File Set Up Run Scripts Options Maintenance Window Qual_Tests Help
Device Work Area
Process Work Area

NODE/DEVICE	TYPE	SUBTYPE	SIZE	STA
300isp				
cpu	CPU	INTEL PENTIUM		
memory	V_MEM		129.21 MB	
video	VIDEO	SVGA		
file	FILE_DA			

NODE/DEVICE	Proc. #	GROUP	STATUS	ERRO

Info: Q-VET version is v5.5 - Built Dec 4 1997 State: SETUP

2. Click on Set Up on the Menu Bar and choose the Select Devices entry. The process definitions are displayed in the Process Work Area.
   You can also keep the pointer in the Device Work Area and press MB3. A popup menu displays valid operations. Point to the Select Devices entry and release MB3.

The devices you selected appear in the Process Work Area.

Deselecting Processes

To deselect processes and remove them from a Q-VET run:

1. In the Process Work Area, highlight the names of the nodes or devices that you want to remove by clicking or clicking and dragging.
2. Choose the Set Up menu's Deselect entry.
3. The highlighted entries disappear from the Process Work Area.

4. You can also keep the pointer in the Process Work Area and press MB3.
   A popup menu displays valid operations. Point to the Deselect entry and release MB3.
5. The process dissapears from the Process Work Area.

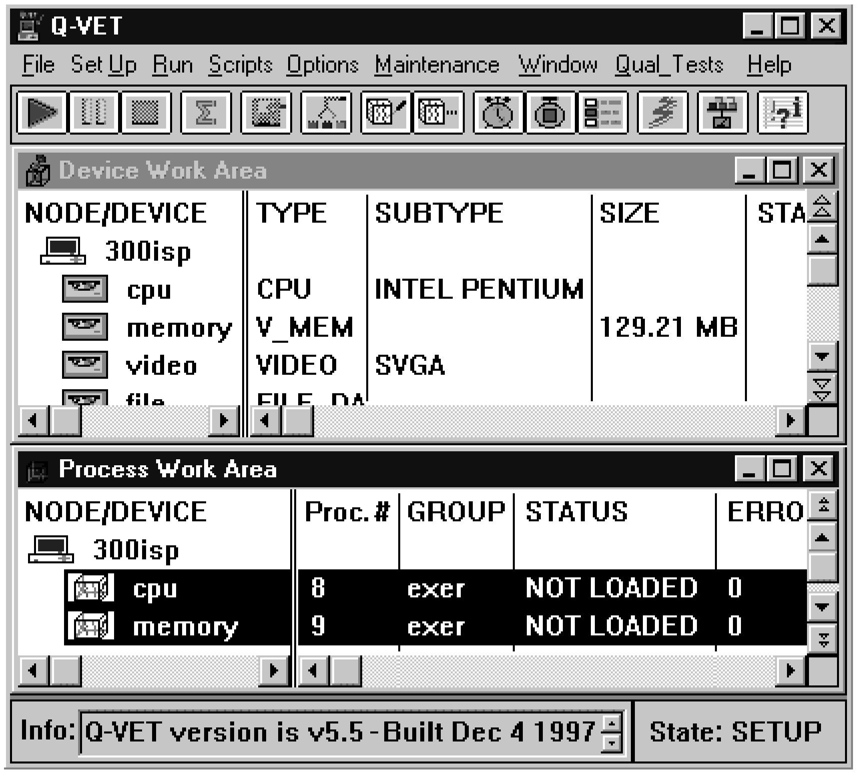
Adding and Dropping Processes

You can add processes to a Q-VET run or drop processes from it. This allows you to increase or decrease the system load gradually.

To add or drop processes:

1. In the Process Work Area, highlight the processes that you want to add to a run or drop from it.
2. Click on the Set Up menu.
3. Click on either Add or Drop.
4. You can also keep the pointer in the Process Work Area and press MB3.
   A popup menu displays valid operations. Point to the Add or Drop entry and release MB3. Examples of a Drop and an Add are shown below.

Processes highlighted for Drop:


Q-VET
File Set Up Run Scripts Options Maintenance Window Qual_Tests Help
Device Work Area
Process Work Area

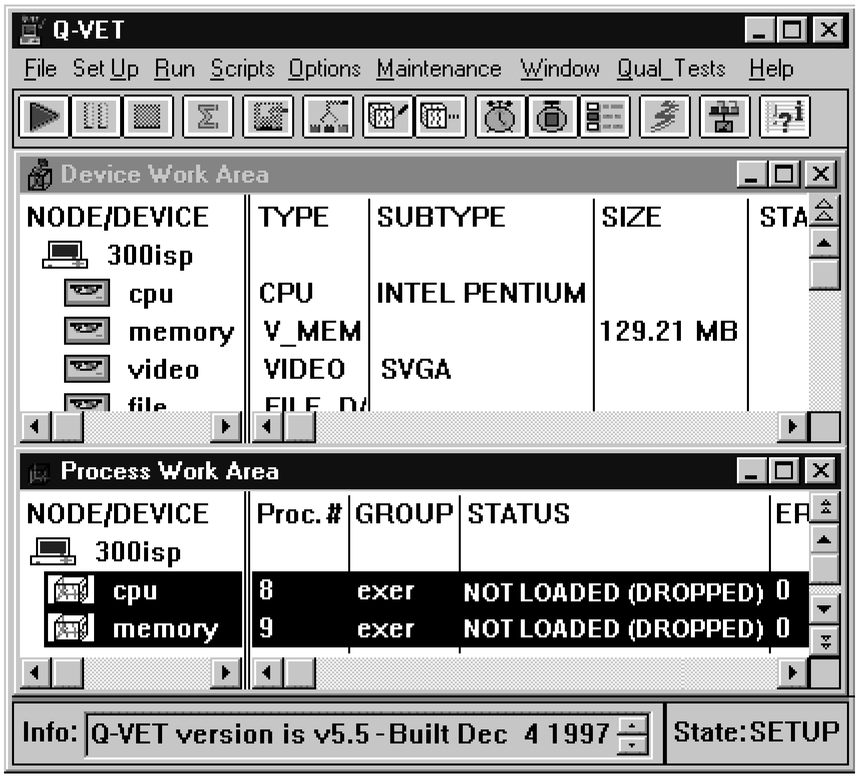
NODE/DEVICE	TYPE	SUBTYPE	SIZE	STA
300isp				
cpu	CPU	INTEL PENTIUM		
memory	V_MEM		129.21 MB	
video	VIDEO	SVGA		
file	FILE_DA			

NODE/DEVICE	Proc. #	GROUP	STATUS	ERRO
300isp				
cpu	8	exer	NOT LOADED	0
memory	9	exer	NOT LOADED	0

Info: Q-VET version is v5.5 - Built Dec 4 1997

State: SETUP

Previously dropped processes highlighted to be added:


Q-VET
File Set Up Run Scripts Options Maintenance Window Qual_Tests Help

Device Work Area

NODE/DEVICE	TYPE	SUBTYPE	SIZE	STA
300isp				
cpu	CPU	INTEL PENTIUM		
memory	V_MEM		129.21 MB	
video	VIDEO	SVGA		
file	FILE_D/			

Process Work Area

NODE/DEVICE	Proc.#	GROUP	STATUS	ER
300isp				
cpu	8	exer	NOT LOADED (DROPPED)	0
memory	9	exer	NOT LOADED (DROPPED)	0

Info: Q-VET version is v5.5 - Built Dec 4 1997 State:SETUP

All processes contained in the Process Work Area before you start the Q-VET run are automatically added. If you include additional processes *during* a Q-VET run, these processes are automatically dropped. You must add them before they begin executing.

Duplicating a Process

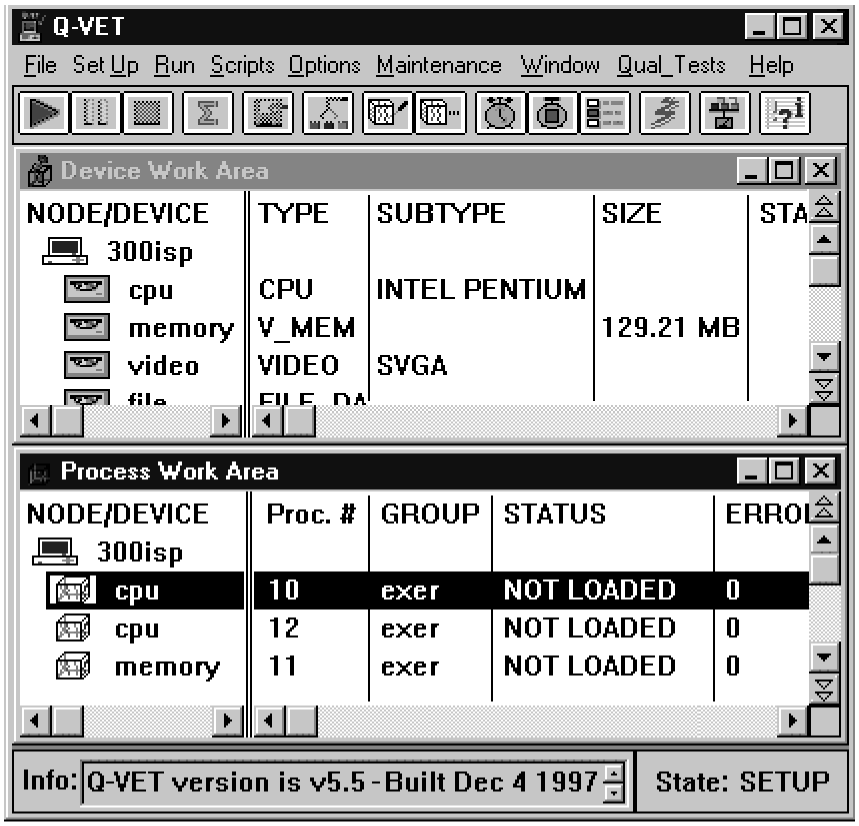
To make one or more copies of an existing process, highlight the processes you wish to duplicate in the Process Work Area, and perform one of the following:

- Choose the Set Up menu's Duplicate entry.
- Keep the pointer in the Process Work Area and press MB3.
  A popup menu displays valid operations. Point to the Duplicate entry and release MB3.

Note

Some processes can only be run one at a time; you will receive a message indicating this at start time.

The highlighted entries are duplicated in the Process Work Area.


Q-VET
File Set Up Run Scripts Options Maintenance Window Qual_Tests Help
Device Work Area
Process Work Area

NODE/DEVICE	TYPE	SUBTYPE	SIZE	STA
300isp				
cpu	CPU	INTEL PENTIUM		
memory	V_MEM		129.21 MB	
video	VIDEO	SVGA		
file	FILE_DA			

NODE/DEVICE	Proc. #	GROUP	STATUS	ERRO
300isp				
cpu	10	exer	NOT LOADED	0
cpu	12	exer	NOT LOADED	0
memory	11	exer	NOT LOADED	0

Info: Q-VET version is v5.5 - Built Dec 4 1997   State: SETUP

Repeat the above once for each duplicate you want to make.

Modifying Process Parameters

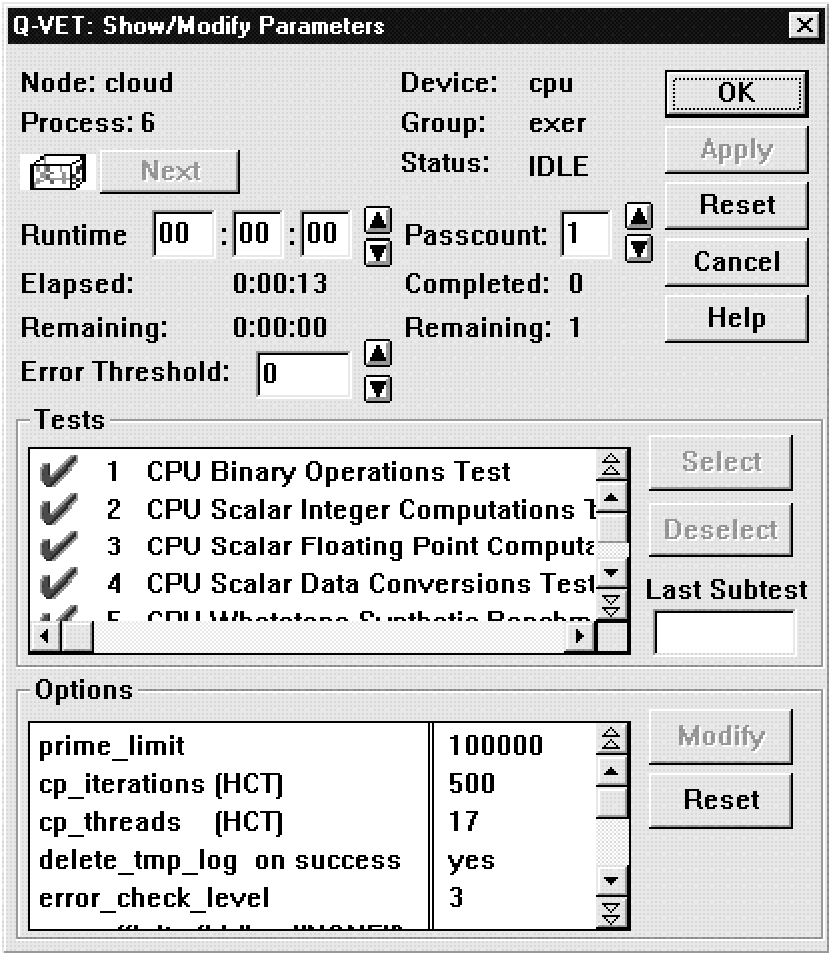
To modify a process definition's parameters:

1. In the Process Work Area, highlight the process.
2. Click on the Set Up menu.

3. Click on the menu's Show/Modify Parameters... entry.
4. You can also keep the pointer in the Process Work Area and press MB3.
   A popup menu displays valid operations . Point to the Show/Modify entry and release MB3.
   The Show/Modify Parameters dialog box is displayed.

Note

The Show/Modify Parameters dialog box and it’s dialog windows will differ from the example according to the process you have chosen to modify.



Node: cloud
Process: 6
Device: cpu
Group: exer
Status: IDLE
Runtime 00 : 00 : 00
Passcount: 1
Elapsed: 0:00:13
Completed: 0
Remaining: 0:00:00
Remaining: 1
Error Threshold: 0

Tests

✓	1	CPU Binary Operations Test
✓	2	CPU Scalar Integer Computations T
✓	3	CPU Scalar Floating Point Computa
✓	4	CPU Scalar Data Conversions Test

Last Subtest

Options

prime_limit	100000
cp_iterations (HCT)	500
cp_threads (HCT)	17
delete_tmp_log on success	yes
error_check_level	3

5. Use the dialog box to view and modify process parameters and options that apply *only to this process*. You can:

   - Set a run time and/or pass count.
   - Select or deselect specific tests within the selected test group. A check mark next to the test indicates whether it has been selected or not (the checkmark is lined out (\) if the test has been deselected).
   - Modify various process options using click boxes and fill-in dialog windows.

If you select several processes in the Process Work Area, you can use the Next button in the Show/Modify Parameters dialog box to examine and change process parameters one process at a time.

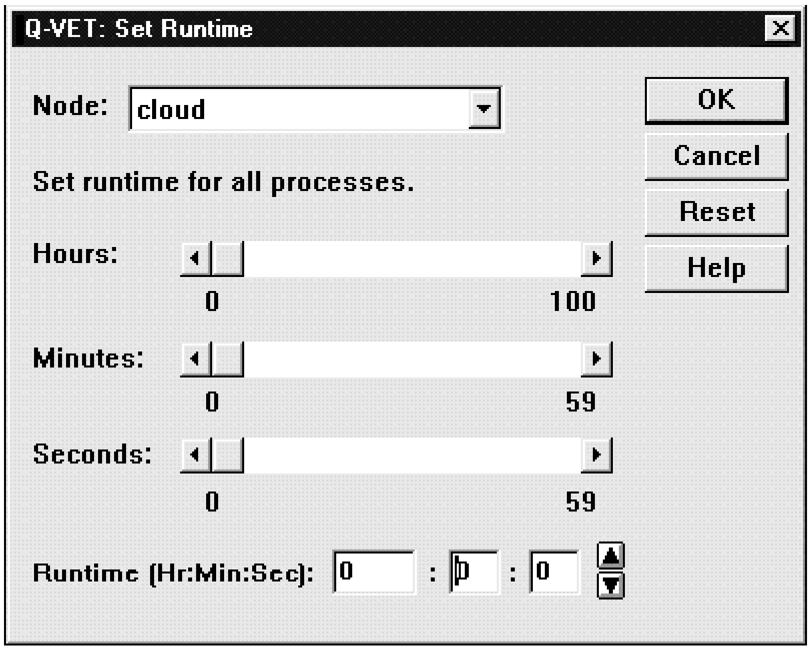
1. Click on the Set Up menu.
2. Click on the menu's Show/Modify Parameters... entry; the Show/Modify Parameters screen appears.
3. Read the contents of the displayed dialog box, and modify as necessary using the buttons and fill-in boxes.
4. Click on Next to view the dialog box associated with the next process if more than one is highlighted.
5. Click on OK, Reset, or Cancel as necessary.

Note

A list of available tests for each device is presented in Appendix D.

Setting the Run Time for All Processes

1. In the main window's Option menu choose Set Runtime...
   The Set Runtime dialog box is displayed.
2. Set the run time by sliding the slide bars, clicking on the stepper buttons, or typing numbers in the text fields.
3. Click on the OK button to activate your settings. (Click on the Reset button to restore the current run time.)


Q-VET: Set Runtime

Node: cloud

Set runtime for all processes.

Hours: 0 100

Minutes: 0 59

Seconds: 0 59

Runtime (Hr:Min:Sec): 0 : 0 : 0

Note

A distinct relationship exists between pass count and run time. If you set both pass count and run time, the first condition that is satisfied ends the Q-VET run.

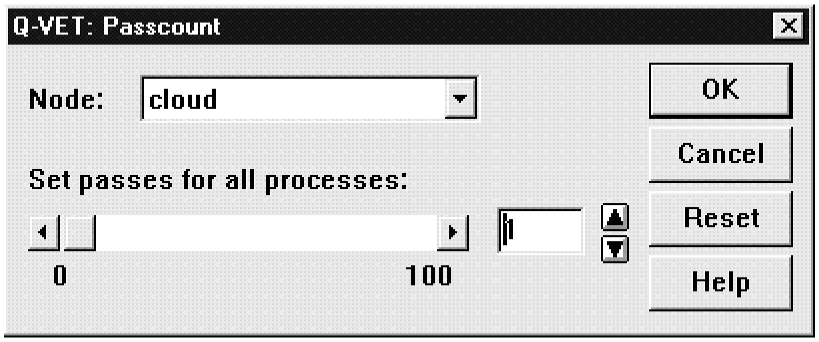
Setting the Pass Count for All Processes

To set the pass count for all processes:

1. In the main window's Options menu, choose Set Passcount...
   The Set Passcount dialog box is displayed.
2. Set the pass count by sliding the slide bar, clicking on the stepper buttons, or typing numbers in the text field. If you set the pass count at 0; Q-VETwill ignore the pass count and use only the run-time value to end the run.
3. Click on the OK button to activate your settings. (Click on the Reset button to restore the current pass count.)

Note

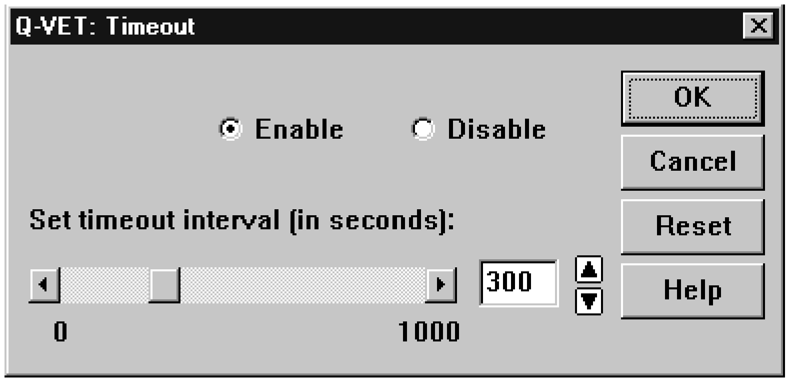
The execution time of one pass is not the same for every exerciser.



Setting Timeout for All Processes

To set the timeout interval for all processes:

1. In the main window's Options menu, choose Set Timeout...
   The Timeout dialog box is displayed.



1. Click on the Enable or Disable button.

2. If you choose to enable timeout, set the timeout interval you wish in seconds by either sliding the horizontal scale and/or typing a new upper value in the right hand box.

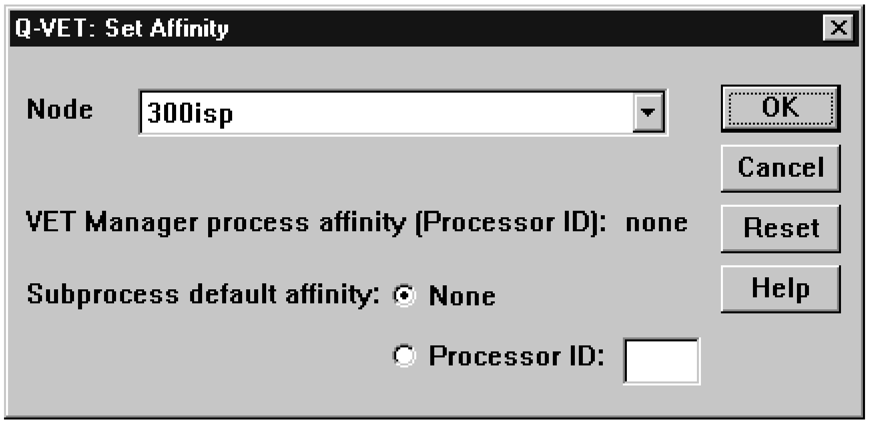
3. Click on the OK button to activate your settings. (Click on the Reset button to restore the current timeout setting.)

Setting the CPU Affinity for All Processes

CPU affinity refers to which CPU the process is bound to (set to run on).

To set the cpu affinity for all processes:

1. In the main window's Options menu, choose Set Affinity...
   The Set Affinity dialog box is displayed.



Node 300isp

VET Manager process affinity (Processor ID): none

Subprocess default affinity: [✓ None] Processor ID:

1. Choose the node if there are more than one.

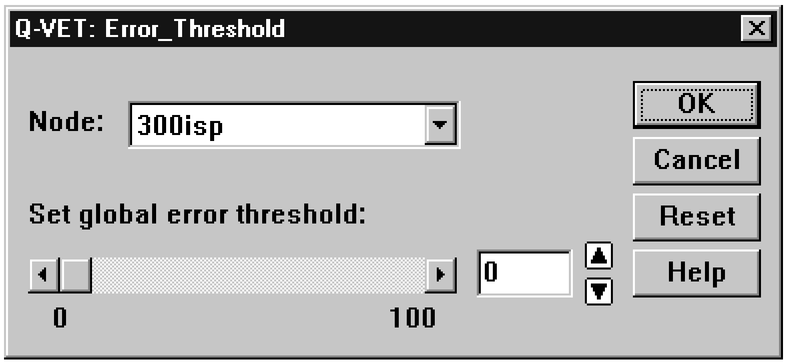
2. Set the subprocess default affinity processor id by clicking on the empty box, then entering the appropriate id number.

3. Click on the OK button to activate your settings. (Click on the Reset button to restore the current affinity id.)

Setting the Error Threshold for All Processes

To set the error threshold for all processes:

1. In the main window's Options menu, choose Set Error_Threshold...
   The Error Threshold dialog box is displayed.



Node: 300isp

Set global error threshold: 0

1. Click on the appropriate Node.

2. Set the error threshold you wish by either sliding the horizontal scale and/or typing a new upper value in the right hand box.

3. Click on the OK button to activate your settings. (Click on the Reset button to restore the current error threshold setting.)

Modifying Other Options

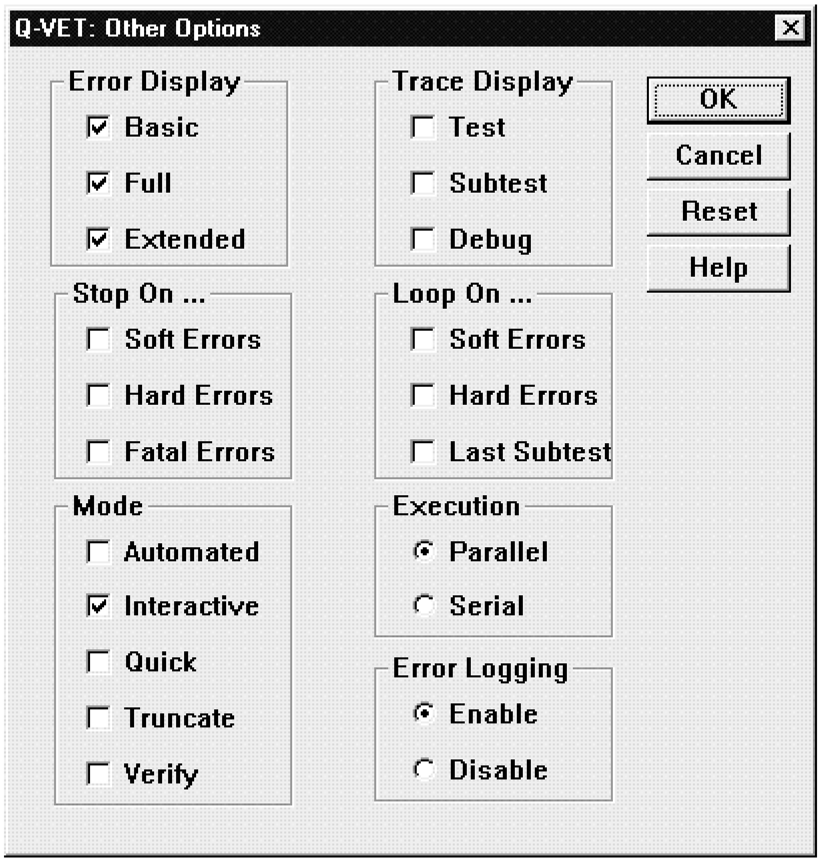
If you choose the Options menu's Other... entry, a dialog box is displayed that lets you modify several Q-VET run-time options. Click on the dialog box's Help button to read about the options.

To modify Other options:

1. Click on Options in the Menu bar.

2. Click on the Other... entry.

The Q-VET Other Options Dialogue Window is displayed.



Error Display: [✓ Basic] [✓ Full] [✓ Extended]

Trace Display: Test, Subtest, Debug

Stop On ...: Soft Errors, Hard Errors, Fatal Errors

Loop On ...: Soft Errors, Hard Errors, Last Subtest

Mode: Automated [✓ Interactive] Quick, Truncate, Verify

Execution: [✓ Parallel] Serial

Error Logging: [✓ Enable] Disable

To set Other options, do the following.

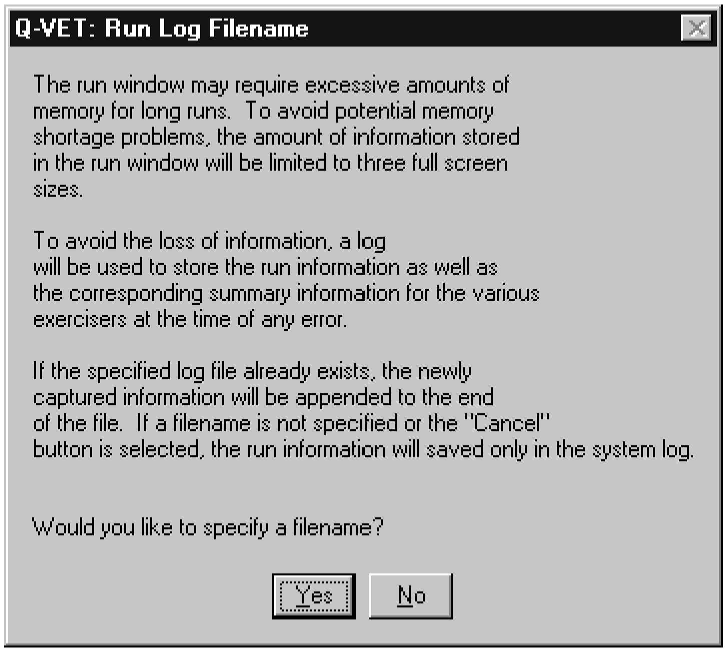
1. Click on the desired option button(s).
2. Click on the OK button.

Starting a Q-VET Run

To start a Q-VET run, click on Run in the Menu Bar and choose Start All. This starts the run that was set up by the Select or Load options.

If this is your first run in this Q-VET session, a Run Log Filename screen will appear. This screen asks if you wish to specify an error log and run information filename. If you do not specify a filename, the run information will only be saved in the system log. If you do specify a filename, information displayed in the run window during this and all subsequent runs in this Q-VET session will be appended to the specified file.

If you wish to specify a filename, click Yes.


Q-VET: Run Log Filename

The run window may require excessive amounts of memory for long runs. To avoid potential memory shortage problems, the amount of information stored in the run window will be limited to three full screen sizes.

To avoid the loss of information, a log will be used to store the run information as well as the corresponding summary information for the various exercisers at the time of any error.

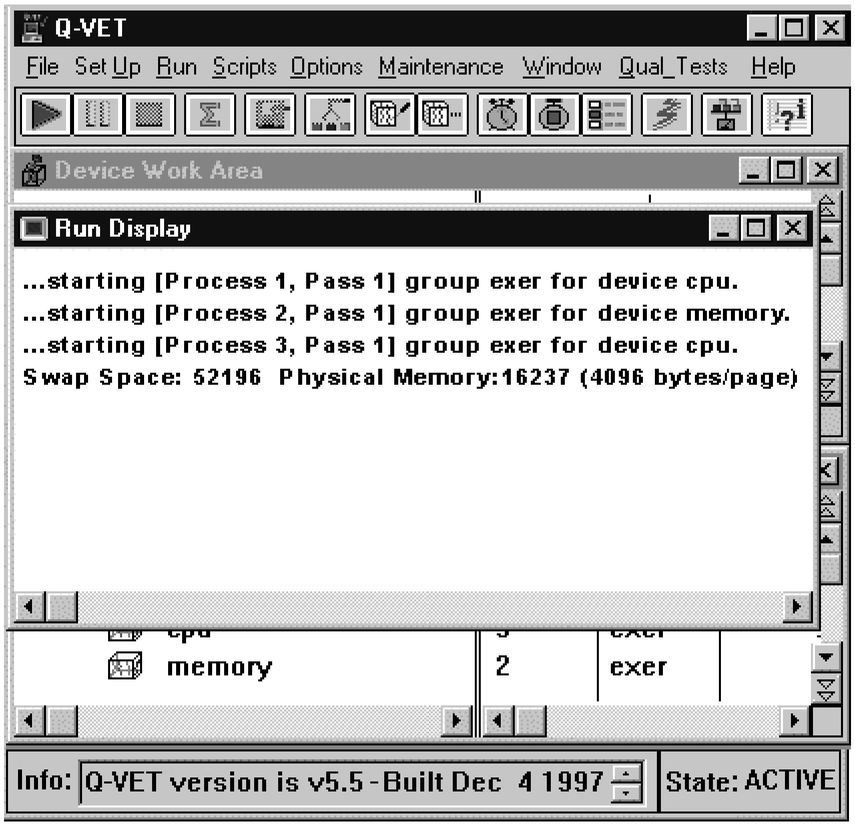
If the specified log file already exists, the newly captured information will be appended to the end of the file. If a filename is not specified or the "Cancel" button is selected, the run information will saved only in the system log.

Would you like to specify a filename?

Note

If you have previously started a test run during this session, Q-VET skips the Run Log Filename Screen. (If you wish to change the log file location, you may exit Q-VET and start a new session to obtain the Run Log Filename screen).
If a run was started using the Qual Menu (see Running Q-VET Qual Tests), the run log filename will automatically be chosen as `c:/dec/vet/tool_logs/vet_err_nnnnnn-mm-dd-yyyy.log`, where c: is the drive Q-VET is installed on, nnnnnn is the nodename of the system under test, and mm-dd-yyyy is the date when the test was started.

The Q-VET Run Display window appears. It shows the events that occur during the run---process start, error reports, and process completion.


Q-VET
File Set Up Run Scripts Options Maintenance Window Qual_Tests Help
Device Work Area
Run Display
cpu
memory
2
exer
Info:
State: ACTIVE

```
...starting [Process 1, Pass 1] group exer for device cpu.
...starting [Process 2, Pass 1] group exer for device memory.
...starting [Process 3, Pass 1] group exer for device cpu.
Swap Space: 52196  Physical Memory:16237 (4096 bytes/page)
```

Q-VET version is v5.5 - Built Dec 4 1997

Wait for the run to complete, or suspend and terminate the test when you desire.

Viewing Test Results

Test results can appear in many places. Error messages are displayed in the Run Display window. The Summary window contains brief error messages and run statistics. The System Event Log contains errors that were detected by the system during the run.

Note

When the session is terminated, a summary file containing test summary information and any system errors which may have occured during the test run is generated. This summary file can be found in the \dec\vet\tool_logs directory in DIGITAL UNIX, and its equivalent in Open VMS, sys$specific:[sysmgr.tool_logs].

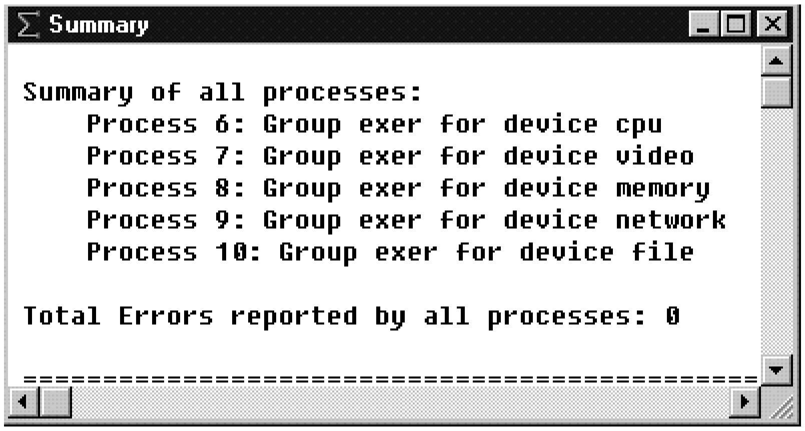
Viewing a Run Summary

You can view a summary of the run any time during the run or after its completion.

To view a run summary:

1. Click on Window in the Menu Bar, and choose Summary Display.

   A summary window is displayed showing the state of the Q-VET run at the time you invoked the window.



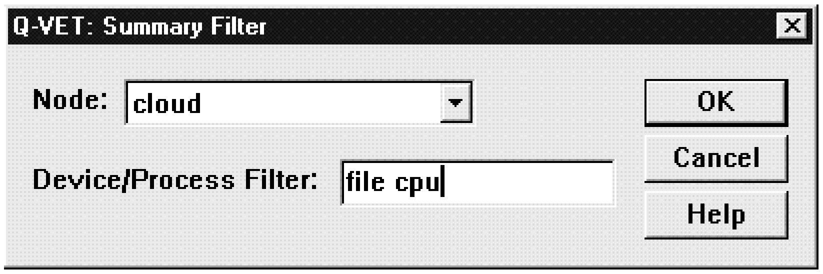
```
Summary of all processes:
    Process 6: Group exer for device cpu
    Process 7: Group exer for device video
    Process 8: Group exer for device memory
    Process 9: Group exer for device network
    Process 10: Group exer for device file

Total Errors reported by all processes: 0

===============================================
```

During a run you can view a summary for any individual device or process by using the Summary Filter as follows:

1. While viewing the Summary Window, Select the Summary Filter... entry from the Windows menu.
   The Summary Filter dialog box appears which allows you to specify individual devices for which you would like a summary.



Node: cloud

Device/Process Filter: file cpu

2. Enter the desired device or process in the Device/Process Filter field.
3. Click on OK.

4. The Summary Window is updated to show information for the individual device or process you requested.

Note

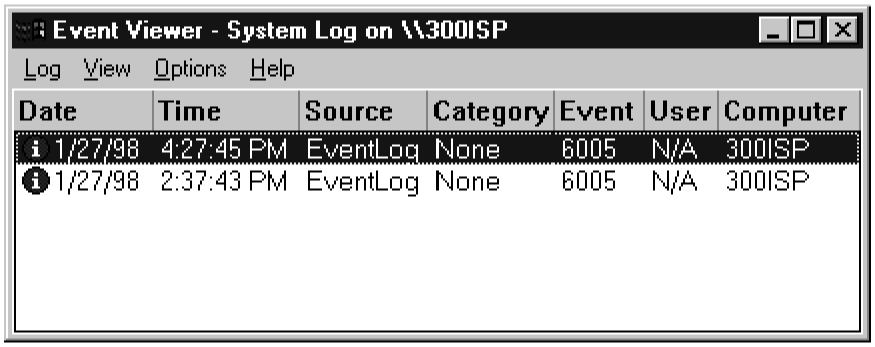
The summary filter affects all summary displays until you clear the filter. Clearing the filter field causes a summary for all devices to be displayed.

While viewing the summary, you can click on the File menu Save As option to save the summary in a file.

Viewing the System Event Log

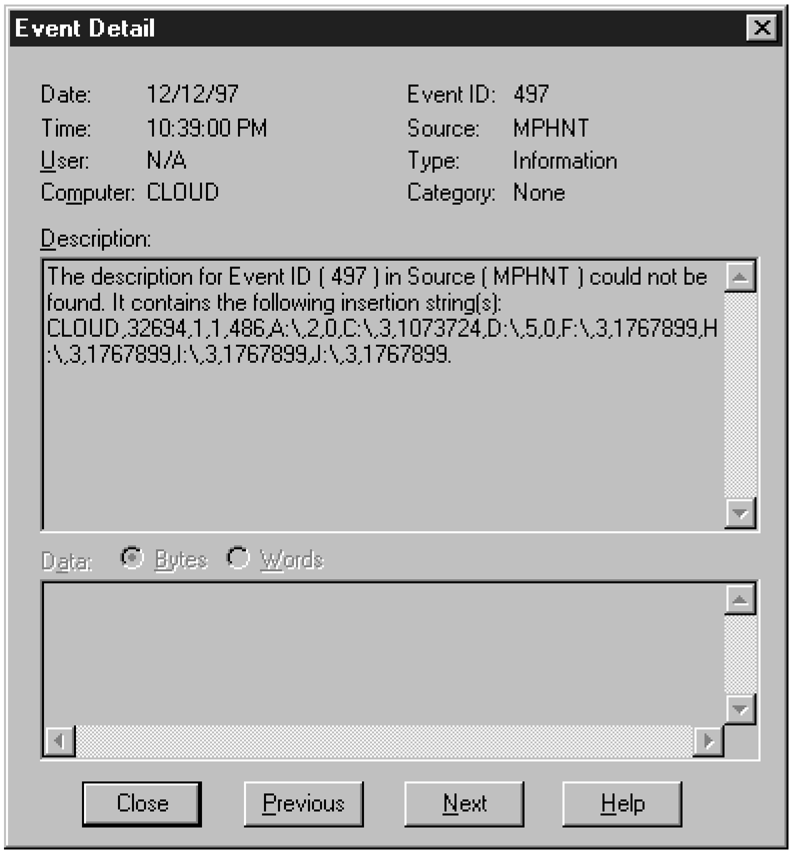
To view the System Event Log:

1. Click on Window in the Main Menu and choose Event Viewer.



Date	Time	Source	Category	Event	User	Computer
1/27/98	4:27:45 PM	EventLog	None	6005	N/A	300ISP
1/27/98	2:37:43 PM	EventLog	None	6005	N/A	300ISP

2. To get a detailed description of an entry:
3. Double click on an entry in the Event Viewer Window. An Event Detail window will be displayed.


Event Detail

Date: 12/12/97
Time: 10:39:00 PM
User: N/A
Computer: CLOUD
Event ID: 497
Source: MPHNT
Type: Information
Category: None

Description:

```
The description for Event ID ( 497 ) in Source ( MPHNT ) could not be
found. It contains the following insertion string(s):
CLOUD,32694,1,1,486,A:\,2,0,C:\,3,1073724,D:\,5,0,F:\,3,1767899,H
:\,3,1767899,I:\,3,1767899,J:\,3,1767899.
```

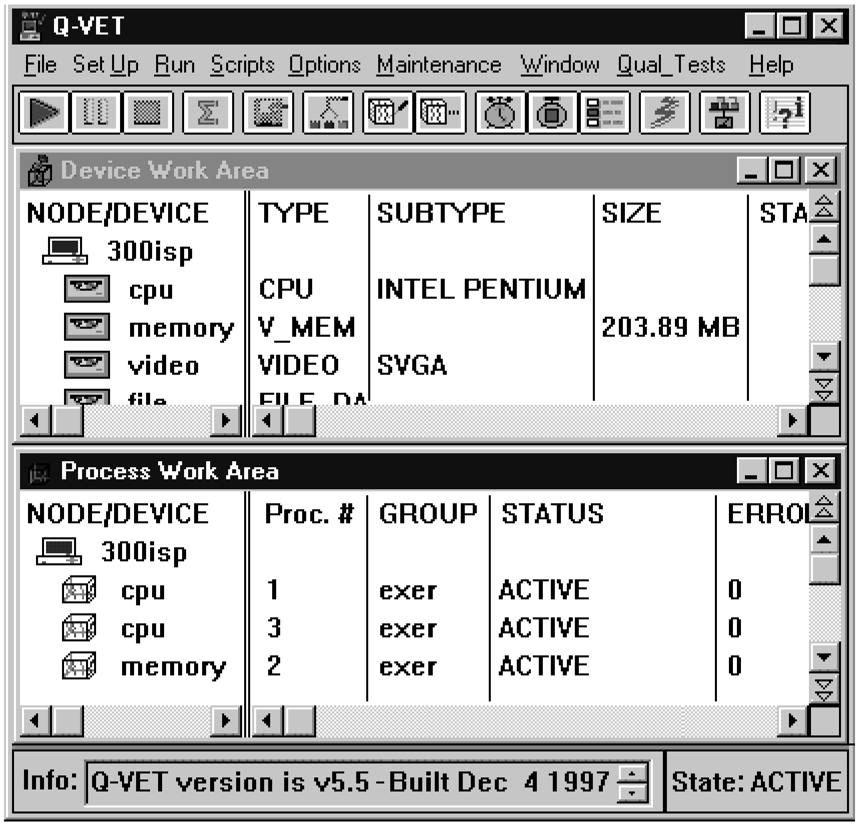
4. Click on Close to exit the Event Detail Window.
5. Select the exit entry from the Log menu to exit the Event Viewer Window.

Suspending, Continuing, and Terminating a Q-VET Run or Process

You can suspend, continue, or terminate an entire Q-VET run, or individual processes by choosing Run from the Menu Bar. The Run Menu is divided into upper and lower sections. The upper section lists Start All, Suspend, Continue All, and Terminate All; these choices affect the entire Q-VET run, i.e. all processes whether highlighted or not. The lower section of the Run Menu lists Suspend, Continue and Terminate; these choices affect individual processes which have been highlighted.

Suspending, Continuing, and Terminating a Q-VET Run

Click on Run in the Menu Bar to suspend a Q-VET run, continue a suspended run, or terminate an entire Q-VET run.

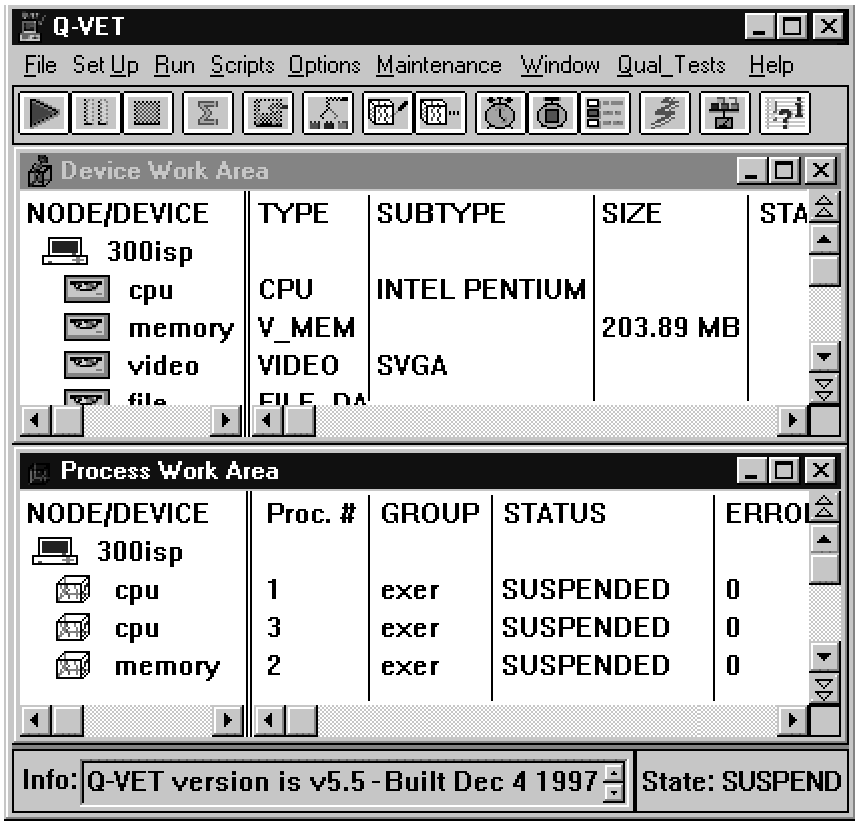

Q-VET
File SetUp Run Scripts Options Maintenance Window Qual_Tests Help
Device Work Area
Process Work Area

NODE/DEVICE	TYPE	SUBTYPE	SIZE	STA
300isp				
cpu	CPU	INTEL PENTIUM		
memory	V_MEM		203.89 MB	
video	VIDEO	SVGA		
file	FILE_DA			

NODE/DEVICE	Proc. #	GROUP	STATUS	ERRO
300isp				
cpu	1	exer	ACTIVE	0
cpu	3	exer	ACTIVE	0
memory	2	exer	ACTIVE	0

Info: Q-VET version is v5.5 - Built Dec 4 1997 State: ACTIVE

Choosing Suspend from the upper section of the Run menu suspends all Q-VET internal tests. (Note that it does not suspend external tests started by Q-VET processes.) Each one's status is listed as suspended in the Process Work Area.


Q-VET
File Set Up Run Scripts Options Maintenance Window Qual_Tests Help
Device Work Area
Process Work Area

NODE/DEVICE	TYPE	SUBTYPE	SIZE	STA
300isp				
cpu	CPU	INTEL PENTIUM		
memory	V_MEM		203.89 MB	
video	VIDEO	SVGA		
file	FILE_DA			

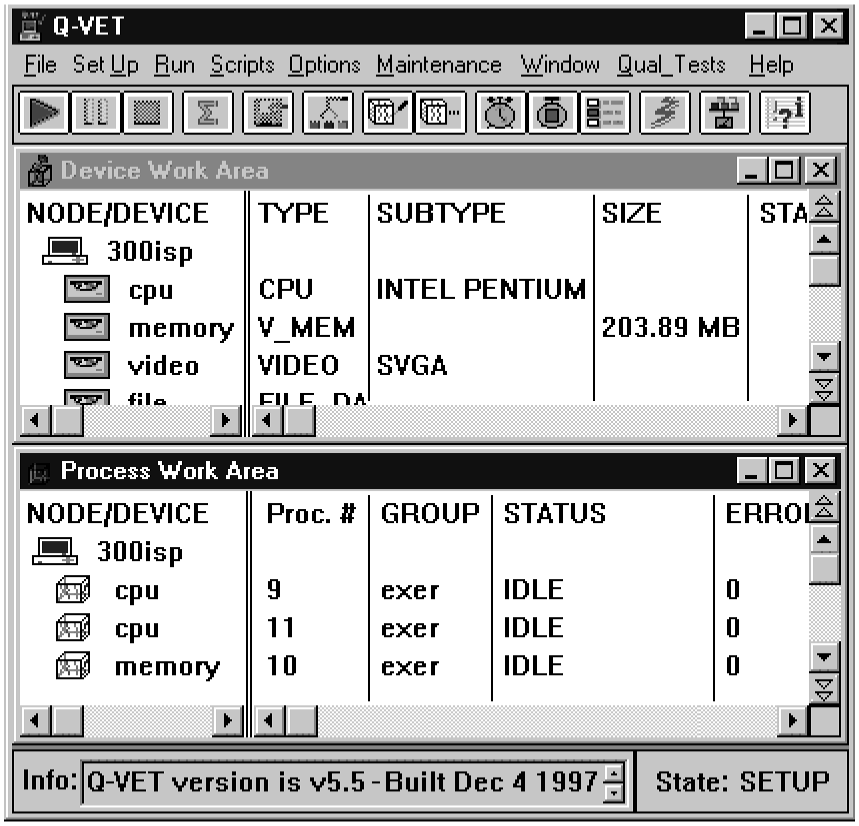
NODE/DEVICE	Proc. #	GROUP	STATUS	ERRO
300isp				
cpu	1	exer	SUSPENDED	0
cpu	3	exer	SUSPENDED	0
memory	2	exer	SUSPENDED	0

Info: Q-VET version is v5.5 - Built Dec 4 1997 State: SUSPEND

Choosing Continue All from the upper section of the Run menu causes all processes to resume executing. Each one's status is listed as active in the Process Work Area.

To terminate a Q-VET run while it is running, you must first suspend the processes which are running.

Click on the Run Menu and choose Suspend then choose Terminate All. This terminates the execution of all processes whether or not they have completed. Each one's status is listed as idle in the Process Work Area.


Q-VET
File Set Up Run Scripts Options Maintenance Window Qual_Tests Help
Device Work Area
Process Work Area

NODE/DEVICE	TYPE	SUBTYPE	SIZE	STA
300isp				
cpu	CPU	INTEL PENTIUM		
memory	V_MEM		203.89 MB	
video	VIDEO	SVGA		
file	FILE_DA			

NODE/DEVICE	Proc. #	GROUP	STATUS	ERRO
300isp				
cpu	9	exer	IDLE	0
cpu	11	exer	IDLE	0
memory	10	exer	IDLE	0

Info: Q-VET version is v5.5 - Built Dec 4 1997 State: SETUP

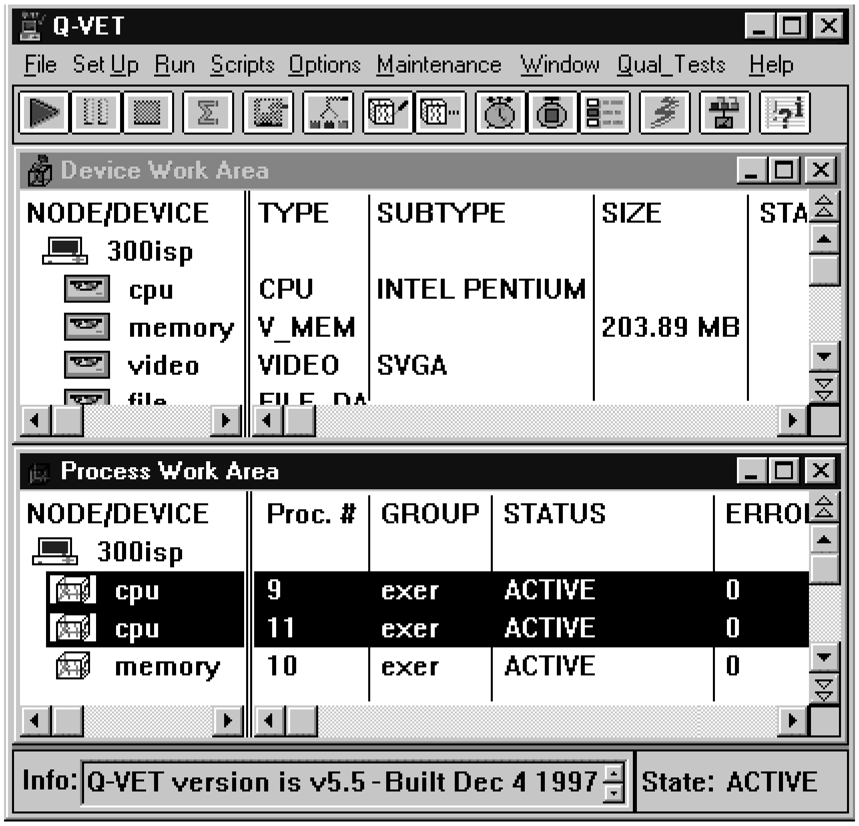
Suspending, Continuing, and Terminating Individual Processes

After you have started a run, you can suspend, continue, or terminate individual processes by using the lower section of Run menu entries. . . .

To suspend, continue, or terminate individual processes:

1. In the Process Work Area, highlight the names of the processes that you want to suspend, continue, or terminate.
2. Click on the Run menu.
3. In the lower section of the Run menu, click on Suspend, Continue, or Terminate as desired.

In this example, two cpu processes have been highlighted.

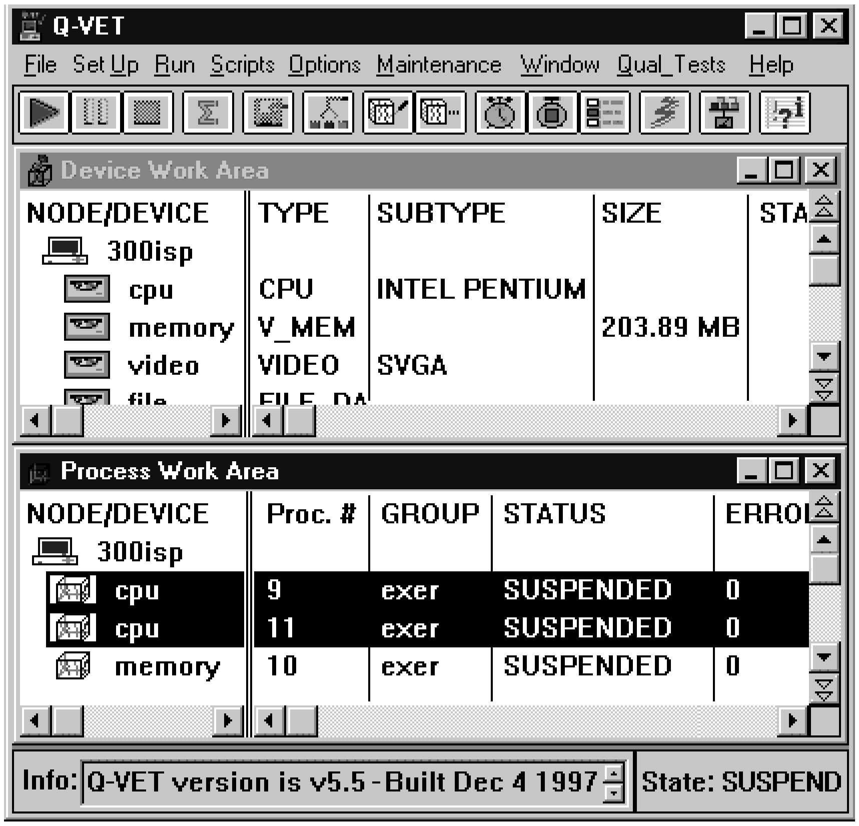

Q-VET
File Set Up Run Scripts Options Maintenance Window Qual_Tests Help
Device Work Area
Process Work Area

NODE/DEVICE	TYPE	SUBTYPE	SIZE	STA
300isp				
cpu	CPU	INTEL PENTIUM		
memory	V_MEM		203.89 MB	
video	VIDEO	SVGA		
file	FILE_DA			

NODE/DEVICE	Proc. #	GROUP	STATUS	ERRO
300isp				
cpu	9	exer	ACTIVE	0
cpu	11	exer	ACTIVE	0
memory	10	exer	ACTIVE	0

Info: Q-VET version is v5.5 - Built Dec 4 1997 State: ACTIVE

Choose the Suspend option in the lower section of the Run menu to stop the highlighted processes. Q-VET stops the processes.

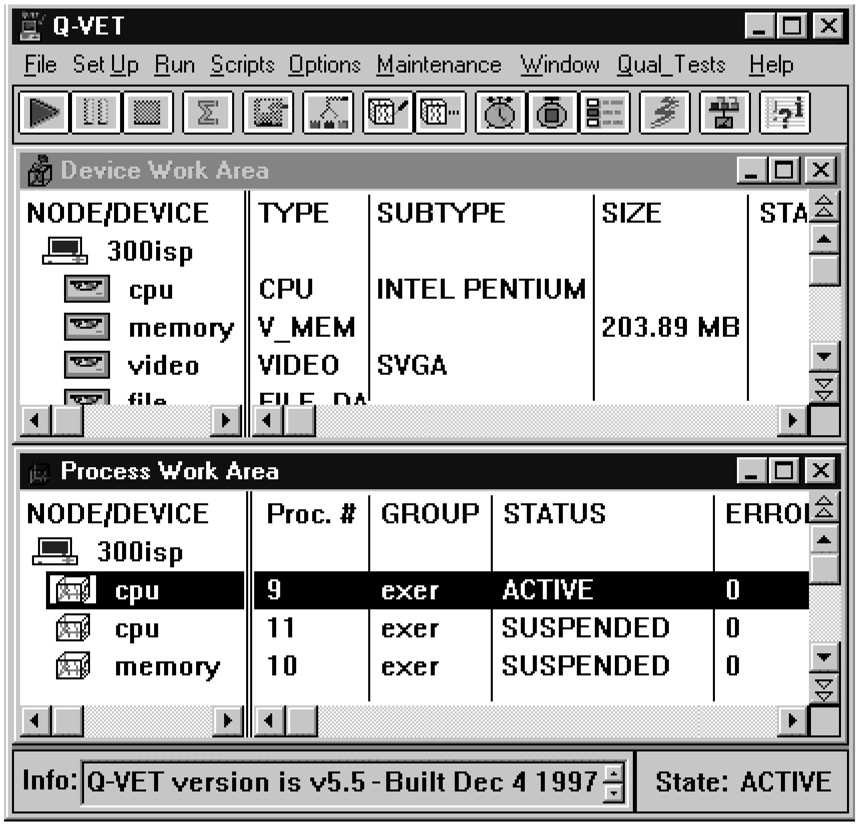

Q-VET
File Set Up Run Scripts Options Maintenance Window Qual_Tests Help
Device Work Area
Process Work Area

NODE/DEVICE	TYPE	SUBTYPE	SIZE	STA
300isp				
cpu	CPU	INTEL PENTIUM		
memory	V_MEM		203.89 MB	
video	VIDEO	SVGA		
file	FILE_DA			

NODE/DEVICE	Proc. #	GROUP	STATUS	ERRO
300isp				
cpu	9	exer	SUSPENDED	0
cpu	11	exer	SUSPENDED	0
memory	10	exer	SUSPENDED	0

Info: Q-VET version is v5.5 - Built Dec 4 1997 State: SUSPEND

In this example, all processes have been suspended; and one cpu process has been highlighted and continued by selecting the Continue option from the lower Run menu.

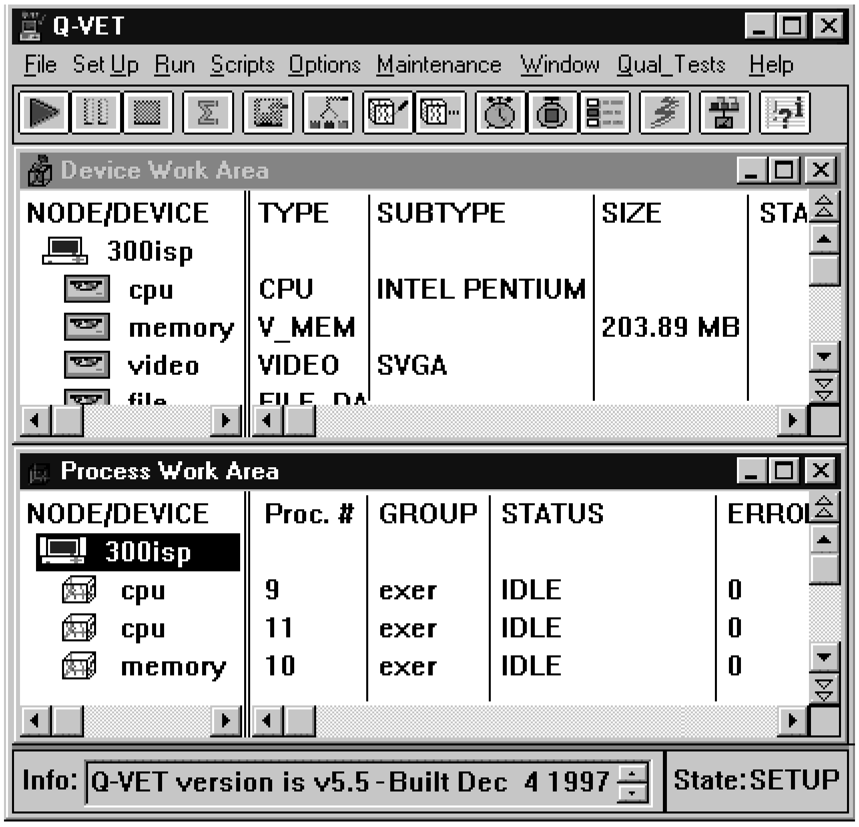

Q-VET
File Set Up Run Scripts Options Maintenance Window Qual_Tests Help
Device Work Area
Process Work Area

NODE/DEVICE	TYPE	SUBTYPE	SIZE	STA
300isp				
cpu	CPU	INTEL PENTIUM		
memory	V_MEM		203.89 MB	
video	VIDEO	SVGA		
file	FILE_DA			

NODE/DEVICE	Proc. #	GROUP	STATUS	ERRO
300isp				
cpu	9	exer	ACTIVE	0
cpu	11	exer	SUSPENDED	0
memory	10	exer	SUSPENDED	0

Info: Q-VET version is v5.5 - Built Dec 4 1997 State: ACTIVE

In this example, all processes have been highlighted and terminated by selecting the Terminate option from the Run menu.


Q-VET
File Set Up Run Scripts Options Maintenance Window Qual_Tests Help
Device Work Area
Process Work Area

NODE/DEVICE	TYPE	SUBTYPE	SIZE	STA
300isp				
cpu	CPU	INTEL PENTIUM		
memory	V_MEM		203.89 MB	
video	VIDEO	SVGA		
file	FILE_DA			

NODE/DEVICE	Proc. #	GROUP	STATUS	ERRO
300isp				
cpu	9	exer	IDLE	0
cpu	11	exer	IDLE	0
memory	10	exer	IDLE	0

Info: Q-VET version is v5.5 - Built Dec 4 1997 State: SETUP

Saving, Clearing, and Restoring Setup

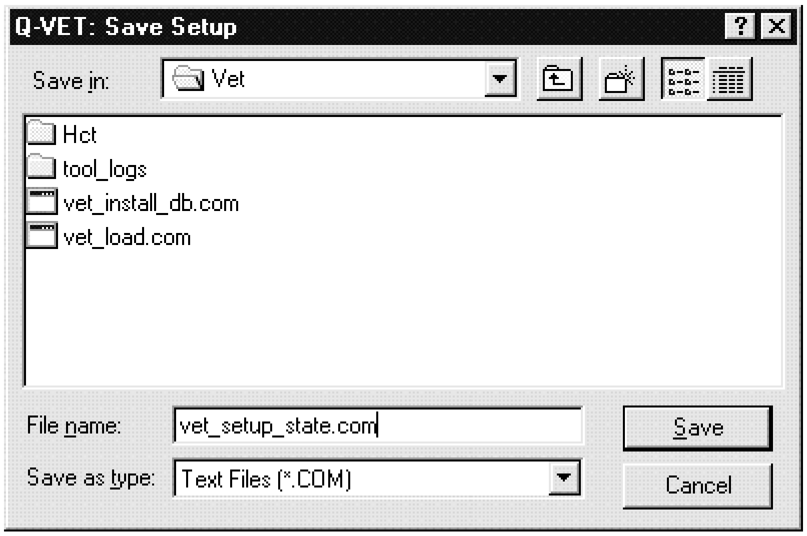
You can save or clear the current configuration displayed in the Device Work and Process Work Areas, current process parameters, and Q-VET options, and you can recall this saved state later during the current Q-VET session or in future ones.

Saving a Setup State

To save the current setup state, follow these steps.

1. Click on the Set Up menu.
2. Click on the Set Up menu's Save Setup entry.

   The Save Setup dialog box is displayed.



3. You can save the setup in the default file whose name is provided, or you can save the setup in a file of another name:

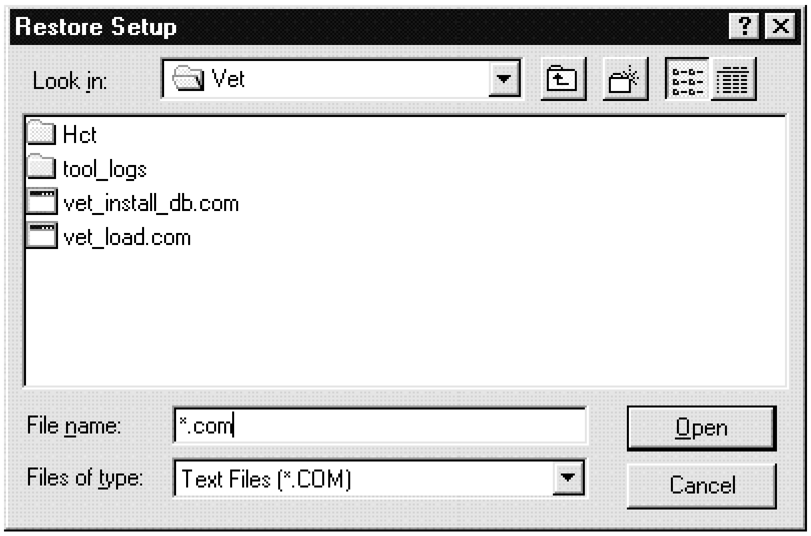
- Point within the File Name selection field and click on MB1.
- Delete the file name that already appears.
- Enter the name of the file in which you want to save the setup.

4. Click on the Save button to save the state in the file.

Restoring a Setup State

To restore a saved setup state:

1. Click on the Set Up menu. Click on the menu's Restore Setup entry. The Restore Setup dialog box is displayed.



1. Specify a file name by clicking on a file name in the list or by typing a file name in the File Name field.
2. Click on the Open button.

   Q-VET clears the current setup state and restores the state contained in the specified file.

The contents of the Device and Process work areas are first cleared before the saved devices and processes are restored.

Editing and Executing Script Files

You can edit and execute script files by selecting the Scripts menu. A script or command file is a file containing commands that Q-VET can execute.

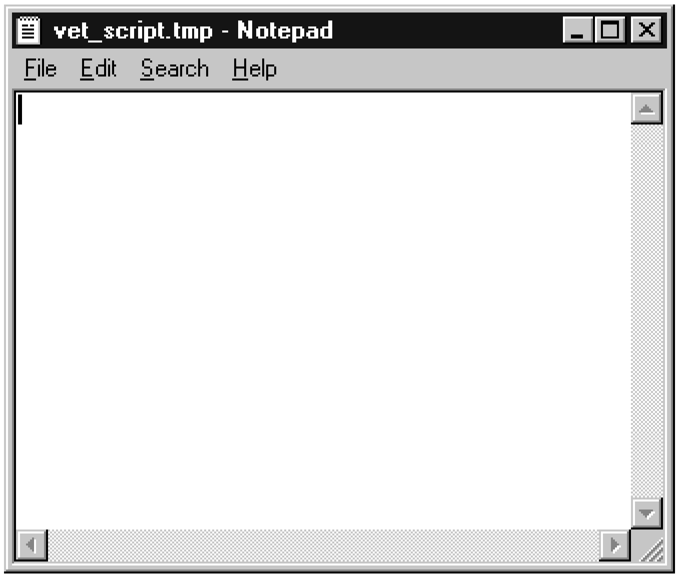
Use the script buffer to create and execute temporary scripts. Use script files to save your scripts permanently.

To edit the script buffer:[6]

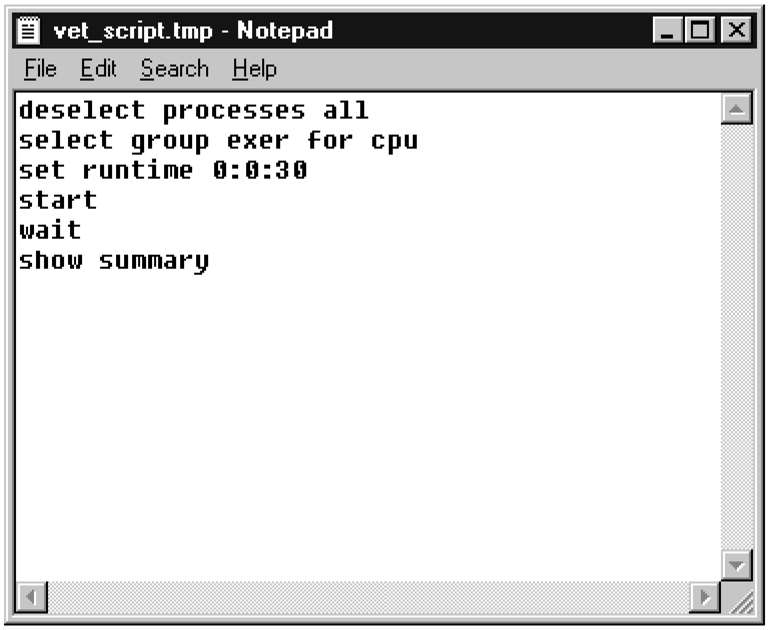
1. Click on Scripts in the Menu Bar.
2. Click on the menu's Edit entry.

[6]The script buffer's contents are not permanent. The first time you issue the **edit** command during a Q-VET session, the script buffer is empty. The commands you enter can be overwritten during the session. The script buffer's contents are lost when you terminate the session.

3. Choose the Script Buffer... entry of the submenu.
   The Script Buffer dialog box is displayed.



4. Activate the text field by pointing within it and clicking on MB1.
5. Enter one Q-VET command per line in the text field.



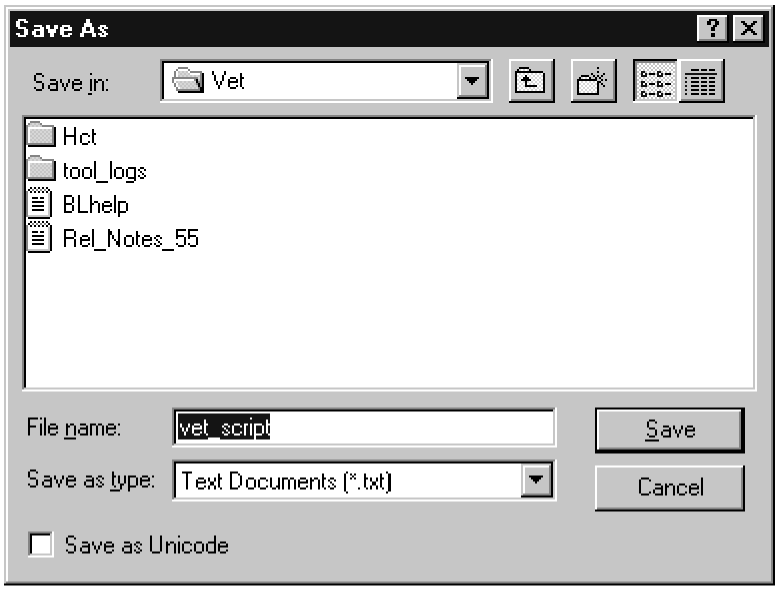
```
deselect processes all
select group exer for cpu
set runtime 0:0:30
start
wait
show summary
```

To execute commands in the script buffer:

1. Click on the Scripts menu.
2. Click on the menu's Execute entry.
3. Choose the Script Buffer entry of the submenu. Q-VET will now execute the commands in the script buffer..

To create a permanent script file, do the following:

1. Save the script buffer by pulling down the File Menu and choosing either Save or (Save As to give it a filename you select). The Save Script screen appears.

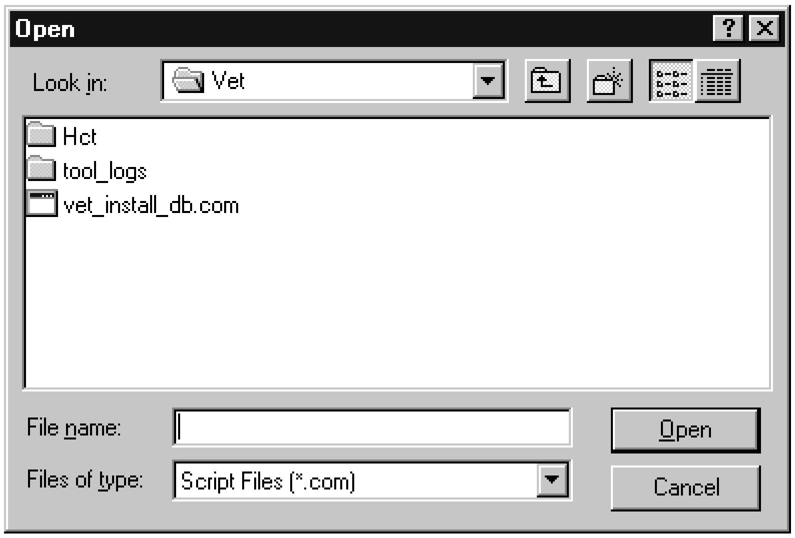


2. Click on Save.

To edit a permanent script file, do the following

1. Click on the Scripts menu.
2. Choose the menu's Edit entry.
3. Choose the File... entry of the submenu.

The Edit Script dialog box is displayed, so that you can select a file name.



To execute a permanent script file:

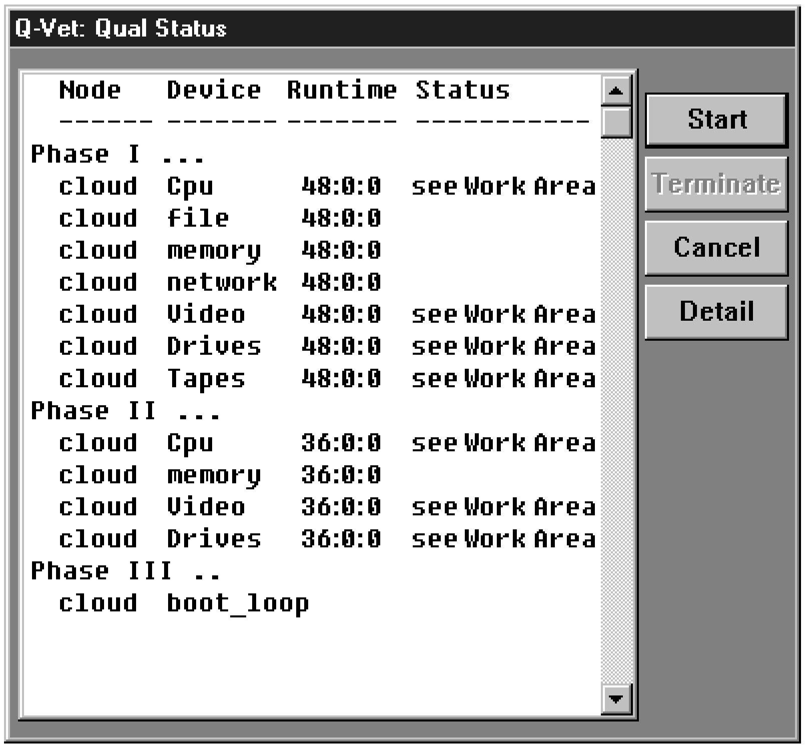
1. Click on the Scripts menu.
2. Choose the menu's Execute entry.
3. Choose the File... entry of the submenu.
   The Execute Script dialog box is displayed, so that you can select a file. (This is essentially the same screen as the Edit Script dialog box.)

Running Q-VET Qual Tests

Q-VET provides several predefined Qual Tests. You can run Q-VET Qual Tests by doing the following:

1. Click on Qual_Tests on the Menu Bar.
2. Choose “Cert. Tests” from the pull down menu.
3. Choose “Engineering” from the next pull down menu.

The Q-VET Qual Status screen appears.



```
  Node   Device  Runtime Status
  ------ ------- ------- -----------
Phase I ...
  cloud  Cpu      48:0:0  see Work Area
  cloud  file     48:0:0
  cloud  memory   48:0:0
  cloud  network  48:0:0
  cloud  Video    48:0:0  see Work Area
  cloud  Drives   48:0:0  see Work Area
  cloud  Tapes    48:0:0  see Work Area
Phase II ...
  cloud  Cpu      36:0:0  see Work Area
  cloud  memory   36:0:0
  cloud  Video    36:0:0  see Work Area
  cloud  Drives   36:0:0  see Work Area
Phase III ..
  cloud  boot_loop
```

4. Use the Start or Terminate buttons to start or terminate the Qual Tests. The Cancel button lets you exit the screen without taking action, and the Detail button lets you view the qual test details for a particular device.

Terminating a Q-VET Session

You can exit the Q-VET session by clicking on the Exit entry in the File Menu.

4 Q-VET Command Interface

This chapter discusses the Q-VET command interface which is applicable for all operating systems - DIGITAL UNIX, OpenVMS, and Windows NT.

Online Help

When you need assistance with a specific Q-VET command, you can issue **help** *command-name* to view a description of the command.

For this example, enter **`help load`**

```
Q-VET_setup> help load

         Creates a default load for the system configuration that
         you are testing. You then issue the start command to start
         the run.

 Format:

load { none                                              }
     { with default                                      }
     { with [cpu n1] [memory n2] [file n3] [network n4] [video n5]
      [disk n6]  }

   When you issue the load command without arguments, the Q-VET Software
    .
    .
    .

Q-VET_setup>
```

Issuing the **help** command without arguments lists available topics about which you can obtain information.

Getting Started with the Q-VET Command Interface

This section reproduces a Q-VET session using its command interface. The session has been designed so that you can follow the examples.

The session does not use every available command-line function display. You can also use Q-VET's online help (see Online Help) to find out about commands, keywords, and arguments.

Starting Q-VET Session

Note

Before you start Q-VET, make sure that you have followed the system setup procedures described in Chapter 2 for the DIGITAL UNIX and OpenVMS operating systems, and Chapter 3 for the Windows NT operating system. This includes making sure that you have the proper corresponding versions of Q-VET and DIGITAL UNIX, OpenVMS, or Windows NT installed.

Issue the **`vet`** command at your system prompt:

```
% vet -nw
```

You need not specify `-nw`, if your DIGITAL UNIX system does not support DECwindows Motif or you have not set the DISPLAY environment variable. (See Chapter 2.)

After you issue the **`vet`** command, a message should be displayed indicating that the system sizer is running.[7]

```
Running system sizer on node chaaly . . . please wait
```

Should you see a different message beforehand, one indicating that a discrepancy exists between the expectant operating system's version number and the existing operating system's version number, then you'll need to follow the instructions that are displayed. These instructions will inform you of Q-VET's current version number, along with the expected operating system's version number that is known to be compatible with that version of Q-VET. It will also inform you of the existing operating system's version

[7]The system sizing process finds the devices that are connected to a system and obtains device information needed by Q-VET programs.

number and indicate that you should update either the version of Q-VET, or the version of the operating system, so that a known compatible pair will be run together.

After the sizer completes, you *may* also see another informational message indicating that the sizer process could not access all devices on your system.

```
Sizer was unable to open all files in /dev.
System configuration information may not be complete.
```

You can ignore this message: it indicates only that you do not have the proper privileges to access some devices on your system. You can still run Q-VETand test the devices that you *can* access.

Q-VET's user prompt is displayed:

```
Q-VET_setup>
```

To view a list of the devices that were identified by the system sizer, issue the **`show devices all`** command.

```
Q-VET_setup> show devices all
 SYSTEM  DEVICE                TYPE      SUBTYPE     SIZE      STATUS
 chaaly  memory                V_MEM               67100672
 chaaly  file                  FILE_DATA
 chaaly  cpu                   CPU       risc
 chaaly  network               NETWORK   tcpdec
 chaaly  /dev/rmt0l            TAPE      tk70                  LOCAL_RW
 chaaly  /dev/nrmt0l           TAPE      tk70                  LOCAL_RW
 chaaly  /dev/rmt0h            TAPE      tk70                  LOCAL_RW
 chaaly  /dev/nrmt0h           TAPE      tk70                  LOCAL_RW
 chaaly  terminal              TERMINAL
 chaaly  printer               PRINTER
 chaaly  video                 VIDEO
Q-VET_setup>
```

Learning to Use Q-VET

The following sections outline how to perform basic operations in Q-VET. Each section first describes how to perform the operation in general. At the end of each section is an example which you can try to perform using the information given in the beginning of the section. The examples start with “For this example...”. Each example depends on all previous examples being performed first.

Viewing Process Definitions

To view process definitions, issue the **show process** command.

```
Q-VET_setup> show process all

 Process 1, group exer for device cpu
   Status is NOT LOADED
   Requested runtime:          0 hours   0 minutes   0 seconds
   Elapsed runtime:            0 hours   0 minutes   0 seconds
   Remaining runtime:          0 hours   0 minutes   0 seconds

   Requested passcount: 1
   Completed passcount: 0

   Options:
      error_check_level                   : 3
      cpu_affinity (id # or "NONE")       : none
   Error Threshold: 0

 Process 2, group exer for device memory
   Status is NOT LOADED
   Requested runtime:          0 hours    0 minutes   0 seconds
   Elapsed runtime:            0 hours    0 minutes   0 seconds
   Remaining runtime:          0 hours    0 minutes   0 seconds

   Requested passcount: 1
   Completed passcount: 0

   Options:

      memory_to_allocate                   : <defined at runtime>
      min_segment_size                     : <defined at runtime>
      max_segment_size                     : <defined at runtime>
      pattern                              : 0
      enable_writes                        : yes
      error_check_level                    : 3
      cpu_affinity (id # or "NONE")        : none

   Error Threshold: 0

   Process 3, group exer for device network
   Status is NOT LOADED
   Requested runtime:         0 hours     0 minutes   0 seconds
   Elapsed runtime:           0 hours     0 minutes   0 seconds
   Remaining runtime:         0 hours     0 minutes   0 seconds
```

```
Requested passcount: 1
Completed passcount: 0

Options:
   transport (1=TCP 2=UDP 3=DECnet)     : 1
   interface (1=sockets 2=XTI)          : 1
   nodes_list                           : <defined at runtime>
   single_step (TCP/DECnet)             : no
   packet_size                          : 1024
   total_bytes (per connect)            : 0
   total_writes (UDP only)              : 1
   packets (total targets)              : 5000
   pattern                              : 0
   delay                                : 0
   error_check_level                    : 3
   cpu_affinity (id # or "NONE")        : none

Error Threshold: 0

Process 4, group exer for device file
Status is NOT LOADED
Requested runtime:        0 hours     0 minutes   0 seconds
Elapsed runtime:          0 hours     0 minutes   0 seconds
Remaining runtime:        0 hours     0 minutes   0 seconds

Requested passcount: 1
Completed passcount: 0

Options:
   file_name                            : <defined at runtime>
   enable_writes                        : yes
   reads_per_iteration                  : 1
   block_size                           : 512
   start_block                          : 0
   end_block                            : 499
   step                                 : 0
   iterations                           : 1000
   delay                                : 0
   pattern                              : 0
   error_check_level                    : 3
   read_only_verify                     : no
   key                                  : <defined at runtime>
   save_file                            : no
   cpu_affinity (id # or "NONE")        : none

Error Threshold: 0
```

Starting a Q-VET Run

To start a Q-VET run, issue the **start** command. In this example, issuing the command starts the run that you set up when you issued the **load** command.

Q-VET displays messages describing events that occur during the run. The messages indicate when each process starts and stops, display testing error reports, and indicate when the run is complete.

Wait for the completion of the run that you have started before going on to Viewing a Run Summary.

Viewing a Run Summary

To view a run summary during or after a run, issue the **show summary** command. The summary displays the state of the Q-VET run at the time you issued the command.

Deselecting Processes

Deselecting a process removes it from the Q-VET run. To deselect processes, issue the **deselect processes** command, including the numbers of the processes you wish to deselect, for example:

```
deselect processes 1 3
```

For this example, deselect all processes created by the **load** command.

```
Q-VET_setup> deselect processes all
```

Selecting Devices for a Q-VET Run

To view the list of available devices, issue the **show devices all** command.

To select devices for testing, issue the **select devices** command.

For this example, select the cpu and memory devices.

```
Q-VET_setup> show devices all

SYSTEM  DEVICE                 TYPE      SUBTYPE     SIZE      STATUS
 chaaly  memory                V_MEM               67100672
 chaaly  file                  FILE_DATA
 chaaly  cpu                   CPU       risc
 chaaly  network               NETWORK   tcpdec
 chaaly  /dev/rmt0l            TAPE      tk70                  LOCAL_RW
 chaaly  /dev/nrmt0l           TAPE      tk70                  LOCAL_RW
 chaaly  /dev/rmt0h            TAPE      tk70                  LOCAL_RW
 chaaly  /dev/nrmt0h           TAPE      tk70                  LOCAL_RW
 chaaly  terminal              TERMINAL
 chaaly  printer               PRINTER
 chaaly  video                 VIDEO

Q-VET_setup> select devices cpu memory
Process 10, group exer for device cpu
Process 11, group exer for device memory

Q-VET_setup>
```

Setting the Run Time for All Processes

To view the current run time settings, issue the **show runtime command**.[8]

To set the run time for all processes, issue the **set runtime** command. Do not include device names or process numbers in the command.

For this example, set the run time to 3 minutes.

```
Q-VET_setup> set runtime 3

Q-VET_setup> show runtime

Process 10 Runtime:
    Requested runtime: 0 hours 3 minutes 0 seconds
    Elapsed runtime: 0 hours 0 minutes 0 seconds
    Remaining runtime: 0 hours 3 minutes 0 seconds

Process 11 Runtime:
    Requested runtime: 0 hours 3 minutes 0 seconds
    Elapsed runtime: 0 hours 0 minutes 0 seconds
    Remaining runtime: 0 hours 3 minutes 0 seconds

Q-VET_setup>
```

Setting the Pass Count for All Processes

To view the current pass count, issue the **show passcount** command.

[8]A process's run time is the total time for which it is allowed to execute.

To set the pass count for all processes, issue the **set passcount** command. Do not include device names or process numbers in the command.

For this example, set the pass count to 0. This causes Q-VET to ignore the pass count and use only the run-time value to end the run.

```
Q-VET_setup> set passcount 0

Q-VET_setup> show passcount
Process 10 Passcount:
    Requested passcount: 0
    Completed passcount: 0

Process 11 Passcount:
    Requested passcount: 0
    Completed passcount: 0

Q-VET_setup>
```

Setting the CPU Affinity for All Processes

To view the current cpu affinity id, issue the **show affinity** command.

To set the cpu affinity for all processes, issue the **set affinity** command.

Setting the Error Threshold for All Processes

To view the current error_threshold value, issue the **show error_threshold** command.

To set the error threshold for all processes, issue the **set error_threshold** command.

For this example, set the error threshold value to 5. This causes Q-VET to halt when a total of 5 errors occurs from any combination of processes.

```
Q-VET_setup> set error_threshold 5

Q-VET_setup> show error_threshold

Process 10 Error Threshold: 5

Process 11 Error Threshold: 5

Process 12 Error Threshold: 5

Q-VET_setup>
```

Duplicating a Process

To make one or more copies of an existing process definition, issue the **duplicate process** command.

For this example, make two additional copies of the cpu process.

```
Q-VET_setup> duplicate process 10 2
Process 12, group exer for device cpu
Process 13, group exer for device cpu
```

Starting, Suspending, Continuing, and Terminating Processes

To start this run, issue the **start** command. . .

```
Q-VET_setup> start
...starting [Process 10, Pass 1] group exer for device cpu.
...starting [Process 11, Pass 1] group exer for device memory.
...starting [Process 12, Pass 1] group exer for device cpu.
...starting [Process 13, Pass 1] group exer for device cpu.
```

Wait approximately 30 seconds. Issue the **stop** command. All processes temporarily suspend executing and enter the suspend execution state.

```
Q-VET_active> stop
...stopping [Process 13] group exer for device cpu.
...stopping [Process 10] group exer for device cpu.
...stopping [Process 12] group exer for device cpu.
...stopping [Process 11] group exer for device memory.
Q-VET_suspend>
```

Issue the **continue** command. All processes resume executing and enter the active execution state.

```
Q-VET_suspend> continue
...continuing [process 10] group exer for device cpu.
...continuing [process 11] group exer for device memory.
...continuing [process 12] group exer for device cpu.
...continuing [process 13] group exer for device cpu.
```

Issue the **stop** command again.

```
Q-VET_active> stop
...stopping [Process 10] group exer for device cpu.
...stopping [Process 13] group exer for device cpu.
...stopping [Process 12] group exer for device cpu.
```

```
...stopping [Process 11] group exer for device memory.
Q-VET_suspend> )
```

Issue the **terminate** command. All processes stop executing permanently; their status is idle.

```
Q-VET_suspend> terminate
...terminated [process 10] group exer for device cpu.
...terminated [process 11] group exer for device memory.
...terminated [process 12] group exer for device cpu.
...terminated [process 13] group exer for device cpu.
Q-VET_setup>
```

To suspend, continue, or terminate individual processes, issue the **stop processes**, **continue processes**, and **terminate processes** commands.

For this example, enter **start** again. Then stop two processes.

```
Q-VET_setup> start
...starting [Process 10, Pass 1] group exer for device cpu.
...starting [Process 11, Pass 1] group exer for device memory.
...starting [Process 12, Pass 1] group exer for device cpu.
...starting [Process 13, Pass 1] group exer for device cpu.

Q-VET_active> stop processes 10 11
...stopping [Process 11] group exer for device memory.
...stopping [Process 10] group exer for device cpu.
```

Suspend all processes.

```
Q-VET_suspend> stop
...stopping [Process 13] group exer for device cpu.
...stopping [Process 12] group exer for device cpu.
```

Continue one process. Then continue the remaining processes.

```
Q-VET_suspend> continue process 12
...continuing [process 12] group exer for device cpu.

Q-VET_active> continue
...continuing [process 10] group exer for device cpu.
...continuing [process 11] group exer for device memory.
...continuing [process 13] group exer for device cpu.
```

Suspend the processes again and terminate them.

```
Q-VET_active> stop
...stopping [Process 12] group exer for device cpu.
```

```
...stopping [Process 13] group exer for device cpu.
...stopping [Process 11] group exer for device memory.
```

```
Q-VET_suspend> terminate
...terminated [process 10] group exer for device cpu.
...terminated [process 11] group exer for device memory.
...terminated [process 12] group exer for device cpu.
```

Modifying Other Options

To modify several Q-VET run-time options, issue the **enable/disable** or **set** commands.

For this example, enable the stopping on hard and fatal errors.

```
Q-VET_setup> enable stop hard fatal
```

Adding and Dropping Processes

You can add processes to a Q-VET run or drop processes from it. This allows you to increase or decrease the system load gradually.

To add or drop processes, issue the **add** and **drop** commands.

All processes defined *before* you issue the **start** command are automatically added to the run. If you define additional processes *during* a Q-VET run, these processes are automatically dropped. You must add them before they will begin executing.

For this example:

1. Drop all processes from your local node.

```
Q-VET_setup> drop process all
```

2. Add one process to your local node.

```
Q-VET_setup> add process 10
```

3. Start the run.

```
Q-VET_setup> start
 ...starting [Process 10, Pass 1] group exer for device cpu.
 ...starting [Process 1, Pass 1] group exer for device cpu.
```

```
...starting [Process 2, Pass 1] group exer for device memory.
*** End of message ***
```

4. Gradually add the remaining local processes one at a time.

```
Q-VET_active> add process 11
...starting [Process 11, Pass 1] group exer for device memory.
Q-VET_active> add process 12
...starting [Process 12, Pass 1] group exer for device cpu.
Q-VET_active> add process 13
...starting [Process 13, Pass 1] group exer for device cpu.
```

5. Wait for the run to complete.

```
*** From node poges ***
...completed [process 1] group exer for device cpu.
*** End of message ***
...completed [process 10] group exer for device cpu.
...completed [process 12] group exer for device cpu.
...completed [process 13] group exer for device cpu.
...completed [process 2] group exer for device memory.

...testing completed. Total errors reported by all processes = 0.
*** End of message ***
...completed [process 11] group exer for device memory.

...testing completed. Total errors reported by all processes = 0.
Q-VET_setup>
```

Modifying Process Parameters

You can modify a process definition's parameters by:

- Setting a run time and/or pass count that applies *only to this process.*
- Selecting or deselecting specific tests within the selected test group.
- Modifying process options.

For this example:

1. Issue the **show process** command to view one of your process definitions.[9]

```
Q-VET_setup> show process 11
Process 11, group exer for device memory
    Status is IDLE
    Requested runtime: 0 hours 3 minutes 0 seconds
    Elapsed runtime: 0 hours 3 minutes 0 seconds
    Remaining runtime: 0 hours 0 minutes 0 seconds

    Requested passcount: 1
    Completed passcount: 1

    Options:
      memory_to_allocate : <defined at runtime>
      min_segment_size : <defined at runtime>
      max_segment_size : <defined at runtime>
      pattern : 0
      enable_writes : yes
      error_check_level : 3
      cpu_affinity (id # or "NONE") : none

    Error Threshold: 0
```

2. Set a run time for this process.

```
Q-VET_setup> set runtime 0:0:20 for 11
```

3. Change the options for this process.

```
Q-VET_setup> select option enable_writes no for 11
Q-VET_setup> select option pattern 3 for 11
```

4. View the process definition again.

```
Q-VET_setup> show process 11
Process 11, group exer for device memory
```

[9]A process definition is a description of a process that is created during a Q-VET run. The process definition contains all process-specific run-time parameters that the process uses when it executes.

```
Status is IDLE
Requested runtime: 0 hours 0 minutes 20 seconds
Elapsed runtime: 0 hours 3 minutes 0 seconds
Remaining runtime: 0 hours 0 minutes 20 seconds

Requested passcount: 0
Completed passcount: 1

Options:
  memory_to_allocate : <defined at runtime>
  min_segment_size : <defined at runtime>
  max_segment_size : <defined at runtime>
  pattern : 3
  enable_writes : no
  error_check_level : 3
  cpu_affinity (id # or "NONE") : none

Error Threshold: 0
```

Editing and Executing Command Scripts

You can edit and execute command scripts with the **edit** and **execute** commands. A command script is a file containing commands that Q-VET can execute.[10]

Use the script buffer to create and execute temporary scripts. Use script files to save your scripts permanently. Always use lower case letters in the script buffer.

To edit the script buffer, issue the **edit** command without a filename. This invokes the editor defined by your EDITOR environment variable. Use this editor to enter Q-VET commands. When done, exit from the editor.

To execute commands in the script buffer, issue the **execute** command.

To save the script buffer contents in a permanent file, issue the **save** command.

To create or edit a permanent script file, issue the **edit** command followed by a file name. This invokes the editor defined by your EDITOR environment variable.

To execute the commands contained in a permanent script file, issue the **execute** command followed by the file's name.

[10]Script Files discusses creating script files to run under Q-VET.

For this example:

1. Edit the script buffer.

```
Q-VET_setup> edit
```

2. Place the following commands in the buffer:

```
deselect processes all
select group exer for cpu
set runtime 0:0:30
start
wait
show summary
```

3. Exit from the editor.
4. Execute the script buffer contents.

```
Q-VET_setup> execute
```

5. Save the script buffer contents in a permanent file.

```
Q-VET_setup> save my_file.script
```

6. Open the file you just created so that you can edit it.

```
Q-VET_setup> edit my_file.script
```

7. Change the file's second line from:

```
select group exer for cpu
```

to

```
select group exer for memory
```

8. Execute the file you have just modified.

```
Q-VET_setup> execute myfile.script
```

Terminating a Q-VET Session

To end a Q-VET session, issue the **exit** or **quit** command.

```
Q-VET_setup> exit
```

Menu Entries and Command Equivalents

Table 4-1 lists the Q-VET menu entries and their command equivalents.

Table 4-1 Q-VET Menu Entries and Command Equivalents

Menu Entry	Function	Equivalent
	File Menu	
Set Path	Displays the current Q-VET path list.	**show path**
	Modifies the path list.	**set path**
Write Error Log	Writes a text string to the operating system error log file.	**write errorlog**
Foreign	Selects for execution a program that was not designed specifically to run under Q-VET.	**foreign**
Exit	Terminates the Q-VET session on all nodes and returns control to the operating system.	**exit** or **quit**
	Setup Menu	
Connect Node	Connects a node.	**connect nodes**
Connect Device	Adds a device's definition to the configuration database.	**connect devices**
Disconnect	Removes a node or device from the configuration database.	**disconnect devices** **disconnect nodes**
Duplicate	Creates one or more duplicate copies of processes.	**duplicate process**
Deselect	Deselects a process or processes.	**deselect processes**
Drop	Drops processes from the current run.	**drop processes**
Add	Adds processes to the current run.	**add processes**

Menu Entry	Function	Equivalent
Change Group	Changes the group for a process.	**select groups**
Show/Modify Parameters	Allows you to examine and change the constituent tests, run time, pass count, error threshold, cpu affinity, and other options of a selected process.	**select options** **select tests** **set passcount** **set runtime** **set error_threshold** **set affinity**
Load	Creates a default system load.	**load**
Save Setup	Saves the contents of Q-VET's Device Work and Process Work Areas and the settings of Q-VET's process options.	no equivalent
Restore Setup	Restores a Q-VET setup state that you have saved with Save Setup.	no equivalent
Clear Setup	Clears the contents of the Device Work Area and the Process Work Area. All process definitions are removed.	**deselect devices all** **deselect processes all** **disconnect nodes all**
	Windows Menu	
Run Display	Displays output of current run.	Standard Q-VET output after **sta** issued
Summary	Displays a summary of the current or most recent Q-VET run.	**show summary**
Groups	Displays information about the test groups installed in the Q-VET database.	**show groups all**
System Error Log	Displays the contents of the operating system error log file.	**display errorlog**
Command Line	Lets you enter Q-VET commands directly.	Q-VET prompt in command line
	Run Menu	
Start All	Starts a q-vet run.	**start**
Stop All	Temporarily stops the execution of Q-VET processes.	**stop** **[Ctrl/C]**
Continue All	Resumes execution of suspended processes.	**continue**

Menu Entry	Function	Equivalent
Terminate All	Permanently terminates suspended processes.	**terminate**
Stop	Temporarily stops the execution of specified processes.	**stop processes** **stop devices**
Continue	Resumes execution of specified processes.	**continue processes** **continue devices**
Terminate	Permanently terminates specified processes.	**terminate processes** **terminate devices**
	Scripts Menu	
Execute Script Buffer	Executes the contents of the script buffer.	**execute**
Execute File	Executes the contents of a script file.	**execute *filename*.**
Edit Script Buffer	Opens the script buffer for editing.	**edit**
Edit File	Opens a new or existing script file for editing.	**edit *filename***
	Options Menu	
Set Runtime	Sets the run time of all Q-VET processes.	**set runtime**
Set Passcount	Sets the pass count for all Q-VET processes.	**set passcount**
Set Timeout	Sets the time interval Q-VET uses to determine if processes are "hung."	**enable timeout** **set timeout**
Set Affinity	Sets the cpu affinity id for all Q-VET processes.	**set affinity**
Set Error_Threshold	Sets the error threshold for a Q-VET run.	**set error_threshold**
Other	Sets parameters that control a Q-VET run.	
The following options, displayed in the Other Options dialog box, are included for complet		
Error Display	Enables a level of error-reporting detail.	**enable logging error**
Trace Display	Enables a type of trace display.	**enable trace**
Stop on...	Stops on an error type.	**enable stop**

Menu Entry	Function	Equivalent
Loop on...	Creates an execution loop on an error type.	**enable loop**
Mode	Selects any combination of run-time modes.	**enable mode**
Execution	Controls execution type---serial or parallel.	**set execution**
Error Logging	Enables or disables Q-VET’s writing messages to the system's error log, during execution of Q-VET runs.	**enable/disable logging error**
	Maintenance Menu	
Install/Remove Files...	Displays the contents of Q-VET’s program database.	**show installed**
	Allows you to add new files to the database and delete existing ones.	**install** **remove**
Set Default Group...	Designates a test group as the default of a device type.	**select default**
Accept...	Runs acceptance procedures for new systems and devices.	**accept**
	Main Window Areas	
Device Work Area	Displays connected nodes and devices.	**show devices**
Process Work Area	Displays process definitions.	**show process**

Commands and Arguments

This section discusses Q-VET commands and covers the following topics:

Entering Q-VET Commands

Script Files

Frequently Used Arguments

Execution States

Test Organization

Run-Time Modes

The section in this chapter, Command Descriptions, describes the format and syntax of each command.

Entering Q-VET Commands

Each Q-VET command consists of a command and, optionally, a keyword or qualifier and one or more arguments. (Frequently Used Arguments discusses frequently used arguments and their form.)

As with other commands, you must terminate each Q-VET command by pressing **[Return]**.

Remember the following when you enter Q-VET commands, keywords, and arguments:

- You can enter Q-VET commands, keywords, and arguments in uppercase or lowercase characters, or both.
- If the argument is a list, use spaces to separate the items, for example:

  ```
  Q-VET_setup> select devices cpu memory
  ```

- You indicate a range of numbers in an argument by using a hyphen (-) to separate the first and last numbers of a series, for example:

  ```
  Q-VET_setup> show process 1 2 4-6 9
  ```

- You can abbreviate a Q-VET command to the minimum number of letters Q-VET needs to distinguish it from any other command.
- You can enter a comment by beginning a command line with an exclamation point (!) or a pound sign (#). See the discussion of the **comment-line character** later in this chapter.
- You can enter a continuation character to continue a command on the next line. The operating system under which Q-VET runs determines the continuation character that is used. See the discussion of the **continuation character** later in this chapter.

Script Files

Scripts can contain any valid series of Q-VET commands, responses to user prompts, and comments. Q-VET executes the commands contained in a script after you issue the **execute** command. Q-VET searches for scripts by following either the default path list or the path list that you have defined with the **set path** command. Issue the **show path** command to find the current directory path list.

Creating Scripts

You can create scripts outside of a Q-VET session by using an editor or you can issue the **edit** command to create scripts during a Q-VET session. The **edit** command invokes an editor of your choice; you choose the editor by defining the EDITOR environment variable before running Q-VET, as in the following example for csh:

```
% setenv EDITOR emacs
```

If you do not define EDITOR, Q-VET invokes the default editor.

Each command or response contained in a script must appear on a new line. You must start a comment line by entering either the **!** or **#** character.

If you issue the **edit** command without specifying a file name, Q-VET places the commands you type in its script buffer. Issue the **execute** command to execute the commands. Issue the **save** command to save the commands contained in the script buffer in a permanent file.

Specifying User Responses in Scripts

Q-VET and Q-VET programs sometimes request responses to user prompts. You place such responses in a script as follows:

1. Precede each response with the **response>** prompt, followed by a space.
2. Place each response on a new line.
3. List the responses in the script in the same order in which Q-VET expects them.

The following is an example of a user response contained in a script:

```
select group exer with options for cpu
! response to "error_check_level" question
response> 2
```

If you do not include expected responses in a script, one of these events may occur:

- If Q-VET is not running in interactive mode, Q-VET takes default responses during the run.
- If Q-VET is running in interactive mode, Q-VET prompts for responses during the run. (Run-Time Modesand the description of the **set** command discuss modes of Q-VET operation.)

This feature lets you create a script that contains all the parameters of a Q-VET run, except the user responses. The user can run the script, customizing the run dynamically by responding to on-the-spot user prompts.

Using the Wait Command

Issue the wait command in a script to control when the next command is read. The wait command without a time interval causes Q-VET to complete the run initiated by the start before processing the next command.[11]

The following is an example of issuing a **wait** command in a script:

```
# Select device rr0c
select device /dev/rr0c
# Set pass count to 15
set passcount 15
# Start run.
start
# Wait for run to complete.
wait
# Display run summary.
show summary
```

Frequently Used Arguments

The following sections discuss the most frequently specified arguments to the Q-VET commands.

Node Names and Lists

Several commands accept the *node-name* and *node-list* arguments. The following general rules apply:

- Node names are network node names.
- In the connect command, the format of the name specifies the transport mechanism used in referencing the node. Append a colon (:) to represent TCP/IP;append two colons (::) to represent DECnet.

  The following command connects anode that supports the TCP/IP mechanism:

  ```
  VET_SETUP> connect nodes micros:
  ```

  Specifying the transport is not necessary in other commands.

[11]In active state (see Execution States) Q-VET immediately fetches the command following the **start** command in a script file. By issuing **wait**, you control when Q-VET reads the next command.

- The *node-list* argument is a list of node names. A blank character (space) must separate items in the list.

Device Names and Lists

Several commands accept the *device-name* and *device-list* arguments. The following general rules apply:

- Device names have a combination of letters and numbers, for example, *rrz1c* and *rmt0h*, and include the path name to the device, such as */dev/rrz1c*.
- Device names can include node names, such as *mynode: /dev/rra1c*, as in the command:

  ```
  Q-VET_setup> select devices testnode:/dev/rrz1c
  ```

  If you enter a device name without a node name, Q-VET assumes that the device is local.
- The *device-list* argument is a list of one or more devices, optionally including the path, node, or both to which a device belongs. A blank character (space) must separate items in the list. If you enter **all** as the *device-list*, Q-VET acts on all devices.

Issue the **show devices all** command to find the names of devices connected to your system.

Path Names

Several commands accept the *path* argument. In addition, a path description can precede a device or file name.

The following general rules apply:

- A device name includes the path name to that device, for example, */dev/rra0c*. (If you enter a device name without a node name, Q-VET assumes that the device is local.)
- A file specification can include the path name to that file, for example, */exer/disk/old/ra_exer*
- Path names can include node names, such as: *mynode:/dev /rra1c*
- If you do not include a path name when specifying a file, Q- searches the directory path search list to locate the file.

Issue the **show path** command to view the current directory path search list.

Group Names and Lists

Several commands accept the *group-name* and *group-list* arguments. The following general rules apply:

- Each group has a name that can consist of letters, numbers, and the underscore character, for example, MEMEXER_2.
- The *group-list* argument is a list of one or more groups. A blank character (space) must separate items in the list.

Issue the **show groups all for all** command to find the names of all test groups.

Test Numbers and Lists

Several commands accept the *test-number* and *test-list* arguments. The following general rules apply:

- Each test in a test group has an identifying number.
- The *test-list* argument is a list or range of test numbers. A hyphen must separate the first and last numbers of a range.
- A blank character (space) must separate items in the list.
- If you enter **all** as the *test-list*, Q-VET acts on all tests.

Issue the **show groups** *group-list* **for** *device-name* command to find the number of each test in the specified groups for the specified devices

Subtest Numbers

Several commands accept the *subtest-number* argument. The following general rules apply:

- Q-VET displays a message every time a process executes a new subtest, if you have previously issued the **enable trace** command. The message indicates the subtest number.
- You can find subtest numbers by reading Q-VET documentation or by referring to the header portion of error reports.
- You can sometimes enter **last_subtest** in place of *subtest-number*. In this case, Q-VET acts on the subtest specified in the **select groups** command as the last subtest.

Process Numbers and Lists

Several commands accept the *process-number* and *process-list* arguments. The following general rules apply:

- Whenever Q-VET creates a process, it assigns the process an identifying number and displays a message indicating the process number.
- The *process-list* argument can be one of the following:
  - A process number
  - A list of numbers
  - A range of numbers, whose first and last numbers are separated by a hyphen
  - A combination of the above
  - The **last** argument, which refers to the most recently referenced process number
  - The **all** argument, which indicates all

A blank character (space) must separate items in the process list.

Issue the **show process all** command to display all processes and their attributes.

Time

Several commands accept the *time* argument.

The *time* argument can take the following forms:

mm

hh:mm

hh:mm:ss

where:

hh is a whole number indicating hours

mm is a whole number indicating minutes

ss is a whole number indicating seconds

If you specify one number without the delimiter (:), Q-VET interprets the argument as minutes. There is no practical limit to the number. If you specify 4:344:97987 as a time, Q-VET interprets the argument as 4 hours, 344 minutes, and 97987 seconds.

Execution States

When you run Q-VET it operates in one of three *execution states*:

- Setup state
- Active state
- Suspend state

Q-VET displays a different prompt for each state.

Table 4-2 lists the commands that you may enter in each execution state.

Setup State

When you activate Q-VET, it enters setup state and displays the setup state prompt:

```
% vet
Running system sizer on node mingo4 ... please wait.
Q-VET_setup>
```

You can execute any user command except the following in the setup state:

continue

stop

terminate

wait

Active State

After you have selected the test groups and devices to run, enter the **start** command to put Q-VET in the active state and begin executing tests:

```
Q-VET_setup> start
Q-VET_active>
```

Suspend State

If you enter the **stop** command or press **[Ctrl/C]** while Q-VET is in the active state, Q-VET suspends testing and enters the suspend state. The suspend state prompt is:

```
Q-VET_suspend>
```

Issue the **continue** command to cause Q-VET to resume executing in active state; issue the **terminate** command or press **[Ctrl/C]** to cause Q-VET to enter the setup state.

Table 4-2 Execution States and Commands

Command	Setup	Active	Suspend
accept	Yes	No	No
add devices	Yes	Yes	Yes
add processes	Yes	Yes	Yes
certify	Yes	No	No
connect	Yes	No	No
continue	No	Yes	Yes
control-c	Yes	Yes	Yes
deselect devices	Yes	No	No
deselect groups	Yes	No	No
deselect processes	Yes	No	No
deselect tests	Yes	No	No
disable error_display	Yes	Yes	Yes
disable logging	Yes	No	No
disable loop	Yes	Yes	Yes
disable mode	Yes	No	No
disable scrolling	Yes	Yes	Yes
disable stop	Yes	Yes	Yes
disable timeout	Yes	Yes	Yes
disable trace	Yes	Yes	Yes
disconnect	Yes	No	No
display errorlog	Yes	Yes	Yes
drop devices	Yes	Yes	Yes
drop processes	Yes	Yes	Yes
duplicate processes	Yes	Yes	Yes
edit	Yes	No	No
enable error_display	Yes	Yes	Yes
enable logging	Yes	No	No

enable loop	Yes	Yes	Yes
enable mode	Yes	No	No
enable scrolling	Yes	Yes	Yes
enable stop	Yes	Yes	Yes
enable timeout	Yes	Yes	Yes
enable trace	Yes	Yes	Yes
execute	Yes	No	No
exit	Yes	No	Yes
foreign	Yes	No	No
help	Yes	Yes	Yes
install	Yes	No	No
load	Yes	No	No
quit	Yes	No	Yes
remove	Yes	No	No
save	Yes	No	No
select default	Yes	No	No
select devices	Yes	Yes	Yes
select groups	Yes	Yes	Yes
select options	Yes	Yes	Yes
select tests	Yes	Yes	Yes
set execution	Yes	No	No
set affinity	Yes	No	No
set error_threshold	Yes	Yes	Yes
set logfile	Yes	No	No
set passcount	Yes	Yes	Yes
set path	Yes	No	No
set runtime	Yes	Yes	Yes
set screen_size	Yes	Yes	Yes
set timeout	Yes	Yes	Yes
show devices	Yes	Yes	Yes
show affinity	Yes	Yes	Yes
show error_display	Yes	Yes	Yes
show error_threshold	Yes	Yes	Yes
show execution	Yes	Yes	Yes
show groups	Yes	Yes	Yes
show installed	Yes	Yes	Yes
show logging	Yes	Yes	Yes

show loop	Yes	Yes	Yes
show mode	Yes	Yes	Yes
show nodes	Yes	Yes	Yes
show passcount	Yes	Yes	Yes
show path	Yes	Yes	Yes
show process	Yes	Yes	Yes
show runtime	Yes	Yes	Yes
show screen_size	Yes	Yes	Yes
show scrolling	Yes	Yes	Yes
show selected devices	Yes	Yes	Yes
show selected groups	Yes	Yes	Yes
show stop	Yes	Yes	Yes
show summary	Yes	Yes	Yes
show timeout	Yes	Yes	Yes
show trace	Yes	Yes	Yes
show version	Yes	Yes	Yes
start	Yes	No	No
stop	No	Yes	No
system	Yes	Yes	Yes
terminate devices	No	No	Yes
terminate processes	No	No	Yes
wait	No	Yes	No
write errorlog	Yes	Yes	Yes

Test Organization

Q-VET tests are organized into the following categories:

Test Groups

Tests

Subtests

Test Groups

A Q-VET *test group* is the primary Q-VET testing entity; it consists of one or more tests. Each test group provides a complete testing function; for example, a group called DEVICE_EXERCISER might test each I/O function on disk drives.

The **select groups** command selects a group for execution. Generally, users want to run all the tests associated with the selected group. However, you can use the **select/deselect tests** command to include or exclude specific tests belonging to the selected group.

When a Q-VET program is added to the program database, you can specify a default test group to associate witha device. Q-VET runs this group by default, if you do not select another group for execution before issuing the **start** command.

Tests

A Q-VET *test* is a set of one or more device-testing procedures (*subtests*) that have been grouped together; for example, the group DEVICE_EXERCISERthat tests RA-type disks might consist of two tests: a read-only test and a read/write test.

When you issue the **select groups** command to select a test group for execution, you can specify arguments to include or exclude tests from the group.You can also use the **select tests** command to specify tests.

Each test in a test group is independent of the others. Unlike a subtest its execution is never contingent upon the outcome of another test.

Subtests

A *subtest* is a set of one or more device-testing procedures. All subtests are written to be members of a test. Subtests are not necessarily independent of each other; one subtest might depend on the execution of the previous subtest. You cannot select specific subtests for execution.

There is no limit to the number of subtests in each test. Subtests are numbered from one to the maximum number of subtests in the test of which they are a part.

Subtests are the parts of a test on which Q-VET can *loop* when it encounters an error, that is, repeatedly call the same subtest.

You use the **enable loop** command to direct Q-VET to loop when it finds an error. In addition, you can use the **select groups last_subtest** command to specify that a test execute all subtests up to and including a particular subtest and then loop on that subtest.

Run-Time Modes

Q-VET provides five user-selectable run-time modes. These modes are not related to each other. You can use the **enable mode** command to activate any combination of them.

The modes are as follows:

- Automated---This mode is enabled when Q-VET is running under the control of a higher-level automated environment, such as an automated testing system used by manufacturing. When Q-VET operates in this mode, process startup, pass count, and completion messages are suppressed.

  This mode is disabled by default.
- Interactive---This mode is enabled when a user is present to respond to queries displayed on a terminal. When Q-VET operates in this mode, it sends its requests to the user's terminal and fetches input from either the user or a script file.

  This mode is enabled by default; however, if it is disabled, Q-VET does not prompt for user responses and instead takes default values.
- Quick---This mode is a shortened mode of operation.

  Some Q-VET programs provide a quick mode of operation. Use the **enable mode quick** command to activate them. The documentation of Q-VET programs tells you whether they provide quick mode operation.

This mode is disabled by default.

- Truncate---When this mode is enabled and a Q-VET process reports a device error, subsequent passes of that process execute only up to the subtest that preceded the failing one; then Q-VET starts the next pass.

  You use this mode as an aid in isolating intermittent errors; you use it to identify the minimum hardware activity needed to cause the error.

  This mode is disabled by default.

- Verify---This mode causes script file input lines to be displayed as Q-VET reads them from the file. (Note - not all tests support this mode.)

  This mode is disabled by default.

You use the **enable mode** command to select the run-time mode or modes under which you want Q-VET to operate. You use the **disable mode** command to deselect the run-time modes.

Command Descriptions

This section is an alphabetical listing of all Q-VET commands. Each command description contains the following elements, as needed:

- Description---A brief description of the command's function
- Format---The syntax for using the command
- Keywords and arguments---A list of keywords and arguments that can modify the command and a description of each one's effect
- Restrictions---A list of factors that affect the use of the command
- Examples---An example of how to use the command

accept

Description

Executes a Q-VET run that is part of Digital Equipment Corporation's customer acceptance procedure for new systems and devices.

After you have issued the **accept** command, Q-VET prompts for a DIGITAL order number before it begins acceptance testing. Q-VET tests devices for a minimum of 10 minutes.

Format

accept { none }
{ **[selected] [for runtime *time*] [with output *file-spec*]** }

Keywords and Arguments

none

Q-VET tests all devices connected to the local node that have default test groups assigned to them.

selected

Run the acceptance procedure on previously selected devices.

Before issuing the **accept** command, issue the **select devices** command to select the device or devices to test. The default test group assigned to each device tests the device. You can choose to run another group by issuing the **select groups** *group-name* **for** *device-name* command or you can choose another test group to be the default for the device by issuing the **select default** *group-name* **for** *device-name* command.

for runtime *time*

Run the acceptance procedure for the specified amount of time, if this amount is greater than the default minimum amount. There is no maximum time limit.

The section Time describes the *time* argument.

with output *file-spec*

Create and print an output file (*file-spec*) containing both test start and end times and a place for signatures. The following example shows an **accept** command output file.

```
COPYRIGHT DIGITAL EQUIPMENT CORPORATION 1993 ALL RIGHTS RESERVED
```

```
     Verification Acceptance Process for DEC order number 123-456-7890
     started on Wed Feb 24 10:02:13 1993


     Acceptance tested system VETSYS for 10 minutes.

________________________________.
________________________________
Completed acceptance on Wed Feb 24 10.12.58 1993
```

Restrictions

- You can issue the **accept** command only while is operating in the setup execution state.
- You cannot run the **accept** command distributively on remote nodes.
- Devices to test must be available and correctly configured.
- You cannot issue commands in active mode, while acceptance testing is in progress.

Example

```
Q-VET_setup> **select devices file cpu memory**

Process 4, group exer for device file
Process 5, group exer for device cpu
Process 6, group exer for device memory

Q-VET_setup> **show selected devices**
 file
 cpu
 memory

Q-VET_setup> **accept selected for runtime 00:11:00 with output vettest.txt**

 This is the Digital System Acceptance routine.
 Please type DEC order number to continue or <CR> to exit routine: (string [
]):
 123-456-7890 [Return]

 These devices which were previously selected will be tested.

 vetsys FILE                  FILE_DATA                       LOCAL R_W
 vetsys CPU                   CPU         DEC 3000 Model 500
 vetsys MEMORY                V_MEM               16777216    LOCAL R_W

 COPYRIGHT DIGITAL EQUIPMENT CORPORATION 1993 ALL RIGHTS RESERVED

     Verification Acceptance Process for DEC order number 123-456-7890

    started on Wed Feb 24 10:02:13 1993
 ...starting [Process 1, Pass 1] group exer for device file
 ...starting [Process 3, Pass 1] group exer for device cpu
 ...starting [Process 4, Pass 1] group exer for device memory
 ...runtime has expired for [Process 1] group exer for device file
 ...runtime has expired for [Process 3] group exer for device cpu
 ...runtime has expired for [Process 4] group exer for device memory
 ...completed [Process 3] group exer for device cpu
 ...completed [Process 4] group exer for device memory
 ...completed [Process 1] group exer for device file

 ...testing completed. Total errors = 0

 The signoff form for this run is in the file vettest.txt.
 Completed acceptance on Wed Feb 24 10:13:28 1993

Q-VET_setup> **exit**
```

add

Description

Adds processes or devices to the Q-VET run. Both the **add** and **drop** commands let you alter the test load dynamically.

If you issue the **select groups**, **select devices**, or **duplicate process** command to select processes *before* you issue the **start** command, Q-VET automatically adds the selected processes to the run.

If you issue the **select groups**, **select devices**, or **duplicate process** command to select processes *after* you have issued the **start** command, you must issue the **add** command to add them to the run; otherwise, they will not execute.

Use the **drop** command to remove processes from the run.

Format

add { **devices** ***device-list*** **[on *node-list*]** }
{ **processes** ***process-list*** **[on *node-list*]** }

Keywords and Arguments

devices device-list [on node-list]

Add all selected processes for the devices that you specify in *device-list*. Device Names and Lists describes the *device-list* argument.

The optional phrase **on** *node-list* specifies the node(s) on which this command takes effect. If you omit this phrase, the local node is the default. Node Names and Lists describes the *node-list* argument.

processes process-list [on node-list]

Add the processes specified in *process-list* to the current run. Process Numbers and Lists describes the *process-list* argument.

The optional phrase **on** *node-list* specifies the node(s) on which this command takes effect. If you omit this phrase,the local node is the default. Node Names and Lists describes the *node-list* argument. None concerning execution state: you can issue this command while Q-VET is operating in the setup, active, or suspend execution state.

Restrictions

- None concerning execution state: you can issue this command while Q-VET is operating in the setup, active, or suspend execution state.

- Processes selected after you have issued the **start** command will not execute until you add them by using the **add** command.

If you use the on *node-list* phrase, you cannot include node names in a device name or device list.

Example

```
Q-VET_setup> select devices file
Process 1, group exer for device file
Q-VET_setup> duplicate processes 1 3
Process 2, group exer for device file
Process 3, group exer for device file
Process 4, group exer for device file
Q-VET_setup> drop processes 2 3 4
Q-VET_setup> start .
   .
   .
   .
Q-VET_active> add processes 2 3 .
   .
   .
   .
Q-VET_active> add processes 4
```

certify

Description

Performs simple quality assurance testing of programs written to execute under Q-VET All programs selected for a Q-VET run are tested.

You issue the **certify** command instead of the **start** command.

Quality assurance testing of a Q-VET program includes the display of all error reports, even i.f the corresponding failure does not occur.

Format

```
certify
```

Restrictions

- You can issue the certify command only while Q-VET is operating in the setup execution state.
- You must have selected a device or group before issuing the certify command.
- The certify command is valid only on the local node.

Example

```
Q-VET_setup> select devices cpu
Q-VET_setup> select groups exer for file
Q-VET_setup> certify
```

comment-line character

Description

Begins a comment line. You can use either the **!** or the **#** character.

Q-VET treats a command line whose first nonblank character is either **#** or **!** as a comment line and does not try to execute its contents.

You use the comment-line character when you document script files.If you have used the **enable logging terminal** command to enable logging, you can enter a comment line at your terminal, and Q-VET records it in the log file.

Format

{ **!** ***Line of comments*** }
{ **#** ***Line of comments*** }

Example

```
Q-VET_setup> ! adding device  /dev/rra1c to increase load
Q-VET_setup> add devices /dev/rra1c
```

connect

Description

Adds hardware devices to the *configuration database*, that is, to the database of testable devices. Q-VET cannot test a device, unless the configuration database contains its description.

When you activate Q-VET it runs a *system sizing process* on the local system and fills in the configuration database. You issue the **connect devices** command to define a device that the system sizing process has not previously recognized. You issue the **connect nodes** command to run the system sizing process on a remote node(s), place the corresponding device definitions in the configuration database, and establish communication with it.

You issue the **show devices all** command to view the contents of the configuration database.

Note

Q-VET does not verify that the device you added with the **connect** command is available, until it executes the corresponding exerciser.

Format

```
Connect { nodes [nosizer] [affinity cpu-id] node-list }
        { node local }
        { node  node name using transport type }
        { devices [on node-list] }
```

Keywords and Arguments

nodes [nosizer] [affinity cpu-id] *node-list*

Add devices existing on remote systems to Q-VET's configuration database. The optional **nosizer** keyword disables the system sizing process on the remote node. The optional **affinity** keyword may be included to specify a specific processor to run the remote Q-VET processes on. If processor affinity is not supported by the remote node or an invalid cpu-id is specified the conection will be terminated. Node Names and Lists describes the *node-list* argument.

node local

Add devices to the configuration database for the local system, that is, the system on which Q-VET is executing. Q-VET does this automatically when you start a Q-VET session.

You can use this command to rerun the system sizing process on the local node.

node *node name* using *transport type*

Add devices to the configuration database for the remote node specified by the node name. The transport argument indicates which transport mechanism is to be used when a connection is established with the node specified. The two valid transport mechanisms are decnet and tcpip.

devices *device-list* [on *node-list*]

Enter device descriptions into the configuration database. Use this command if you must add a device to the configuration database that is not recognized by the Q-VET sizing process.

After you issue the **connect device** command, Q-VET prompts you for the device characteristics to be entered in the configuration database.

Device Names and Lists describes the *device-list* argument.

The optional phrase **on** *node-list* specifies the node(s) on which this command takes effect. If you omit this phrase, the local node is the default. Node Names and Lists describes the *node-list* argument.

Restrictions

- You can issue the connect command only Q-VET is operating in the setup execution state.
- When Q-VET terminates, the configuration and sizing information are not saved.
- If you use the on node-list phrase, you cannot include node names in a device name or device list.

Example

```
Q-VET_setup> connect devices mynode:/dev/rrz1c
Enter component type (string [MISC]): disk
Enter component subtype (string [UNKNOWN]):  rz56
Enter component size (decimal [0]): <Return>
Enter component status (string [LOCAL_RO]): LOCAL_RW

Q-VET_setup> show devices mynode:/dev/rrz1c

SYSTEM  DEVICE                TYPE      SUBTYPE     SIZE      STATUS
mynode /dev/rrz1c             DISK      rz56                  LOCAL_RW
```

continuation character

Description

Continues a command on the next line. You may use either a backslash (\) or hyphen (-).

Format

```
-
\
```

Restriction

The continuation character works only if it is preceded by a space, as in the example below.

Example

```
Q-VET_setup> select groups exer with options \
reads_per_iteration 5 iterations 100 for /dev/rra0c
```

continue

Description

Resumes execution of a specific suspended process or all suspended processes.

The discussions of the **stop** command and **[Ctrl/C]** explain how to suspend running processes. A process may also be suspended if it detects an error and you have previously issued the **enable stop** command. The discussion of the **enable stop** command describes the Q-VET stop-on-error feature.

Format

```
continue { none }
         { devices device-list [on node-list] }
         { processes process-list [on node-list] }
```

Keywords and Arguments

none

Resume executing all suspended processes on all nodes.

devices *device-list* [on *node-list*]

Resume executing suspended processes associated with the devices specified in *device-list*. Device Names and Lists describes the *device-list* argument.

The optional phrase **on** *node-list* specifies the node(s) on which this command takes effect. If you omit this phrase,the local node is the default. Node Names and Listsdescribes the *node-list* argument.

processes *process-list* [on *node-list*]

Resume executing the suspended processes listed in *process-list*. Process Numbers and Lists describes the *process-list* argument.

The optional phrase **on** *node-list* specifies the node(s) on which this command takes effect. If you omit this phrase,the local node is the default. Node Names and Lists describes the *node-list* argument.

Restrictions

- You cannot issue the continue command Q-VET is operating in the setup execution state.
- If you use the on node-list phrase, you cannot include node names in a device name or device list.

Example

```
Q-VET_setup> select groups exer for network
Process 1, group exer for device network
Q-VET_setup> start
...starting [Process 1, Pass 1] group exer for device network.

Q-VET_active> stop ...stopping [Process 1] group exer for device network.

Q-VET_suspend> continue devices network
...continuing [Process 1] group exer for device network.
```

control-c

Description

Pressing **[Ctrl/C]** does one of the following:

- Terminates a Q-VET session, if Q-VET is operating in the setup execution state, unless
  Q-VET is executing a script file.
- Causes Q-VET to reenter the setup execution state, if it is executing a script file.
- Suspends executing processes and enters the suspend execution state, if Q-VET is operating in the active state
- Causes Q-VET to enter the setup execution state, if Q-VET is operating in the suspend state

Table 4-3 lists the effects of pressing **[Ctrl/C]**.

Table 4-3 Effect of [Ctrl/C]

Original State	Non-Script New State	Script New State
setup	Shell	setup
Active	Suspend	Suspend
Suspend	setup	setup

After you have pressed **[Ctrl/C]** to place Q-VET in the suspend execution state, you can issue the **continue** command to cause Q-VET to reenter the active state.

Format

[CtrlC]

Restriction

When you are running Q-VETin a distributed environment and you press [Ctrl/C], Q-VET enters the suspend execution state if the remote nodes are operating in either the suspend or the setup execution state. In a distributed environment, Q-VET enters the setup execution state, only if all remote nodes are also operating in the setup execution state.

Example

```
Q-VET_setup> select devices /dev/rra1c /dev/rra2c
Process 1, group exer for device /dev/rra1c
Process 2, group exer for device /dev/rra2c
Q-VET_setup> start
   .
   .
   .

Q-VET_active> [Ctrl/C]
...stopping [Process 2] group exer for device /dev/rra2c.
...stopping [Process 1] group exer for device /dev/rra1c.
Q-VET_suspend>
```

control-z

Description

Suspends Q-VET and all processes running under it. Control is returned to the operating system.

Format

[Ctrl/Z]

Example

```
Q-VET_suspend> [Ctrl/Z]
%
```

deselect

Description

Removes a previously selected device, test group, or test; removes one or more processes from the current Q-VET run.

See the discussion of the **select** command for information on how to select devices, groups, tests, and processes for execution.

Format

```
deselect { devices device-list [on node-list] }
         { groups group-list for device-name [on node-list] }
         { tests test-list for process-list [on node-list] }
         { processes process-list [on node-list] }
```

Keywords and Arguments

devices *device-list* [on *node-list*]

Deselect the devices specified in *device-list* and do not test them. Device Names and Lists describes the *device-list* argument.

The optional phrase **on** *node-list* specifies the node(s) on which this command takes effect. If you omit this phrase,the local node is the default. Node Names and Lists describes the *node-list* argument.

groups *group-list* for *device-name* [on *node-list*]

Deselect all test groups that you specify in *group-list* for the device that you specify with *device-name*. Group Names and Lists describes the *group-name* and *group-list* arguments.

The optional phrase **on** *node-list* specifies the node(s) on which this command takes effect. If you omit this phrase, the local node is the default. Node Names and Lists describes the *node-list* argument.

tests *test-list* for *process-list* [on *node-list*]

Deselect the tests that you specify in *test-list* for a particular process. Test Numbers and Lists describes the *test-list* argument. This keyword and argument do not change the state of the tests that you have not listed in *test-list*. Process Numbers and Lists describes the *process-list* argument.

The optional phrase **on** *node-list* specifies the node(s) on which this command takes effect. If you omit this phrase,the local node is the default. Node Names and Lists describes the *node-list* argument.

processes *process-list* [on *node-list*]

Remove the process(es) that you specify in *process-list*. Process Numbers and Lists describes the *process-list* argument.

The optional phrase **on** *node-list* specifies the node(s) on which this command takes effect. If you omit this phrase, the local node is the default. Node Names and Lists describes the *node-list* argument.

Restrictions

- You can issue the **deselect** command only while Q-VET is operating in the setup execution state.
- If you use the on node-list phrase, you cannot include node names in a device name or device list.

Example

```
Q-VET_setup> select devices /dev/rra1c /dev/rra2c /dev/rra3c on mynode
Process 1, group  exer for device /dev/rra1c
Process 2, group  exer for device /dev/rra2c
Process 3, group  exer for device /dev/rra3c
Q-VET_setup> deselect devices /dev/rra2c on mynode
```

disable

Description

Disables Q-VET features that are enabled by default or were previously enabled. (See the discussions of the corresponding **enable** commands.)

Format

```
disable { error_display display-type }
        { logging logging-type }
        { loop loop-type }
        { mode mode-type }
        { scrolling }
        { stop stop-type }
        { timeout }
        { trace trace-type }
```

Keywords and Arguments

error_display *display-type*

Shut off the display of error messages from all nodes during a Q-VET run. The description of the **enable error_display** command discusses the following display types:

basic

full

extended

all

Issue the **enable error_display** command to enable a level of reporting.

logging *logging-type*

Disable logging to a terminal log file or the system error log file. The description of the **enable logging** command contains a detailed discussion of logging types, which are as follows:

terminal

command

errors

You issue the **enable logging** command to enable the logging types.

loop ***loop-type***

Disable the looping options on all nodes.The description of the **enable loop** command contains a detailed discussion of loop types, which are as follows:

all

hard

last_subtest

soft

You issue the **enable loop** command to enable looping on error.

mode ***mode-type***

Deselect the run-time mode. The description of the **enable mode** command contains a detailed discussion of mode types, which are as follows:

automated

interactive

quick

truncate

verify

Run-Time Modes discusses Q-VET run-time modes. You issue the **enable mode** command to select the run-time mode.

scrolling

Disable noncontinuous scrolling of user display text on the local node. If scrolling is disabled, user terminal output text is displayed continuously without stopping.

You issue the **enable scrolling** command to enable user prompting on the local node after Q-VET has displayed *screen-size* lines. See the discussion of the **set screen_size** command.

stop ***stop-type***

Disable the stop-on-error options on all nodes. The description of the **enable stop** command contains a detailed discussion of stop types, which are as follows:

hard

soft

fatal

all

You issue the **enable stop** command to enable stop types.

timeout

Disable the ability of Q-VET to identify and stop "hung" processes. If timeout-checking is enabled, Q-VET attempts to stop processes that have not communicated with it within a predetermined time span.

Timeout-checking is enabled by default. You can also issue the **enable timeout** command to reenable the Q-VET timeout feature, if it has been disabled during the current session.

trace *trace-type*

Disable the trace message display types on all nodes. The description of the **enable trace** command contains a detailed discussion of trace types, which are as follows:

test

subtest

debug

all

You issue **enable trace** to enable trace types.

Restrictions

- You can issue the disable command with the following keywords while Q-VET is operating in the setup, active, or suspend execution state:

  error_display
  loop
  scrolling
  stop
  timeout

- You can issue the **disable logging** and **disable mode** commands only while Q-VET is operating in the setup execution state.
- A command line may contain only one disable command and one keyword and its corresponding arguments.

Example

```
Q-VET_setup> disable mode interactive
Q-VET_setup> disable scrolling
Q-VET_setup> disable trace all
```

disconnect

Description

Removes nodes or devices from the *configuration database*, that is, from the database of testable devices. The discussion of the **connect** command explains how to add a device or node to the Q-VET configuration database.

Format

```
disconnect  { nodes node-list }
            { node local }
            { devices device-list [on node-list] }
```

Keywords and Arguments

nodes *node-list*

Remove all the specified nodes' devices from the database. Node Names and Lists describes the *node-list* argument.

node local

Remove the local system devices from the database.

devices *device-list* [on *node-list*]

Remove the devices specified in *device-list*. Device Names and Lists describes the *device-list* argument.

The optional phrase **on** *node-list* specifies the node(s) on which this command takes effect.If you omit this phrase,the local node is the default. Node Names and Lists describes the *node-list* argument.

Restrictions

- You can issue the **disconnect** command only while Q-VET is operating in the setup execution state.
- If you use the **on** *node-list* phrase, you cannot include node names in a device name or device list.

Example

```
Q-VET_setup> **disconnect devices /dev/rra0c**
Q-VET_setup> **show devices /dev/rra0c**
     No matching devices found in configuration database.
```

display errorlog

Description

Displays the operating system's error log file. You may specify system log qualifiers with this command.**display errorlog** [**with** *argument-list*]

Format

```
display errorlog [with argument-list]
```

Arguments

argument-list

The argument list may contain any valid qualifiers for the system error log report facility.If you do not specify an argument list, the **display errorlog** command displays all errors that have occurred on the current day.

Restrictions

- None concerning execution state: you can issue this command while Q-VET is operating in the setup, active, or suspend execution state.
- Use this command only if the operating system under Q-VET is running has an error log report generator.
- You need superuser privileges to display the error log.

Example

In this example, the user issues the **display errorlog** command to view the contents of the error log since 01-Jan-1993.

```
Q-VET_setup> display errorlog with “-t s:01-Jan-1993”
```

drop

Description

Drops specified processes and devicesfrom the current run.

You can use the **drop** command to drop some or all selected devices or processes before you issue the **start** command; you can then use the **add** command to add one or more devices or processes at a time to load the system gradually. You also **drop** devices or processes gradually from a fully loaded system after you have issued the **start** command.

Q-VET stops dropped processes if they are running.

Format

```
drop { devices device-list [on node-list] }
     { processes process-list [on node-list] }
```

Keywords and Arguments

devices *device-list* [on *node-list*]

Drop all specified processes for the devices specified in *device-list*. Device Names and Lists describes the *device-list* argument.

The optional phrase **on** *node-list* specifies the node(s) on which this command takes effect. If you omit this phrase, the local node is the default. Node Names and Lists describes the *node-list* argument.

processes *process-list* [on *node-list*]

Drop processes selected in *process-list*. Process Numbers and Lists describes the *process-list* argument.

The optional phrase **on** *node-list*specifies the node(s) on which this command takes effect. If you omit this phrase, the local node is the default. Node Names and Lists describes the *node-list* argument.

Restrictions

- None concerning execution state: you can issue this command while Q-VET is operating in the setup, active, or suspend execution state.
- If you use the **on** *node-list* phrase, you cannot include node names in a device name or device list.

Example

```
Q-VET_setup> **drop devices all**
Q-VET_setup> **add device /dev/rra0c**
Q-VET_setup> **enable loop hard**
Q-VET_setup> **start**
...starting [Process 1, Pass 1] group exer for device /dev/rra0c.

Q-VET_active> **add device /dev/rra1c**
...starting [Process 1, Pass 1] group exer for device /dev/rra1c.

Q-VET_active> **add device /dev/rra2c**
...starting [Process 1, Pass 1] group exer for device /dev/rra2c.
```

duplicate process

Description

Creates duplicate copies of a process definition. The copies contain all the process-specific run-time parameters of the original process.

After you have duplicated a process definition, you can use the following commands to act on the process's contents or show its characteristics:

add	**continue**	**deselect**
drop	**select**	**set**
show	**stop**	**terminate**

Format

```
duplicate process process-number [duplication-count] [on node-list]
```

Arguments

***process-number [duplication-count]* [on *node-list*]**

Duplicate the process specified by *process-number*. Process Numbers and Lists describes the *process-number* argument.

The optional *duplication-count* argument specifies the number of copies; if you omit this argument, Q-VET makes only one copy.

The optional phrase **on** *node-list* specifies the node(s) on which this command takes effect. If you omit this phrase, the local node is the default. Node Names and Lists describes the *node-list* argument.

Restrictions

- None concerning execution state: you can issue this command while Q-VET is operating in the setup, active, or suspend execution state.
- If you issue the **duplicate process** command while Q-VET is operating in the active execution state, the duplicate processes are initially dropped and must be added (see **add** command) to begin executing.

- If the duplication-count exceeds system-wide process limits, the duplicate command will duplicate only up to the process limits.

Example

```
Q-VET_setup> select device file
Process 1, group exer for device file
Q-VET_setup> duplicate proc 1 3
Process 2, group exer for device file
Process 3, group exer for device file
Process 4, group exer for device file
```

edit

Description

Creates or modifies a script. Script Files discusses script files.

When you issue the **edit** command, Q-VET invokes the text editor, as previously defined by your system.

Format

```
edit { none }
     { file-spec }
```

Arguments

none

Q-VET opens the script buffer for editing and invokes the text editor. Issue the **save** command to save the script buffer contents permanently. (See the note below.) Issue the **execute** command to execute the commands contained in the script buffer.

file-spec

Edit the script file specified by *file-spec*. Issue the **execute** *file-spec* command to execute the commands contained in the file.

Restriction

You can issue the **edit** command only when Q-VET is operating in the setup execution state.

Note

The script buffer's contents are not permanent. The first time you issue the **edit** command during a Q-VET session, the script buffer is empty. The commands you enter can be overwritten later. The script buffer's contents are lost when you terminate the Q-VET session.

Example

```
Q-VET_setup> edit my_script_file
```

enable

Description

Controls Q-VET run-time characteristics.

Keywords and Arguments

error_display *display-type*

Control the level of error reporting on all nodes. Display types can be as follows:

- **basic---**Enables display of **basic** error messages; enabled by default.

  Basic messages display a short statement identifying the kind of error being reported, such as write-check error for a disk. Every test contains a basic error message that it displays if it encounters an error.
- **full---**Enables display of **basic** and **full** error messages; enabled by default.

  Full messages contain more information about errors than basic messages, for example, the contents of relevant device registers. A test need not contain this level of detail.
- **extended---**Enables display of **basic**, **full**, and **extended** error messages; enabled by default.

  Extended messages display data when large amounts are available, for example, the contents of a large data buffer. A test need not contain this level of detail.

logging *logging-type*

Enable Q-VET to write to terminal log files and to error log files. The *logging-type* argument can be as follows:

- **terminal---**Enables logging of all terminal input and output both to your terminal and to a log file that you have specified with a **set logfile** command.

  If you have not specified a log file, Q-VET uses the default log file, *vet_termlog*.
- **command---**Enables logging of all commands (input to your terminal) to a command file that you have specified with the **set comfile** command.

If you have not specified a command file, Q-VET uses the default command file, **vet_command_log.**

- **errors---**Enables writing of messages to the system error log file. Logging of error messages is enabled by default,if you have the appropriate privileges for writing to the system error log file.

loop ***loop-type***

Control looping within a Q-VET process when it encounters an error. The *loop-type* argument can be as follows:

- **all---**Causes a process to start looping on the current subtest when either a hard error or a soft error is encountered.
- **hard---**Causes a process to start looping on the current subtest when a hard error is encountered; disabled by default.
- **last_subtest---**Causes a process to start looping on the subtest specified with the **select groups** command's **last_subtest** modifier.
- **soft---**Causes a process to start looping on the current subtest when a soft error is encountered; disabled by default.

Appendix A discusses types of errors encountered during tests.

mode ***mode-type***

Select any combination of various Q-VET run-time modes. The *mode-type* argument can be as follows:

- **automated---**Enabled when Q-VET is running under the control of a higher-level automated environment, such as a manufacturing automated testing system; disabled by default. When Q-VET operates in this mode, process startup, pass count, and completion messages are suppressed.
- **interactive---**Indicates that a user is present to respond to queries displayed on a terminal; enabled by default.
- If *interactive* mode is enabled, Q-VET fetches responses to user prompting messages from either the user's terminal or a command script if one is active. If *interactive* is disabled,
  Q-VET does not request user responses but takes the default values of all requests.
- **quick---**Causes tests to run in *quick* mode if they support it. Quick mode is a shortened mode of operation; disabled by default.

- **truncate---**If enabled and a Q-VET process reports an error, subsequent passes of that process execute only up to the subtest before the failing one; Q-VET then starts the next pass. Truncate mode helps to isolate intermittent errors by attempting to identify the minimum hardware activity needed to cause the error; disabled by default.
- **verify---**Causes script file input lines to be displayed as they are read from the file; disabled by default.

scrolling

Enable noncontinuous scrolling of user display text. Scrolling is continuous by default. If you enable scrolling, Q-VET displays *screen-size* lines and then asks you to press **[Return]** to view the next screen. Q-VET initially sets the screen size to the current size of your terminal screen. (See the discussion of the **set screen_size** command.)

stop *stop-type*

Causes Q-VET to stop executing a process once it detects and reports an error. This option is effective on all nodes. The *stop-type* argument can be as follows:

- **hard---**Causes a Q-VET process to stop execution on the first occurrence of a device hard error; disabled by default.
- **soft---**Causes a Q-VET process to stop execution on the first occurrence of a device soft error; disabled by default.
- **fatal---**Causes a Q-VET process to stop execution on the first occurrence of a device fatal error; disabled by default.
- **all**---Equivalent to issuing the **enable stop hard**, **enable stop soft**, and **enable stop fatal** commands.

Appendix A discusses types of errors.

timeout

Enables Q-VET's ability to identify and stop "hung" processes. If this option is enabled, Q-VET terminates processes that have not communicated with it within a specified time span. Timeout checking is enabled by default.(See the description of the **set timeout** command.)

trace *trace-type*

Control the display of trace messages during a
Q-VET run. Trace messages include the name of the currently executing subtest and optional programmer-supplied information. The *trace-type* argument can be as follows:

- **test---**Causes Q-VET to display the name of each test when it begins and completes;disabled by default.
- **subtest---**Causes Q-VET to display the name of each subtest when it completes; disabled by default.
- **debug---**Causes Q-VET to display programmer-defined trace messages;disabled by default.
- **all---**Equivalent to issuing the **enable trace test**, **enable trace subtest**, and **enable trace debug** commands.

Restrictions

- You can issue the enable command with the following keywords while Q-VET is operating in the setup, active, or suspend execution state:

  error_display

  loop

  scrolling

  stop

  timeout

  trace
- You can issue the **enable logging** and **enable mode** commands only while Q-VET is operating in the setup execution state.

Examples

```
Q-VET_setup> enable logging terminal ❶
   Tue Dec 31 10:37:02 1993
   Terminal logging is enabled. The logfile is vet_termlog.

   Q-VET_setup> drop devices all ❷

   Q-VET_setup> add device memory ❸

   Q-VET_setup> enable loop hard ❹
```

In order to isolate failures on device memory, the user:

❶ Makes sure that a log file records Q-VET generated input to and output from the terminal.

❷ Drops all devices from testing

❸ Adds device memory for testing

❹ Enables looping on a hard error

execute

Description

Runs the commands in your script buffer or in a script file. Issue an **enable mode verify** command to view script command lines as they are executed.

Argument

none

Execute the commands contained in the script buffer.

***script-file* [on *node-list*]**

Execute the commands contained in *script-file*. Script Files discusses the format and contents of Q-VET script files.

The optional phrase **on** *node-list* specifies the node(s) on which the script is executed. If you omit this phrase, the local node is the default. Node Names and Lists describes the *node-list* argument.

Restriction

You can issue the execute command only while Q-VET is operating in the setup execution state.

Example

```
Q-VET_setup> execute diskseek on mynode yournode hisnode
```

exit

Description

Terminates a Q-VET session on both the local and remote node(s) and returns control to the operating system. The **exit** command is equivalent to the **quit** command.

Note

If you issue the **exit** command while Q-VET is operating in the suspend execution state, Q-VET first terminates all suspended processes and then terminates the Q-VET session.

Format

```
exit
```

Restriction

You can issue the **exit** command only while Q-VET is operating in the setup or suspend execution state.

Example

```
Q-VET_setup> exit
```

foreign

Description

Selects for execution a program that does not conform to the Q-VET interfaces, that is, was not designed specifically to run under Q-VET.

You issue the **start** command to run the program.

Note

A file specified in a **foreign** command executes during the Q-VET run; in contrast,an operating-system-level command issued in a **system** command executes immediately.

Format

```
foreign command-string
```

Argument

command string

Specify the name of the program to run and its options or qualifiers, if required.

Restrictions

- You can issue the **foreign** command only while Q-VET is operating in the setup execution state.
- Programs that do not conform to Q-VET interfaces are unaffected by Q-VET execution control commands, except **add**, **drop**, **start**, and **terminate**.
-

Example

```
Q-VET_setup> foreign my_program
Process 1, FOREIGN: ./my_program
Q-VET_setup> foreign my_program -a option1 -b \

option2
Process 2, FOREIGN: ./my_program -a option1 -b option2
Q-VET_setup> foreign your_program -a -b -c
Process 3, FOREIGN: ./your_program -a -b -c
```

help

Description

Displays information about Q-VET commands, keywords, and arguments.

The **help** command:

- Displays a list of available Q-VET commands
- Describes the context in which the commands are used
- Explains run-time parameters

Format

```
help [topic[subtopic]]
```

Arguments

none

Display a list of commands and other topics about which help is available.

topic [subtopic]

Display information about the topic and the optional subtopic.

Restriction

None concerning execution state: you may issue this command while Q-VET is operating in the setup, active, or suspend execution state.

Example

```
Q-VET_setup> help

Type a topic or subtopic keyword to get help text
Type "?" for a list of subtopics
Type RETURN to back up to the previous topic level

help

          Displays information about Q-VET
          commands, keywords, and arguments.
```

```
     Format:

          help [topic [subtopic]]

          The help command without a topic displays a list of commands

          about which help is available.

Topics available:

accept              add                 certify             command_summary
comment-line        connect             continuation        continue
control-c           control-z           deselect            disable
disconnect          display             drop                duplicate
edit                enable              execute             exit
foreign             help                install             load
quit                remove              save                select
set                 show                system              start
stop                terminate           vet                 wait
write

Help topic:

Q-VET_setup>
```

install

Description

Adds a Q-VET program to the *program database* file.

Before you can run a Q-VET program, you must install it in Q-VET's program database, which contains information about each Q-VET program known to Q-VET including:

File name

Supported device types

List of the options and default values

List of groups

List of tests for each group

The Q-VET installation procedure automatically installs all Q-VET generic exercisers in the system permanent program database.

Use the **show installed** command to view the file names of installed Q-VET programs. If you want a group that is part of the installed file to constitute the default test group for a device, you must then issue the **select default** command.

Format

```
install [install-type] file-spec [on node-list]
```

Arguments

[install-type] file-spec [* on *node-list]

The optional *install-type* can be **permanent** or **temporary**. Differences are as follows:

- A program installed by means of the **permanent** argument becomes part of the permanent Q-VET program database and is available during all subsequent Q-VET sessions. This is the default.
- A program installed by means of the **temporary** argument is installed in a temporary database and is available only during the current Q-VET session.

Note

If you install a file specifying the **temporary** installation type, you must remove the file specifying the **temporary** installation type.

The *file-spec* argument names the file to install.

The optional **on** *node-list* specifies the node(s) on which this command takes effect.If you omit this phrase, the local node is the default. Node Names and Lists describes the *node-list* argument.

Restrictions

- You can issue the **install** command only while Q-VET is operating in the setup execution state.
- The temporary database takes precedence over the permanent database. Thus, if a group name is the same in the temporary and permanent databases, only the group in the temporary database is selected.
- **You cannot install a Q-VET program if you have already issued the** select command to select a device or process. Deselect processes or devices before issuing the **install** command.
- If a permanent database file does not exist, Q-VET creates one.
- The temporary database file that you create is deleted when you terminate the Q-VET session.

Example

```
Q-VET_setup> install vet_exer_file

Installation completed successfully for vet_exer_file

Q-VET_setup> select default exer for file
```

load

Description

Creates a default load for the system configuration that you are testing.

When you issue the **load** command, Q-VET sets up a system load of Q-VET processes to test it. Q-VET's ability to run many exerciser processes simultaneously means that system loads needed for installation verification and other systemwide testing can be created.

After you issue the **load** command, you can issue the **show selected groups for all** or the **show processes all** command to see what groups and devices the **load** command has selected. You then issue the **start** command to start the run.

If you have already issued the **connect** command to connect a remote node(s), Q-VET includes the node(s) in the system load.

Note

If you want to add a tape to the devices tested, you must **select** the device before you **start** the run.

Format

```
load { none }
     { with default }
     { with cpu n1 memory n2 file n3 network n4 disk n5 }
```

Keywords and Arguments

none

If you issue the command without keywords or arguments, Q-VET prompts you for the number of processes to load for each device---cpu, memory, file, network, and disk. If you do not enter a reply and press **[Return]**, Q-VET takes the default number of processes.
Q-VET also prompts you for the number of disks to include in the load. If you do not enter a reply and press **[Return]**, Q-VET takes the default number of disks.

with default

Create a default load of one process per device for the system that you are testing. Devices tested include the cpu, memory, file system, network, and disk(s).

with cpu *n1* memory *n2* file *n3* network *n4* disk *n5*

Specify the number of processes to run for each device that you specify (**cpu**, **memory**, **file**, **network**, and **disk**). You may enter one or more device types and the corresponding number of processes. If you do not specify a device, Q-VET takes the corresponding default number of processes.

Examples

In the following example, the user enters the **load** command without keywords; Q-VET prompts for the number of processes corresponding to each device and the number of disk drives to test.

In each case, the user accepts the default by pressing **[Return]** without entering a response. The system loads the default tests and returns the Q-VET_setup> prompt. The user must issue the **start** command to start the tests.

```
Q-VET_setup> load

Enter number of processes for CPU device (decimal [1]): [Return]
Enter number of processes for MEMORY device (decimal [1]): [Return]
Enter number of processes for NETWORK device (decimal [1]): [Return]
Enter number of processes for FILE device (decimal [1]): [Return]
Enter number of processes for DISK device (decimal [1]): [Return]
Enter number of disks (decimal [5]): [Return]

Process 1, group exer for device cpu
Process 2, group exer for device memory
Process 3, group exer for device network
Process 4, group exer for device file .
   .
   .
   .
Q-VET_setup> start
```

In the following example, the user wishes to run the default exercisers and enters only the **load with defaults** command. The system loads the default tests and returns the Q-VET_setup> prompt. The user must issue the **start** command to start the tests.

```
Q-VET_setup> load with defaults
Process 1, group exer for device cpu
Process 2, group exer for device memory
Process 3, group exer for device network
Process 4, group exer for device file
Process 5, group exer for device /dev/rra0a
Process 6, group exer for device /dev/ra0a
Process 7, group exer for device /dev/rra0b
Process 8, group exer for device /dev/ra0b
Process 9, group exer for device /dev/rra0c
Q-VET_setup> start
```

In this example, the user enters the **load with cpu 4** command and argument to direct Q-VET to run four processes on the CPU. Q-VET runs the default number of processes on all other devices. The user also specifies that only one disk drive is to be tested out of a possible five. The system loads the tests and returns the Q-VET_setup> prompt. The user must issue the **start** command to start the tests.

```
Q-VET_setup> load with cpu 4

Enter number of disks (decimal [5]): 1

Process 1, group exer for device cpu
Process 2, group exer for device cpu
Process 3, group exer for device cpu
Process 4, group exer for device cpu
Process 5, group exer for device memory
Process 6, group exer for device network
Process 7, group exer for device file
Process 8, group exer for device /dev/rra0a

Q-VET_setup> start
```

quit

Description

Terminates a Q-VET session and returns control to the operating system. The **quit** command is equivalent to the **exit** command. You can issue the **quit** command only while Q-VET is operating in the setup or suspend execution state.

Format

```
quit
```

Restriction

You can issue the quit command only while Q-VET is operating in the setup or suspend execution state.

Example

```
Q-VET_setup> quit
%
```

remove

Description

Removes a Q-VET program from the Q-VET *program database*, which contains information about each Q-VET program.

The **remove** command is the opposite of the **install** command. See the discussion of the **install** command for detailed information.

You issue the **show installed** command to view the file names of installed Q-VET programs.

Format

```
remove install-type file-spec [on node-list]
```

Arguments

***install-type file-spec* [on *node-list*]**

The *install-type* can be either **temporary** or **permanent**. If you install a file specifying the **temporary** installation type, you must remove the file specifying the **temporary** installation type.

The *file-spec* argument specifies the name of the Q-VET program to remove from the database.

The optional phrase **on** *node-list* specifies the node(s) on which this command takes effect. If you omit this phrase,the local node is the default. Node Names and Lists describes the *node-list* argument.

Restriction

You can issue the **remove** command only while Q-VET is operating in the setup execution state.

Example

```
Q-VET_setup> show installed
Installed files (temporary):
 vet_exer_cpu
Installed files (/usr/sbin/vet_database):
 vet_exer_file
Q-VET_setup> remove temporary vet_exer_cpu
Q-VET_setup> remove permanent vet_exer_file
```

save

Description

When you use the **edit** command without a file specification to create a new script file, Q-VET stores that file in its script buffer. You use the **save** command to save the contents of the script buffer *after* you have terminated the editing session.

Format

```
save file-spec
```

Argument

file-spec

Save the contents of the script buffer in the file specified by the *file-spec* argument.

Restriction

You can issue the save command only while Q-VET is operating in the setup execution state.

Note

The script buffer's contents are not permanent. Save a newly edited script buffer in a file before you exit from the current Q-VET session, if you want a permanent copy.

Example

```
Q-VET_setup> edit ❶
   .
   .
   .
Q-VET_setup> save /usr/users/my_dir/myfile ❷ ❸
```

❶ The user initiates an editing session.

❷ The user edits the script buffer and terminates the editing session.

❸ The user later saves the contents of the script buffer in /usr/users/my_dir/myfile after ending the editing session.

select

Description

Chooses the devices that you want to test,the test groups, tests, and options that you want to run. Sets the default group for a device in the program database.

Format

select { devices *device-list* [on *node-list*] }
{ groups *group-list* [with *selection-criteria*] for *device-name* [on *node-list*] }
{ tests *test-list* [last_subtest *subtest-number*] for *process-number* [on *node-list*] }
{ options [*options-list*] for *process-number* [on *node-list*] }
{ default *group-name* for *device-name* [on *node-list*] }

Keywords and Arguments

devices *device-list* [on *node-list*]

Specify the hardware devices to be tested. Device Names and Lists describes the *device-list* argument. If a default group exists, Q-VET assigns it to the device.

The optional phrase **on** *node-list* specifies the node(s) on which this command takes effect. If you omit this phrase,the local node is the default. Node Names and Lists describes the *node-list* argument.

groups *group-list* [with *selection-criteria*] for *device-name* [on *node-list*]

Select *device-name* for testing if it has not already been selected. Select the test groups to test the device and optionally select run-time options and the tests to be run. If you issue multiple **select groups** commands for the same device, Q-VET creates multiple process definitions.

Group Names and Lists describes the *group-list* argument. The *device-name* argument specifies the device to test.

The optional phrase **with** *selection-criteria* chooses run-time options and includes or excludes the tests that will be run on *device-name*:

- **options** *[option-list]*
  Select group-specific run-time options. You can list them in *options-list* or omit them. If you omit them, Q-VET prompts you for the values of the group's options. The following example illustrates both ways of indicating run-time options.

- **included_tests** *test-list* **[last_subtest** *subtest-number]*
  Run only the tests specified in *test-list*. Test Numbers and Lists describes the *test-list* argument. The tests must belong to the test group that you have indicated. Issue the **show groups** command to display the numbers of the tests belonging to a group.

  The optional **last_subtest** *subtest-number* argument is the number of the last subtest to execute in the last (highest-numbered) selected test. Subtest Numbers describes the *subtest-number* argument.

  If you use a **set passcount** or **set runtime** command to specify multiple program passes and issue the **enable loop last_subtest** command, the program loops on this subtest, as shown in the following example.
- **excluded_tests** *test-list*
  Run all tests belonging to the group *except* those specified in *test-list*. Test Numbers and Lists describes the *test-list* argument. The tests must belong to the test group that you have indicated. Issue the **show groups** command to display the numbers of the tests belonging to a group.

The optional phrase **on** *node-list* specifies the node(s) on which this command takes effect. If you omit this phrase, the local node is the default. Node Names and Lists describes the *node-list* argument.

tests *test-list [* last_subtest *subtest-number]* for *process-number* [on *node-list*]

Modify the selected test list for a particular process.

- The *test-list* argument specifies the tests to be selected. Test Numbers and Lists describes the *test-list* argument.
- Although you cannot select specific subtests for execution, you can use the optional **last_subtest** argument to specify that a test execute only up to and including a particular subtest. Subtest Numbers describes the *subtest-number* argument.

  You can then issue the **select groups** command with the **last_subtest** argument to request that Q-VET loop on that subtest. Refer to Subtests for a further discussion of looping.
- The *process-number* argument specifies the process. You can specify **last** for this argument; **last** refers to the most recently referenced process number.

The optional phrase **on** *node-list* specifies the node(s) on which this command takes effect. If you omit this phrase, the local node is the default. Node Names and Lists describes the *node-list* argument.

options *[option-list]* for *process-number* [on *node-list*]

Modify the group-specific run-time options for a particular process.

If you specify the *options-list* argument, Q-VET modifies only the options that are listed; otherwise, Q-VET prompts for new values for all options.

The *process-number* argument indicates the process. You can specify **last** for this argument; **last** refers to the most recently referenced process number.

The optional phrase **on** *node-list* specifies the node(s) on which this command takes effect. If you omit this phrase, the local node is the default. Node Names and Lists describes the *node-list* argument.

default *group-name* for *device-name* [on *node-list*]

Select the group that is to be the default group for the specified device type. If you install a file and want it to become the default test group of a device, you must issue the **select default** command. Identifying information about the default group is stored in the program database. Use the **show groups** command to find a device's default group. Group Names and Lists describes the *group-name* argument.

The optional phrase **on** *node-list* specifies the node(s) on which this command takes effect. If you omit this phrase, the local node is the default. Node Names and Lists describes the *node-list* argument.

default *group-name* for *device-name* [on *node-list*]

Select the group that is to be the default group for the specified device type. If you install a file and want it to become the default test group of a device, you must issue the select default command. Identifying information about the default group is stored in the program database. Use the **show groups** command to find a device's default group. Group Names and Lists describes the *group-name* argument.

The optional phrase **on *node-list*** specifies the node(s) on which this command takes effect. If you omit this phrase, the local node is the default. Node Names and Lists describes the node-list argument.

Restrictions

- If you use the **on *node-list*** phrase, you cannot include node names in a device name or device list.
- You can issue the **select devices, select groups, select options**, and **select tests** commands while Q-VET is operating in the setup, active, or suspend execution state.

- You can issue the **select default** command only while Q-VET is operating in the setup execution state.
- Issuing the **select options** command to change a started process has no effect until the next time the process is started.
-

Examples

In this example, the user enters the **select groups** command with options and specifies them in the command line.

```
Q-VET_suspend> select group exerciser with options \
reads_per_iteration 5 iterations 100 for /dev/rra0c
```

In this example, the user enters the select groups command with options without specifying them in the command line. Q-VET prompts for the missing options.

```
Q-VET_suspend> select group exer with options for file
Process 1, group exer for device
file file_name (string [""]):
enable_writes ("yes" or "no"[yes]):
reads_per_iteration (decimal [1]):
block_size (decimal [512]):
start_block (decimal [0]):
end_block (decimal [499]):
step (decimal [0]):
iterations (decimal [1000]):
delay (decimal [0]):
pattern (decimal [0]):
error_check_level (decimal [3]):
read_only_verify ("yes" or "no"[no]):
key (decimal []):
save_file ("yes" or "no"[no]):
Q-VET_setup>
```

In this example, the user enters **select groups** command with included tests.

```
Q-VET_suspend> enable loop last_subtest ➊
Q-VET_suspend> select group exerciser with included_tests \
   1-5 last_subtest 14 for /dev/rra1c ➋
```

➊ Enables looping on the last subtest of the group about to run; that is, subtest 14.

➋ The user selects tests 1 through 5 in the exerciser test group to run on device /dev/rra1c and directs that test 5 execute only through subtest 14. (This is the last subtest.) The Q-VET pass completes after this subtest, unless the program finds an error, in which case it loops on subtest 14. See Subtests for a further discussion of looping.

set

Description

Establishes run-time characteristics such as execution mode, pass count, directory path, and run time. You also use the **set** command to specify the log file to which terminal I/O is written or the number of lines of the terminal display.

Format

```
set { affinity cpu-id [on node-list] }
    { error_threshold threshold [for process-list] [on node-list] }
    { execution execution-type [on node-list] }
    { comfile file-spec }
    { logfile file-spec  }
    { passcount pass-number [for process-list] [on node-list] }
    { passcount pass-number [for device-list] [on node-list] }
    { path path-list  [on node-list] }
    { runtime time [for process-list] [on node-list] }
    { runtime time [for device-list] [on node-list] }
    { screen_size number-of-lines }
    { timeout seconds }
```

Keywords and Arguments

affinity *cpu-id* [*on node-list*]

Establishes the default processor affinity for Q-VET subprocesses.
Q-VET subprocesses establish their processor affinity (if required) when they are first started. The default subprocess affinity automatically defaults to that of the Q-VET manager if the **set affinity** command is not specified. Changing the default processor affinity has no effect on existing Q-VET subprocesses.

error_threshold *threshold* [*for process-list*] [*on node-list*]

Defines the error threshold for the overall run or for a particular process that has been selected.

If the error threshold for the overall run is reached, all the processes (which are currently executing) will be stopped and Q-VET will transition to the suspend state.

If the error threshold for a particular process is reached, it (the process) will be stopped. The state transition of Q-VET will depend on the current state of all other processes which have been selected.

execution ***execution-type*** **[on** ***node-list*****]**

Control whether execution of multiple processes occurs in parallel or serially. The *execution-type* argument can be one of the following:

- **parallel---**All Q-VET processes execute concurrently. This execution type is the default.
- **serial---**Only one Q-VET process executes at a time and completes running before the next process starts executing. If multiple process definitions exist for one device, these processes complete in the order in which they were selected, before the processes for the next device begin.

Caution

During serial execution, each process completes a run before the next process is started. That is, each process must satisfy the conditions established by the **set passcount** or **set runtime** command before Q-VET starts the next process. If you have selected infinite run time and pass count, the first process started will run forever and no others will start.

The optional phrase **on** *node-list* specifies the node(s) on which this command takes effect. If you omit this phrase, the command takes effect on all connected nodes. Node Names and Lists describes the *node-list* argument.

comfile ***file-spec***

Specifies the location and name of a file in which to store commands entered during Q-VET execution. If you do not use **set comfile** *file-spec* specifically to establish a command file, Q-VET uses a default file, **vet_command_log**.

Issue the **enable logging command** command to write to the command file.

logfile ***file-spec***

Specify the location and name of a file in which to store terminal I/O created during Q-VET execution. This command affects both the local and remote nodes. If you do not use **set logfile** *file-spec* specifically to establish a log file, Q-VET uses a default file, *vet_termlog*. Issue the **enable logging** command to write to the log file.

passcount ***pass-number*** **[for** ***process-list*****]** **[on** ***node-list*****]**

Indicate the number of passes that each Q-VET process will execute before stopping.

The *pass-number* specifies the number of passes. A *pass* is the one-time execution of all selected tests in a particular Q-VET process.

A pass count of 0 specifies infinite passes. The default value is 1, unless you have issued the **set runtime** command to indicate a discrete run time, in which case the default pass count is 0.

If you issue both the **set passcount** and **set runtime** commands to specify both pass count and run time, the first condition that is satisfied ends the Q-VET run. This condition can be either *passcount* or *runtime*.

The optional phrase **for** *process-list* sets the pass count only for processes that you specify with the *process-list* argument. Process Numbers and Lists describes the *process-list* argument.

The optional phrase **on** *node-list* specifies the node(s) on which this command takes effect. If you omit this phrase and have specified no *process-list* argument, the command takes effect on all connected nodes. Node Names and Lists describes the *node-list* argument.

passcount *pass-number* [for *device-list*] [on *node-list*]

Indicate the number of passes that each Q-VET process will execute before stopping.

The *pass-number* specifies the number of passes. A *pass* is the one-time execution of all selected tests in a particular Q-VET process.

A pass count of 0 specifies infinite passes. The default value is 1, unless you have issued the **set runtime** command to indicate a discrete run time, in which case the default pass count is 0.

If you issue both the **set passcount** and **set runtime** commands to specify both pass count and run time, the first condition that is satisfied ends the Q-VET run. This condition can be either *passcount* or *runtime*.

The optional *device-list* argument is a list of one or more devices, optionally including the path, node, or both to which a device belongs. A blank character (space) must separate items in the list.

You issue the **show devices all** command to find the names of devices connected to your system.

Path names can include node names. (If you enter a device name without a node name, Q-VET assumes that the device is local.)

You can often enter the wildcard character (*) to match all remaining characters.

If you enter **all** as the device-list, Q-VET acts on all devices.

path *path-list* [on *node-list*]

Define a list of directory paths. Path Names describes the *path* argument. The *path-list* argument is a list of directory path names. A blank character (space) must separate items in the list.

Q-VET uses the path list as a directory search list. When Q-VET looks for a file to read or execute, it follows each directory path specification in the list in the order in which it was named, until it finds the first occurrence of the file.

Q-VET initially sets the path list to the user's current working directory plus the directory in which Q-VET resides.

The optional phrase **on** *node-list* specifies the node(s) on which this command takes effect. If you omit this phrase, the command takes effect on all connected nodes. Node Names and Lists describes the *node-list* argument.

runtime *time* [for *process-list*] [on *node-list*]

Indicate the total run time for which each Q-VET process is allowed to execute. The section Time describes the *time* argument.

A run time of 0 indicates infinite time; the default value is 0. Since the default pass count is 1, the length of a Q-VET run is one pass, unless otherwise specified.

If you use the **set passcount** command to specify the pass count with the **set runtime** command to specify run time, the first condition completed (either pass count or run time) ends the Q-VET run. If you use the **set runtime** command to specify run time but no pass count, tests execute until the run time expires, even if multiple passes are required to attain the specified run time.

If you use the **set execution** command to set the execution mode to **serial** and select multiple devices,each Q-VET process executes for the specified run time.

The optional phrase **for** *process-list* sets the run time only for specified processes. Process Numbers and Lists describes the *process-list* argument. You cannot specify the *process-list* argument and the *device-list* argument in the same command.

The optional phrase **on** *node-list* specifies the node(s) on which this command takes effect. If you omit this phrase and have specified no *process-list* argument, the command takes effect on all connected nodes. Node Names and Listsdescribes the *node-list* argument.

runtime *time* [for *device-list*] [on *node-list*]

Indicate the total run time for which each Q-VET process testing the specified device(s) is allowed to execute. The section Time describes the *time* argument.

A run time of 0 indicates infinite time; the default value is 0. Since the default pass count is 1, the length of a Q-VET run is one pass, unless otherwise specified.

If you use the **set passcount** command to specify the pass count with the **set runtime** command to specify run time, the first condition completed (either pass count or run time) ends the Q-VET run. If you use the **set runtime** command to specify run time but no pass count, tests execute until the run time expires, even if multiple passes are required to attain the specified run time.

If you use the **set execution** command to set the execution mode to **serial** and select multiple devices, each Q-VET process executes for the specified run time.

The optional phrase **for** *device-list* sets the run time only for the specified devices. Device Names and Lists describes the *device-list* argument. You cannot specify the *process-list* argument and the *device-list* argument in the same command.

The optional phrase **on** *node-list* specifies the node(s) on which this command takes effect. If you omit this phrase and have specified no *device-list* argument, the command takes effect on all connected nodes. Node Names and Lists describes the *node-list* argument.

screen_size *number-of-lines*

Specify the number of lines of text to be displayed for each screen. If you issue an **enable scrolling** command to enable noncontinuous scrolling, the screen size is automatically set to the current size of your terminal screen.

The *number-of-lines* argument specifies the number of lines that will be displayed at a time; to display more lines, press **[Return]**.

timeout *seconds*

Specify the timeout interval in seconds. The timeout interval is the amount of time Q-VET waits between checks to see if Q-VET programs are running. See also **enable/disable timeout**.and **set timeout**

Restrictions

- You can issue the **set passcount, set runtime**, and **set screen_size** commands while Q-VET is operating in the setup, active, or suspend execution state.
- You can issue the **set execution, set logfile, set path**, and **set timeout** commands only while Q-VET is operating in the setup execution state.
- A command line may contain only one **set** command and one keyword and its corresponding arguments.

Examples

```
Q-VET_setup> deselect devices all

Q-VET_setup> select devices /dev/rra2c  /dev/rra3c

Q-VET_setup> set execution serial

Q-VET_setup> set runtime 1:00
```

show

Description

Displays the system devices, test groups and processes, or the current status of the Q-VET options, modes, and features that you can set or enable.

Format

```
show { affinity [on node-list] }
     { devices device-list [on node-list] }
     { error_display }
     { error_threshold [on node-list] }
     { execution [on node-list] }
     { groups group-list for device-name [on node-list] }
     { installed [on node-list] }
     { logging }
     { loop }
     { mode }
     { nodes }
     { passcount [on node-list] }
     { path [on node-list] }
     { process process-list [on node-list] }
     { runtime [on node-list] }
     { screen_size }
     { scrolling }
     { selected devices [on node-list] }
     { selected groups for device-name [on node-list] }
     { stop }
     { summary [summary-criteria] [on node-list] }
     { timeout }
     { trace }
     { types types-list [on node-list] }
     { version }
```

Keywords and Arguments

affinity [*on node-list*]

Displays the cpu-id for the Q-VET manager's processor affinity as well as the default cpu-id for the processor affinity for Q-VET subprocesses. The Q-VET manager's processor affinity is established when Q-VET is invoked or when a remote node is connected. If no processor affinity has been established (or processor affinity is not supported) then a cpu-id of "none" will be displayed.

devices *device-list* [on *node-list*]

Display hardware information about the devices that you specify in *device-list*. Specify **all** to view a list of all devices on all nodes. Device Names and Lists describes the *device-list* argument.

Q-VET obtains information such as device names, device types, storage capacity, and device status from the configuration database.

The optional phrase **on** *node-list* specifies the node(s) on which this command takes effect. If you omit this phrase, the local node is the default. Node Names and Lists describes the *node-list* argument.

error_display

Display the current status of the error message display options, as established by the **enable/disable error_display** commands. Error reporting levels are as follows:

basic

full

extended

error_threshold [on *node-list*]

Display the current error threshold for the overall run as well as for each process, as established by the **set error_threshold** command.

execution [on *node-list*]

Display the current execution type, as established by the **set execution** command. The descriptions of the **set** command discusses the **parallel** and **serial** execution types.

The optional phrase **on** *node-list* specifies the node(s) on which this command takes effect. If you omit this phrase, the command takes effect on all connected nodes. Node Names and Lists describes the *node-list* argument.

groups *group-list* for *device-name* [on *node-list*]

Display the number and a brief description of each test in the groups specified in *group-list* that correspond to the devices specified by *device-name*. Device Names and Lists describes the *device-name* argument. The device's default group is labeled as such. Group Names and Lists describes the *group-name* and *group-list* arguments.

If you enter **all** as the group list, you can specify *device-list* instead of *device-name*. Device Names and Lists describes the *device-list* argument.

The optional phrase **on** *node-list* specifies the node(s) on which this command takes effect. If you omit this phrase, the local node is the default. Node Names and Lists describes the *node-list* argument.

installed [on *node-list*]

List the file names of the Q-VET programs that have been installed. See the description of the **install** command.

The optional phrase **on** *node-list* specifies the node(s) on which this command takes effect. If you omit this phrase, the command takes effect on all connected nodes. Node Names and Lists describes the *node-list* argument.

logging

Display the current status (enabled or disabled) of error and terminal I/O logging, as set by the **enable logging** and **disable logging** commands.

loop

Display the current status of the looping options, as established by the **enable loop** command. Loop types are as follows:

- **hard---**Causes a process to start looping on the current subtest when a hard error is encountered; disabled by default.
- **soft---**Causes a process to start looping on the current subtest when a soft error is encountered; disabled by default.
- **all---**Causes a process to start looping on the current subtest when either a hard error or a soft error is encountered.
- **last_subtest---**Causes a process to start looping on the subtest specified with the **select groups last_subtest** command.

mode

Display the current status of the default run-time modes or the mode established by the **enable mode** command.

Mode types are as follows:

- **automated---**Enabled when Q-VET is running under the control of a higher-level automated environment, such as a manufacturing automated testing system; disabled by default. When Q-VET operates in this mode, process startup, pass count, and completion messages are suppressed.
- **interactive---**Indicates that a user is present to respond to queries displayed on a terminal; enabled by default. If enabled, Q-VET fetches responses to requests from either the user's terminal or a command script, if one is active. If disabled, Q-VET does not request user responses but takes the default values of all requests.
- **quick---**Causes tests to run in *quick* mode, if they support it. Quick mode is a shortened mode of operation; disabled by default.
- **truncate---**If enabled and a Q-VET process reports a device error, subsequent passes of that process execute only up to the subtest before the failing one; Q-VET then starts the next pass. Truncate mode helps to isolate intermittent errors by attempting to identify the minimum hardware activity needed to cause the error; disabled by default.
- **verify---**Causes script file input lines to be displayed as they are read from the file; disabled by default.

nodes

Display the nodes currently listed in the configuration database and their operating system, transport type, and sizer version.

passcount [on *node-list*]

Display the current value of the pass count for each process, as established by the **set passcount** command.

The optional phrase **on** *node-list* specifies the node(s) on which this command takes effect. If you omit this phrase, the command takes effect on all connected nodes. Node Names and Lists describes the *node-list* argument.

path [on *node-list*]

Display the current directory path search list, as established by the **set path** command.

The optional phrase **on** *node-list* specifies the node(s) on which this command takes effect. If you omit this phrase, the command takes effect on all connected nodes. Node Names and Lists describes the *node-list* argument.

process *process-list* [on *node-list*]

Display the contents of process definitions. Process Numbers and Lists describes the *process-list* argument.

The optional phrase **on** *node-list* specifies the node(s) on which this command takes effect.If you omit this phrase,the local node is the default. Node Names and Lists describes the *node-list* argument.

***runtime* [on *node-list*]**

Display the currently selected run time of each process, as established by the **set runtime** command.

The optional phrase **on** *node-list* specifies the node(s) on which this command takes effect. If you omit this phrase, the command takes effect on all connected nodes. Node Names and Lists describes the *node-list* argument.

screen_size

Display the current user display screen size, as established by the **set screen_size** command.

scrolling

Display whether continuous scrolling of user display text is enabled or disabled, as established by the **enable scrolling** command.

selected devices [on *node-list*]

Display a list of all devices currently selected for testing, as established by the **select devices** command.

The optional phrase **on** *node-list* specifies the node(s) on which this command takes effect. If you omit this phrase, the command takes effect on all connected nodes. Node Names and Lists describes the *node-list* argument.

selected groups for *device-name* [on *node-list*]

Display the name of each selected group for the specified device, as established by the **select groups** command.

If you enter **all** as the device name, Q-VET displays information about all devices.

The optional phrase **on** *node-list* specifies the node(s) on which this command takes effect. If you omit this phrase, the local node is the default. Node Names and Lists describes the *node-list* argument.

stop

Display the current status of the stop-on-error options, as established by the **enable stop** command. Stop types are as follows:

- **hard---**Causes a Q-VET process to stop execution on the first occurrence of a device hard error; disabled by default.
- **soft---**Causes a Q-VET process to stop execution on the first occurrence of a device soft error; disabled by default.
- **fatal---**Causes a Q-VET process to stop execution on the first occurrence of a device fatal error; disabled by default.
- **all---**Equivalent to issuing the **enable stop hard**, **enable stop soft**, and **enable stop fatal** commands.

summary [*summary-criteria*] [on *node-list*]

Display a summary of the most recent Q-VET run. Summaries consist of the following:

- A general display of Q-VET information, such as the run time, number of passes, and errors for each process.
- A process-specific display associated with the selected test groups. Test groups may or may not display such information.

The optional *summary-criteria* argument can be one of the following:

- **with output** *file-spec* [**on** *node-list*]
- The optional phrase **with output** *file-spec* sends the summary report to a device or a file other than the user's terminal.

  The *file-spec* argument specifies the file.

  The optional phrase **on** *node-list* specifies the node(s) on which this command takes effect. If you omit this phrase, the local node is the default.
- **for** *device-list* [**with output** *file-spec*] [**on** *node-list*]

  The optional phrase **for** *device-list* displays the summary for the devices specified in *device-list*. Device Names and Lists describes the *device-list* argument.

  The optional phrase **with output** *file-spec* argument sends the summary to a device or a file other than the user's terminal.

  The *file-spec* argument specifies the file.
- The optional phrase **on** *node-list* specifies the node(s) on which this command takes effect. If you omit this phrase, the local node is the default.

- **for** *process-list* [**with output** *file-spec*] [**on** *node-list*]
  The optional phrase **for** *process-list*, displays summaries only for the specified processes. Process Numbers and Lists describes the *process-list* argument.

  The optional phrase **with output** *file-spec* argument sends the summary to a device or a file other than the user's terminal.

  The *file-spec* argument specifies the file.

  The optional phrase **on** *node-list* specifies the node(s) on which this command takes effect. If you omit this phrase, the local node is the default.

timeout

Display whether timeout checking is enabled, and, if it is, the timeout interval in seconds under which Q-VET is running.

The timeout interval is the amount of time Q-VET waits between checks to see if Q-VET programs are running.

trace

Display the current status of the trace message display options, as established by the **enable trace** command. Trace types are as follows:

- **test---**Causes Q-VET to display the name of each test as it begins.
- **subtest---**Causes Q-VET to display the name of each subtest as it begins.
- **debug---**Causes Q-VET to display programmer-defined trace messages.
- **all---**Equivalent to issuing the **enable trace test**, **enable trace subtest**, and **enable trace debug** commands.

types *types-list* [on *node-list*]

Displays information about the types that you specify in *types-list*.

The *types-list* argument is a list of one or more device types. A blank character (space) must separate items in the list.

You issue the **show types all** command to find the names of device types connected to your system.

If you enter **all** as the *types-list*, Q-VET acts on all types.

version

Display the Q-VET version number.

Restrictions

- None concerning execution state: you may issue this command while Q-VET is operating in the setup, active, or suspend execution state.
- If you use the **on** node-list phrase, you cannot include node names in a device name or device list.
- A command line may contain only one **show** command with its corresponding keyword and argument.

Examples

In this example, the user issues the **show devices all** command to view a list of all devices that are known to Q-VET.

```
Q-VET_setup> show devices all
 SYSTEM  DEVICE               TYPE      SUBTYPE     SIZE      STATUS
 ------  -----                ----      -------     ----      ------
 chaaly  memory               V_MEM               25165824
 chaaly  file                 FILE_DATA
 chaaly  cpu                  CPU       risc
 chaaly  network              NETWORK   tcpdec
 chaaly  /dev/rmt0l           TAPE      tk70                  LOCAL_RW
 chaaly  /dev/nrmt0l          TAPE      tk70                  LOCAL_RW
 chaaly  /dev/rmt0h           TAPE      tk70                  LOCAL_RW
 chaaly  /dev/nrmt0h          TAPE      tk70                  LOCAL_RW
 chaaly  terminal             TERMINAL
 chaaly  printer              PRINTER
 chaaly  video                VIDEO
Q-VET_setup>
```

In this example, the user issues the **show groups all for *tape*** command to view a list of the available tests of a tape device.

```
Q-VET_setup> show groups all for tape
          Group:  exer
        Devices:  tape
           File:  vet_exer_tape
          Tests:  1 - Tape Write/verify/space records Test
                  2 - Tape Write/verify/space file test
          Group:  tape_all
        Devices:  tape
           File:  vet_exer_tape
          Tests:  1 - Tape Write/verify/space records Test
```

```
                  2 - Tape Write/verify/space file test
                  3 - Physical EOT test
Q-VET_setup>
```

In this example, the user runs the cpu exerciser and then issues the **show summary** command to view a summary of testing and test results.

```
Q-VET_setup> select group exer for cpu
Q-VET_setup> start
...starting [Process 1, Pass 1] group exer for device cpu.
...completed [process 1] group exer for device cpu.
...testing completed. Total errors = 0.
Q-VET_setup> show summary

Summary of all processes:
    Process 1: Group exer for device cpu

Total Errors for all processes: 0

====================================================

Process 1: Group exer for device cpu
       Requested runtime:      0 hours   0 minutes   0 seconds
       Elapsed runtime:        0 hours   0 minutes  57 seconds
       Remaining runtime:      0 hours   0 minutes   0 seconds

       Requested passcount:    1
       Completed passcount:    1
   .
   .
   .
End of summary.

Q-VET_setup>
```

start

Description

Starts a Q-VET run.

You issue the **start** command after you have selected the devices to be tested and the test groups that will test them. Issuing the **start** command runs the Q-VET processes that exercise the devices.

After you issue the **start** command, Q-VET enters the active execution state and begins to execute the processes. The active execution state prompt is displayed only after you press **[Return]**.

Format

```
start
```

Restriction

You can issue the start command only while Q-VET is operating in the setup execution state.

Example

```
Q-VET_setup> select devices memory
Process 1, group exer for device memory
Q-VET_setup> start
...starting [Process 1, Pass 1] group exer for device memory.
...completed [process 1] group exer for device memory.
...testing completed. Total errors = 0.
Q-VET_setup>
```

stop

Description

Stops the execution of Q-VET processes.

After you issue the **stop** command, all specified processes stop executing. If all processes are stopped, Q-VET enters the suspend execution state.

Format

```
stop { none }
     { processes process-list [on node-list] }
     { devices device-list [on node-list] }
```

Keywords and Arguments

none

Stop all processes on all nodes.   Issuing the **stop** command without arguments is equivalent to pressing **[Ctrl/C]** during a Q-VET run.

devices *device-list* [on *node-list*]

Stop all processes associated with the devices specified in *device-list*. Device Names and Lists describes the *device-list* argument.

The optional phrase **on** *node-list* specifies the node(s) on which this command takes effect. If you omit this phrase, the local node is the default. Node Names and Lists describes the *node-list* argument.

processes *process-list* [on *node-list*]

Stop the processes specified in *process-list*. Process Numbers and Lists describes the *process-list* argument.

The optional phrase **on** *node-list* specifies the node(s) on which this command takes effect. If you omit this phrase,the local node is the default. Node Names and Lists describes the *node-list* argument.

Restrictions

- If you use the on node-list phrase, you cannot include node names in a device name or device list.
- You can issue the stop command only while Q-VET is operating in the active execution state.

Example

```
Q-VET_setup> start
...starting [Process 1, Pass 1] group exer for device memory.
Q-VET_active> stop
...stopping [Process 1] group exer for device memory.
Q-VET_suspend>
```

system

Description

Passes a command to the operating system for execution.

You can pass any command that you ordinarily specify at the operating system prompt. When the command completes, control returns to Q-VET.

Note

An operating-system-level command issued in a **system** command executes immediately; a file specified in a **foreign** command executes during the Q-VET run.

Format

```
system command-line
```

Argument

command-line

Specify a command line to pass to the operating system for execution. The *command-line* argument must contain a valid command with valid keywords and arguments if needed.

Restriction

None concerning execution state: you may issue this command while Q-VET is operating in the setup, active, or suspend execution state.

Example

```
Q-VET_setup> system ls  -l /exer/disk/old
```

terminate

Description

Terminates specific or all stopped processes and returns Q-VET to the setup execution mode. (You use the **stop** command to stop running processes.)

Format

```
terminate { none }
          { devices device-list [on node-list]    }
          { processes process-list [on node-list] }
```

Keywords and Arguments

none

Terminate all suspended processes on all connected nodes, and display the setup prompt.

devices *device-list* [on *node-list*]

Terminate the suspended processes that are associated with the devices specified in *device-list*. Device Names and Lists describes the *device-list* argument.

The optional phrase **on** *node-list* specifies the node(s) on which this command takes effect. If you omit this phrase, the local node is the default. Node Names and Lists describes the *node-list* argument.

processes *process-list* [on *node-list*]

Terminate the processes specified in *process-list*. Process Numbers and Lists describes the *process-list* argument.

The optional phrase **on** *node-list* specifies the node(s) on which this command takes effect. If you omit this phrase, the local node is the default. Node Names and Lists describes the *node-list* argument.

Restrictions

- If you use the on node-list phrase, you cannot include node names in a device name or device list.
- You can issue the terminate command only while Q-VET is operating in the suspend state.

Example

```
Q-VET_setup> **start**
...starting [Process 1, Pass 1] group exer for device file.
[Ctrl/C]
...stopping [Process 1] group exer for device file.

Q-VET_suspend> terminate
...terminated [process 1] group exer for device file.

Q-VET_setup>
```

vet

Description

Starts a Q-VET session.

You issue the **vet** command at your system prompt.

Format

```
vet { none }
    { element1 [element2] [element3] ... [element n] }
```

Arguments

none

Start a Q-VET session. Q-VET sizes the system

Depending on how your session environment has been set up, Q-VET either enters command-line mode and displays the setup prompt, and you control the session by entering Q-VET commands; or it displays its main window using DECwindows Motif or Windows NT, and you control the session with the mouse and keyboard.

You can control whether Q-VET operates in window or command-line mode along with many other operations by specifying command elements.

element1 [element2] [element3] ... [element n]

For example, create processes according to your arguments, run the processes, and return control to the operating system once testing has completed.

Each element consists of an option possibly followed by an argument. If the option takes more than one argument, surround the argument list with quotation marks and separate arguments with spaces, as in the following example. Options and arguments are as follows:

- **-affinity *cpu-id***
  Specifies a specific processor to run the Q-VET processes on.
- **-d *device-list***
  Specify the devices to be tested. Device Names and Lists describes the device-list argument.
- **-display *display-device-name***
  Redirect the display to the specified display device (DECwindows Motif only).

- **-f** ***file-name***
  Specify a script file (file-name) containing Q-VET commands to be executed by Q-VET.
- **-F**
  Specify the default script file (vet_script.com) to be executed by Q-VET.
- **-g** ***group-list***
  Run the test groups listed in group-list. Group Names and Lists describes the group-list argument. If this option is absent and you specify **-d**, Q-VET executes the default group.
- **/int=char**
  (OpenVMS only) Do not use the OpenVMS interface; use the command interface.
- **-nw**
  (DIGITAL UNIX and Windows NT only) Do not use Q-VET's DECwindows Motif or Windows NT interfaces. Instead use Q-VET's command interface.
- **-p** ***passcount***
  Run the test groups for the number of passes specified in passcount. If this option is absent and you specify **-d**, the test group runs for a default pass count of 1 or the specified run time, whichever completes first.
- **-r** ***time***
  Run the test groups for the amount of time specified in time. If this option is absent and you specify **-d**, Q-VET runs the test groups for the specified pass count. The section Time describes the time argument.
- **-s**
  Display a run summary after the run has completed.
- **-[no]sizer**
  Run or do not run the Q-VET system sizer when Q-VET is started. Running the sizer is the default.

Examples

In this example, the user issues the vet command with the -nw display option. Q-VET starts a new session using its command-line interface. After running Q-VET, the user terminates the Q-VET session, and control returns to the operating system.

```
% vet -nw

Running system sizer on node mynode ... please wait.

Q-VET_setup>
```

```
   .

   .

   .

Q-VET_setup> exit

%
```

In this example, the user issues the vet command with options to exercise tape drives /dev/rmt0h and /dev /rmt2h using the default test group. The user specifies that the run will last for 30 passes or 15 minutes, whichever completes first.

Once the run has completed, Q-VET returns control to the operating system.

```
% vet -d "/dev/rmt0h /dev/rmt2h" -p 30 -r 15
Running system sizer on node mynode ... please wait.

   ...starting [Process 1, Pass 1] group exer for device /dev/rmt0h.
   ...starting [Process 2, Pass 1] group exer for device /dev/rmt2h.

   ...runtime has expired for [Process 1] group exer for device /dev/rmt0h.
   ...runtime has expired for [Process 2] group exer for device /dev/rmt2h.

   ...completed [Process 1] group exer for device /dev/rmt0h.
   ...completed [Process 2] group exer for device /dev/rmt2h.

   ...testing completed.  Total errors = 0

%
```

wait

Description

Controls the amount of elapsed time before Q-VET reads the next script file command. Each command that you do not want to be executed immediately must be preceded by the **wait** command, as in the example below.

Format

```
wait { none }
     { time }
```

Arguments

none

Do not read any more lines from the script file, until this Q-VET run is complete.

time

Specify the amount of time Q-VET waits before reading the next line of the script file. The section Time describes the *time* argument.

Example

```
# Select devices "file".
select devices file
# Set runtime to 30 minutes.
set run 30
# Start run.
start
# Wait for run to complete.
wait
# Display run summary.
show summary
# Duplicate the process.
duplicate process last
# Drop the new process.
drop process last
# Start the run.
start
# Wait 15 minutes.
wait 15
# Add the new process.
add process last
# Wait for run to complete.
wait
# Display run summary.
show summary
```

write errorlog

Description

Writes a text string into your operating system's error log.

Format

```
write errorlog "text-string"
```

Arguments

"text-string"

Write the text string to the operating system error log. If the operating system includes this feature, the text string is placed in the error log.

Restrictions

- None concerning execution state: you may issue this command while Q-VET is operating in the setup, active, or suspend execution state.
- This command works only if the operating system under which Q-VET runs includes an error log facility and you are running Q-VET as superuser. If the operating system includes this feature, the text string is placed in the error log.
- You must have previously set error logging with the **enable logging errors** command.
- Error logging is enabled by default, if you have the appropriate user privileges.

Example

```
Q-VET_suspend> write errorlog "Multiple seek errors on -
          drive #2 called system manager 2 PM PST"
```

A
Error Messages

The Q-VET manager and exercisers communicate with you by displaying messages. These inform you of progress,request information, and signal errors.

This appendix lists and, if necessary, explains Q-VET error messages. When these messages require user action, the explanation also indicates the necessary steps to take.

Note

All errors are logged to the system error log as well as the run display or console.

This appendix covers the following topics:

- Q-VET manager message format
- Q-VET manager error messages
- Q-VET exerciser error message format and types
- Q-VET exerciser error messages

Q-VET Manager Message Format

Q-VET manager messages are displayed in pop-up message boxes for the DECwindows Motif interface and in-line for the command interface. Most messages are self-explanatory; therefore, the following section lists only those which need more explanation, require user action, or are internal errors.

Messages Beginning with ?

Most Q-VET manager messages are informational, and the Q-VET manager continues executing after it displays them. When the Q-VET manager encounters an error, it prefaces the associated message with a question mark (?). Generally the Q-VET manager does not process the command that caused the error.

Messages Containing Formatting Directives

The messages listed below often contain formatting directives---that is, lowercase letters preceded by a "%". Q-VET fills in these directives with the associated values before displaying the message. Table A-1 lists the directives and their meaning.

Table A-1 Formatted Message Output

Formatting Directive	**Meaning**
%d	Decimal number
%o	Octal number
%s	Character string
%x	Hexadecimal number

Messages

`?Attempt to open message catalog failed in vet_mgr_database: %s.`

Meaning: Q-VET was unable to open the file containing its messages.

User Action: Check the protection of the messages file and the settings of the NLSPATH and LANG variables.

`?Can't continue unless run was stopped.`

Meaning: A **`continue`** command was typed while the current run was in progress.

User Action: Stop the run before issuing the **`continue`** command.

`?Device not known: %s`

Meaning: Q-VET was expecting a device name but found the displayed string instead.

User Action: Check for the correct command syntax or the correct device name in the command.

`?Installation timed out while installing file: %s`

Meaning: The Q-VET manager received no response within a specified amount of time when installing a file in the program database. This indicates that the file may not be in an appropriate format for a Q-VET program.

User Action: Rebuild the file in an appropriate format (see the *Q-VET Programmer's Guide*) or use the **`foreign`** command to run it under Q-VET.

`%s is an unknown host.`

Meaning: Q-VET did not find the name of the remote host in the network database file.

User Action: If you are using the TCP/IPtransport. update `/etc/hosts` with required information about the remote host. If you are using DECnet, update the NCP database with required information about the remote host.

`Message not available.`

Meaning: Q-VET could not find a message in the message catalog.

User Action: Check the settings of the NLSPATH and LANG environment variables (see Chapter 2).

`?No matching system name for %s.`

Meaning: Q-VET did not find the system name that was entered in the configuration database. Either you entered an incorrect system name or the specified system was not connected to the configuration database.

User Action: If you entered an incorrect name, reenter the command using the name of a connected system. If the system was not connected, issue the **connect node** command specifying the unconnected system name and the transport.

`...[Process %d] group %s for device %s is hung. Killing  process.`

Meaning: The specified process has not maintained contact with the Q-VET manager. Often, the user has specified options resulting in Q-VET program operations that take longer than normal to complete, for example specifying a huge transfer size for the disk, file, or tape exerciser. If this error occurs, the Q-VET manager calls the termination procedure to stop the process.If you want a Q-VET program to perform operations that take more time than normal to complete, you must either disable timeouts for that run or increase the timeout interval.

User Action: If Q-VET unsuccessfully attempts to stop a process, you must issue system-level commands to stop it. For the next Q-VET run, either disable timeouts or increase the timeout interval.

`?Requested operation not allowed in %s state.`

Meaning: The previous command is invalid in the current Q-VET state.

User Action: Enter only commands that are valid for this state. To reenter the previous command, you must first enter a command that activates a state in which it is valid.

`?Too many processes selected. Maximum system process limit is %d.`

Meaning: No more devices may be selected for this run. Each device selection or duplication causes the creation of a process, and the system or user process limit has been exceeded.

User Action: To select specific devices, first deselect some processes.

`Unable to connect to remote node %s.`

Meaning: The local node was unable to connect to the remote node.

User Action: Perform the following actions, depending on the transport medium:

Transport	Action
DECnet	Make sure that the command script (VETDNET) ensuring Q-VET DECnet communication was updated and placed in the proper location (/usr/sbin). Make sure that the dnet_spawner daemon is executing.)
TCP/IP	Make sure that the remote node's server databases (/etc/services and /etc/inetd.conf) contain the proper server information for Q-VET. Make sure that the /etc/INETd daemon is executing.)

`Unable to get local INET address.`

Meaning: Using TCP/IP, the remote node was unable to get the INET address for the local node.

User Action: Update the /etc/hosts file to include the remote node's information.

`Unable to get remote INET address`

Meaning: Using TCP/IP, the local node was unable to get the INET address for the remote node.

User Action: Update the /etc/hosts file to include the remote node's information.

Manager Internal Errors

The following is a list of internal manager errors. These seldom require user action. Those listed as programmer errors are caused by errors in a Q-VET program.

`?Bad database file.`

`?VET manager - Pipe read error. Errno = %d.`

`?VET manager - Pipe write error. Errno = %d.`

`?Error closing file: %s`

`?Error creating output pipe.`

`?Error creating pipe.`

`?Error from fcntl, errno = %d.`

`?Error in options procedure, terminating process %d.`

Meaning: Programmer error.

`?Group descriptor table not terminated, or too big.`

Meaning: Programmer error.

`?Invalid sizer packet received from node %s`

```
?Invalid status returned by process %d: , group %s, test %d,
subtest %d
```

Meaning: Programmer error.

`?Invalid subprocess list number: %d.`

Meaning: The specified number does not have a corresponding number on the internal subprocess list maintained by Q-VET.

`?Options list not terminated.`

Meaning: Programmer error.

`?Process %d made invalid call to vet_end_display.`

Meaning: Programmer error.

`?PROCESS%d - Pipe read error. Errno = %d.`

`?PROCESS%d - Pipe write error. Errno = %d.`

`?Process %d still owns the display.`

`?Received an illegal message from remote node %s.`

Meaning: Q-VET received an unrecognizable message from the specified remote node.

`?Received unexpected signal`

`?Response packet overflow in %s.`

Meaning: A response packet sent to Q-VET was too large.

`?Software error! , FILE = %s, LINE = %d, PROCEDURE = %s`

Meaning: Programmer error. The formatting directives give the error's location.

`?Supported device list not terminated, or too big.`

Meaning: Programmer error.

`?Test descriptor table not terminated.`

Meaning: Programmer error.

`Unable to create and bind the device socket.`

Meaning: Programmer error.

Exerciser Error Message Format and Types

Exerciser error messages have the same format whether Q-VET is operating in DECwindows Motif or command mode. These messages are displayed as part of the Q-VET manager's run information. The following is a sample exerciser error message with an explanation of its format:

```
*** HARD ERROR ❶ 1 ❷ from process 1 ❸, group exer ❹ for device memory ❺***

File vet_exer_memory.exe ❻, test 1 ❼, subtest 1 ❽, Tue Jan 28 11:03:36 1992❾

Did not read expected data. ❿

*** End of error report from process 1 ***
```

❶ The error type

❷ The number of the error within that type

❸ The number of the process that detected the error

❹ The name of the test group that was executing

❺ The name of the device that the test group was testing

❻ The name of the program file that was executing

❼ The number of the test that was executing

❽ The number of the subtest that was executing

❾ The date and time that the error occurred

❿ A description of the error

Q-VET reports six types of error messages, as follows:

- User errors - These occur when you enter invalid input, for example, an incorrect process option.

```
*** USER ERROR 1 from process 1, group exer for device file ***
File vet_exer_file, test 0, subtest 0 -Fri Jun 14 10:03:20 1991
Bad start_block number.
*** End of error report from process 1 ***
```

- Device Setup errors - These occur before a test runs if the Q-VET program you are running finds that the hardware is improperly set up, for example, when a device is off line or there is insufficient disk space to perform the tests.

```
*** SETUP ERROR 1 from process 1, group exer for device file ***
File vet_exer_file, test 1, subtest 1, Fri Jun 14 10:24:52 1991
Insufficient disk space.
*** End of error report from process 1 ***
```

- Device soft errors - These occur when a Q-VET program encounters device errors from which it is possible to recover. The test can try to recover, for example, by restarting the operation.

```
***SOFT ERROR 1  from process 1, group exer for device /dev/rra1c ***
File disk_functional, test 2, subtest 5, Fri Jun 14 10:33:25 1991
Read-compare error...will retry operation.
*** End of error report from process 1 ***
```

- Device hard errors - These occur when a Q-VET program encounters errors from which it is impossible to recover, such as a disk seek error. Repeated attempts to complete an operation that continually generates a soft error end in a Q-VET program reporting a hard error.

- Device fatal errors-These errors are so serious that it is impossible to continue testing, for example, when a device does not respond to a repeated operation. After a Q-VET program reports a device fatal error, it stops testing.

```
*** FATAL ERROR 1 from process 1, group exer for device file ***
File vet_exer_file, test 1, subtest 1, Fri Jun 14 10:49:08 1991
Bad return status from LSEEK.
*** End of error report from process 1 ***
```

- Software fatal errors-These are catastrophic errors that occur during a Q-VET test program. These errors are unrelated to device testing but make further testing impossible. An example of a software fatal error is a test process that issues a system service call but receives a return status indicating no access rights. After a Q-VET program reports a software fatal error, it stops testing.

```
*** SOFTWARE ERROR 1 from process 1, group exer for device memory ***
File vet_exer_memory, test 1, subtest 1, Fri Jun 14 10:54:08 1991
Attempt to use bad pattern detected.
*** End of error report from process 1 ***
```

Exerciser Messages

The following messages appear as error descriptions in device exerciser error messages.

CPU Exerciser Messages

`Scalar binary operations test failed.`

`Scalar conversions test failed floating point conversions.`

`Scalar conversions test failed integer conversions.`

`Scalar floating point computations test failed.`

`Scalar integer computations test failed.`

Memory Exerciser Message

`Attempt to use bad pattern detected.`

Meaning: An invalid pattern was specified.

User Action: Specify a valid pattern.

`Bad memory_to_allocate , min_segment_size, max_segment_size combination.`

Meaning: An invalid combination of exerciser options was specified.

User Action: Use different option values.

`Can't allocate %d byte segment.`

Meaning: No free memory is available to allocate this segment.

`First mismatch at byte %d - Expected: %2.2x, Actual: %2.2x.`

Disk Exerciser Messages

`Expected value: %d, Actual value: %d`

`Failure status returned from CLOSE.`

`Failure status returned from LSEEK.`

`Failure status returned from OPEN.`

`Failure status returned from READ`

`Failure status returned from WRITE.`

`First mismatch: Logical  block %d, byte %d Expected: %2.2x, Actual: %2.2x`

`Invalid start block, end block, step combination.`

Meaning: Bad options were specified. End block must be greater than start block.

User Action: Use different option values.

File Exerciser Messages

`Expected value: %d, Actual value: %d`

`Failure status returned from CLOSE.`

`Failure status returned from LSEEK.`

`Failure status returned from OPEN.`

`Failure status returned from READ`

`Failure status returned from UNLINK`

`Failure status returned from WRITE.`

`First mismatch: Logical  block %d, byte %d Expected: %2.2x, Actual: %2.2x`

`Insufficient disk space.`

`Invalid start block, end block, step combination.`

Meaning: Bad options were specified. End block must be greater than start block.

User Action: Use different option values.

Tape Exerciser Messages

`End of tape (EOT) was detected.`

Meaning: An unexpected end-of-tape (EOT) was detected during a write operation.

`Error assigning an I/O channel.`

Meaning: The system detected an error when assigning a channel for the tape device. A system call error may have generated this condition or the system may be set up incorrectly for running Q-VET on remote nodes.

User Action:

- Verify that another user has not mounted the same tape device.
- Ensure that the network software (DECnet or TCP/IP) is present and operating.
- Verify that Q-VET is correctly installed in the remote node.
- Verify that the remote and local nodes were properly set up for running Q-VET.

`Error deassigning an I/O channel.`

Meaning: The system detected an error when deassigning the I/O channel for the tape device. Remote execution of Q-VET may have terminated before the local node deassigned the channel.

User Action: Verify that execution of Q-VET terminated on the remote node.

`Error dismounting tape.`

Meaning: The system detected an error when trying to dismount a tape.

User Action: Try using a shell-level command to dismount the tape after the Q-VET session has terminated.

`Error rewinding tape.`

`Error spacing backward %d files.`

`Error spacing backward %d records.`

`Error spacing forward %d files.`

`Error spacing forward %d records on device %s.`

`Error writing a filemark.`

`Expected value: %d, Actual value: %d`

`Failure status returned from CLOSE.`

`Failure status returned from OPEN.`

`Failure status returned from READ`

`Failure status returned from WRITE.`

`File header mismatch detected.`

Meaning: After the exerciser performed a read operation, it checked the file header and found a mismatch between the file header and the current file count.

`Incorrect tape media used in drive`

User Action: Remove incorrect media and replace with correct media for drive type.

`Pattern header error detected.`

Meaning: After the tape exerciser performed a read operation, it performed a header check and detected an invalid pattern header.

`Record header mismatch detected.`

Meaning: After the exerciser performed a read operation, it made a record-header check and found a mismatch between the record header and the current record count.

`Tape is off-line, not inserted in drive, or the incorrect media type is being used.`

User Action: Terminate the exerciser, place the device on line, and restart the exerciser.

`Tape is WRITE LOCKED!`

User Action: Terminate the exerciser, unlock the write protection, and restart the exerciser.

`Unable to clear serious exception for device %s.`

Meaning: On DIGITAL UNIX systems, serious exceptions are generated on the tape device when testing of the tape exceeds the end-of-tape mark (EOT). An error occurred while the clearing of the exception was being performed.

`Unable to obtain device type information.`

`Unable to obtain status information.`

Network Exerciser Messages

`Expected value: %d, Actual value: %d`

`Node: %s - Asynchronous QIO read failed.`

`Node: %s - Can't assign channel (w/mail box).`

`Node: %s - Duplicate packet received (%d).`

`Node: %s - Echo service not known by target.`

`Node: %s - Error binding network endpoint (t_bind failed).`

`Node: %s - Error closing transport endpoint (t_close failed).`

`Node: %s - Error connecting to target.`

`Node: %s - Error connecting to target (t_connect failed).`

`Node: %s - Error creating network end point (t_open failed).`

`Node: %s - Error creating socket.`

`Node: %s - Error setting for asynchronous I/O (fcntl failed).`

`Node: %s - Error setting socket's ACL.`

`Packet %d never received.`

`Node: %s - Error setting socket's ACL.`

`Node: %s - Error unbinding transport endpoint (t_unbind failed).`

`Node: %s - Packet lost (echo never received).`

```
Node: %s - Packet size too large for server mirror object.
Node: %s - QIO failed trying to connect channel to target.
Node: %s - Received packet data mismatch.
Node: %s - Received packet out of order.
Node: %s - Received packet with invalid header data.
Node: %s - Recv call timed-out.
Node: %s - Recv returned bad status.
Node: %s - Recv returned bad transfer count.
Node: %s - Send returned bad status.
Node: %s - Send returned bad transfer count.
Node: %s - Unknown node name.
Packet byte %d - Expected value: %2.2x, Actual value: %2.2x.
Packet %d never received. Selected transport not available.
Status from lib$asn_wth_mbx() = %d.
Status returned (t_errno) : %d (%s).
XTI interface does not support DECnet.
```

Printer Exerciser Messages

```
A queue name was not specified. Please supply a queue name.
A queue has not been specified. Due to user's response, testing is
not being performed.
Error opening file %s.
Error writing to file %s.
Error accessing the file %s. Errno = %d
Error printing the file %s. Errno = %d
Error printing the file %s. Error code = %d
```

Terminal Exerciser Messages

```
Unknown
Error getting screen size.
Error opening device %s.
Error writing to device %s.
```

```
The device %s is allocated to another process.
The device %s is an invalid device name.
The device %s does not exist on the host system.
The device %s is on a remote system.
The device %s does not have proper access privileges.
No I/O channel is available for assignment.
```

Video Exerciser Messages

```
Invalid time specified. Entry was %d sec. (Range is 2 to 360 sec.)
Invalid Top position. Entry was %d sec. (Range is 0 to %d.)
Invalid Left position. Entry was %d sec. (Range is 0 to %d.)
Invalid Height. Entry was %d sec. (Range is 200 to %d.)
Invalid Width. Entry was %d sec. (Range is 486 to %d.)
User declared: Video %s failed.
User closed window before tests finished.
```

B
Generic Exerciser Information

The Q-VET **exer** group and **qual** group are comprehensive sets of exercisers targeted at system I/O and at processor and system testing. The exercisers execute in the Q-VET environment to perform exercise-oriented testing of system hardware and hardware interface software (I/O subsystems).

Each exerciser supports various run-time options that allow you to customize it. After you have specified values for options or taken their defaults, Q-VET passes these values to the exerciser when it is started.

Options generally include data patterns that are written to and read from devices in order to test them; options always include error-check levels, that is, the level of detail written in error reports.

Each exerciser generates a summary display if you request one. The display consists of the following general information:

- Exerciser characterization---The exerciser's run-time options.
- Time data---Start time, summary update time, elapsed time, and CPU time used.
- Operational statistics---Operations completed and performance-related data, according to the exerciser's functions and content.
- Error data---Error entries describing the first two errors and last occurring error; each error entry includes the error designation and the system time and date of the error.

The following sections list generic exerciser options, test patterns where applicable, and error-check levels, and describe summary displays.

CPU Exerciser Information

The following sections list the CPU exerciser option and error-check levels, and describe its summary display.

CPU Exerciser Options

Table B-1 lists the generic CPU exerciser options.

Table B-1 CPU Exerciser Option

Option Name	Description
`prime_limit`	Maximum range of prime numbers.
`error_check_level`	Error-check level, which may be set to 1, 2, or 3.
`go_delay seconds`	Startup delay 0-15 seconds default
`delete_tmp_log`	Delete or save temp files on success
`cpu_affinity`	ID number of the cpu which the created subprocess is to run on.
`cp_iterations` `(Windows NT only)`	Co-processor HCT iterations per pass, default 500.
`cp_threads` `(Windows NT only)`	Co-processor HCT process threads, default 17.

CPU Exerciser Error-Check Levels

CPU exerciser error-check levels are as follows:

- Level 1 results in no error checking.
- Levels 2 and 3 cause results to be checked. Q-VET reports invalid results as Device Hard Errors.

CPU Exerciser Summary Display

The CPU exerciser display consists only of general information.

Memory Exerciser Information

The following sections list the memory exerciser options, test patterns, and error-check levels, and describe its summary display.

Memory Exerciser Options

Table B-2 lists generic memory exerciser options in the order in which the exerciser prompts you for them.

Table B-2 Memory Exerciser Options

Option Name	Description
maximum_memory	Total virtual memory to test in bytes: once the total of segments allocated reaches this value, the exerciser begins decreasing memory usage until no memory is allocated; this constitutes one test pass. The default for test 1 is 50% of physical memory. Setting **maximum_memory** will change the default memory allocation for all memory tests. This can be overridden by setting the memory allocation for each test individually, i.e. **xma2_pages, memx_target_size**, and **gray_pages**.
min_segment_size	Minimum segment size in bytes: minimum segment size must be greater than 0 and less than or equal to maximum memory size; to force a fixed segment size, set minimum segment size equal to maximum segment size.
max_segment_size	Maximum segment size in bytes: maximum segment size must be less than or equal to maximum memory; to force a fixed segment size, set maximum segment size equal to minimum segment size.
pattern	Sets a data pattern to be used for writing. Appendix F lists supported test patterns.
enable_writes	When clear, this option disables data pattern write operations on getting a segment. The test then consists only of allocating and deallocating memory segments. If write operations are disabled, data checking is not performed. Return status from allocating and deallocating segments is always checked.
error_check_level	Error-check level, which may be set to 1, 2, or 3.
ignore_swap_space	Disables checking estimates for memory paging.
xma2_pages	Amount of memory in 512 byte pages. Default is 100% of physical memory.

xma2_iterations per pass	Default.
xma2_data_pattern	16 hex char. base data.
xma2_test_complement data	Y or N.
xma2_data_increment value	Value added to base data pattern, incremented each pass.
memx_time per pass (DIGITAL UNIX only)	In minutes.
memx_target_size (DIGITAL UNIX only)	% physical memory. Default is 100%.
gray_pages	Amount of memory in "memory page size". Default is 100% of physical.
gray_errors report limit	Error limit for gray code tests. Default is 10.
delete_tmp_log on success	Y or N.
cpu_affinity	ID number of the cpu which the created subprocess is to run on.

Memory Exerciser Test Patterns

Appendix F lists supported memory exerciser test patterns.

Memory Exerciser Error-Check Levels

Memory exerciser error-check levels are as follows:

- Level 1---Return status from a service call that allocates or disposes of memory segments is checked. An invalid return status generates a Device Fatal Error. There is no data-comparison checking.
- Level 2---In addition to level-1 checking, the beginning and end of each segment are verified for data-comparison errors immediately before the segment is deallocated (if data write operations are enabled). A data-comparison error generates a Device Hard Error.

- Level 3---In addition to level-2 checking, each segment is fully verified for data-comparison errors immediately before it is deallocated (if data write operations are enabled). A data-comparison error generates a Device Hard Error.

Memory Exerciser Summary Display

In addition to general information, the memory exerciser summary displays total segments allocated during the run.

Disk Exerciser Information

The following sections list the disk exerciser options, test patterns, and error-check levels, and describe its summary display.

Disk Exerciser Options

Table B-3 lists generic disk exerciser options, in the order in which the exerciser prompts you for them.

Table B-3 Disk Exerciser Options

Option Name	Description
`enable_writes`	Causes the exerciser to perform a disk write operation, delay, read from the same location, and compare data. By default no write operations are performed.
`warning_prompt`	Indicates whether the user will be alerted or not with a verification message before performing destructive operations. If set to "no" the warning is disabled; "yes" enables it. The default is “no”.
`reads_per_iteration` (test 1 and test 2)	The number of read operations performed on each iteration: each read and comparison operation counts as one read operation. To disable read operations, set this option to 0.
`block_size` (test 1)	The transfer or I/O size in bytes. This value determines the logical block size and must be a power of 2 value between 512 bytes and 128 Kbytes. The default block size is 512 bytes.
`start_block` (test 1 and test 2)	The first block number of a range of logical blocks to be tested.
`end_block` (test 1 and test 2)	The last block number of a range of logical blocks to be tested. The default end block is the last disk block.

`step` (test 1 and test 2)	The step value for seek operations (in logical blocks). Step values may be positive, negative, or random. A value of 0 signifies random, which is the default. To force a step value of 0, set the start_block value equal to the end_block value.
`iterations` (test 1 and test 2)	Specify number of iterations to perform. One test iteration consists of the following: Perform a disk write operation Delay Perform extra number of seek/read/delays Seek to previously written block, read and compare data Seek to new location Delay Specifying 0 iterations will cause the exerciser to test all blocks from the start_block value to the end_block value using the step value specified, for each pass of the disk exerciser. If the difference between the two values is less than 20 and a random step value is chosen, then it will use a step value of 1 or -1, whichever is more appropriate. If the difference is greater than 20 and a random step size is issued, then at least 20 iterations(blocks]) will be performed. The default number of iterations is 50,000 for hard drives, 200 for CD-ROMs, and floppy drives. This value is divided by 10 for test 2.
`delay` (in msec)	Specifies the delay between any two consecutive read or write operations, as measured in {CLK_TCK}ths of a second. Used to set the duty cycle; “0” is 100%. The default delay is 0 for hard drives, 1000 for CD-ROMs, and 120 for floppy drives.
`pattern`	Sets a data pattern to be used for writing. A data header precedes the test data block, regardless of the pattern. If the I/O size is less than the header size, the header is excluded (disabling some error-check ability). Appendix F lists supported test patterns.
`error_check_level`	Sets the error-check level to 1, 2, or 3.

`read_only_verify`	Enables data checking when an exerciser is performing read-only. (By default, data checks are performed only when write operations are enabled.) Use this option to read only data written with the same key option (if random patterns and/or random seeking is performed). This option has no effect if write operations are enabled.
`key`	Sets the randomizing key used to generate random seek/pattern values. A random key is chosen if not specified.
`delete_temp_logs`	Delete or save temporary files.
`cpu_affinity`	ID number of the cpu which the created subprocess is to run on.
`caching` (Windows NT only)	Caching and buffering enable. The default is enabled.
`hct_threads` (Windows NT only)	Process threads to test against the drive. The default is based on system configuration.
`hct_time` (Windows NT only)	Stress hct run time. The default is based on drive type.

Note

Running multiple disk exercisers that are writing to the same block range on the same device may result in invalid data comparison errors.

Disk Exerciser Test Patterns

Appendix F lists supported disk exerciser test patterns.

The written test data include a header which contains the following:

- PID of the exerciser process
- Logical block number
- Pattern
- Random offset used for random pattern (0 if not random pattern)

If you request a block size that is too small to contain the header, the header is not included in the test data.

Disk Exerciser Error-Check Levels

Disk exerciser error-check levels are as follows:

- Level 1---The following checks are made and reported as a Device Fatal Error:

  Unable to open device

  Bad return status from system close and lseek calls

- Level 2---Same as level 1, plus the following Device Hard Errors:

  Bad return status from system read and write operation calls

  Invalid data header

- Level 3---Same as level 2, plus full data verification. A data mismatch error is reported as a Device Hard Error.

Each byte in the block of data that was written is read back and verified against what was written. This requires large amounts of processor overhead and reduces I/O throughput, but improves error detection.

Disk Exerciser Summary Display

In addition to general information, the disk exerciser summary displays the following statistics:

- Iterations completed
- Total write operations
- Total read operations
- Total seek operations
- Total bytes written: (total write operations]) * (I/O size)
- Total bytes read: (total read operations) * (I/O size)
- Average I/O rate: (total I/O operations) / (elapsed system time)
- Average data rate: (total bytes transferred) / (elapsed system time)

File Exerciser Information

The following sections list the file exerciser options, test patterns, and error-check levels, and describe its summary display.

File Exerciser Options

Table B-4 lists generic file exerciser options in the order in which the exerciser prompts you for them.

Table B-4 File Exerciser Options

Option Name	Description
`file_name`	Sets the name of the temporary work file. If an existing file name is specified, the file is not deleted at completion. If the file name is a directory name, a temporary work file with a default name will be created in the named directory and is deleted by default at completion.
`enable_writes`	When clear, the exerciser will *not* write to a file, then delay, read, and compare. Writes are enabled by default.
`reads_per_iteration`	Number of read operations performed on each iteration: each read operation and comparison of a previous write operation counts as one read operation. To disable read operations, set to 0. The default number of reads_per_iteration is 1.
`block_size`	The transfer or I/O size in bytes. This value determines the logical block size.The default size is 512 bytes. Too large a block size causes Q-VET to time out. The actual point of failure varies according to such things as system I/O rate and system load. Block size should not exceed 20,000 bytes, unless you are certain that Q-VET will not time out.
`start_block`	Specifies the first block number of a range of logical blocks to be tested. Must be greater than or equal to 0. The default start_block is block 0.
`end_block`	Specifies the last block number of a range of logical blocks to be tested. The default size is 499 (500 total).
`step`	Step value for seeks (in logical blocks). Step values may be positive, negative, or random. A step value of 0 signifies random. This value is the default. To force a step value of 0, set the start block value equal to the end block value.

iterations	Specifies how many iterations to perform. One test iteration consists of the following: Perform a file write operation Delay Perform extra number of seek/read/delays Seek to previously written block, read and compare data Seek to new file location Delay Specifying 0 iterations causes the test to run forever. A test running forever can still end if the overall exerciser session run time (set by the user) is exceeded. The default number of iterations is 1000.
delay	Specifies the delay between any two consecutive read or write operations, as measured in {CLK_TCK}ths of a second. The default delay is 0.
pattern	Sets a data pattern to be used for writing. A data header precedes the test data block, regardless of the pattern. If the I/O size is less than the header size, the header is excluded (disabling some error-check ability). Appendix F lists supported test patterns.
error_check_level	Sets the error-check level to 1, 2, or 3.
read_only_verify	Enables data checking when an exerciser is performing read-only. (By default, data checks are performed only when write operations are enabled.) Use this option to read only data written with the same key option (if random patterns and/or random seeking is performed). This option has no effect if write operations are enabled.
key	Sets the randomizing key used to generate random seek/pattern values. A random key is chosen if not specified.
save_file	Disables normal deletion of the work file at completion. If an existing file is named, the file is never deleted.
cpu_affinity	ID number of the cpu which the created subprocess is to run on.

Note

If two exercisers are started that write to the same file, invalid data comparison errors may occur.

File Exerciser Test Patterns

Appendix F lists supported file exerciser test patterns.

The written test data include a header which contains the following:

- PID of the exerciser process
- Logical block number
- Pattern
- Random offset used for random pattern (0 if not random pattern)

If you request a block size that is too small to contain the header, the header is not included in the test data.

File Exerciser Error-Check Levels

File exerciser error-check levels are as follows:

- Level 1---The following checks are made and reported as a Device Fatal Error:
    - Unable to create exerciser work file
    - Unable to open device file
    - Insufficient disk space
    - Bad return status from system close and lseek calls
- Level 2---Same as level 1, plus the following Device Hard Errors:
    - Bad return status from system read and write operation calls
    - Invalid data header---the data header is not checked if write operations are disabled
    - Level 3---Same as level 2, plus full data verification. A data mismatch error is reported as a Device Hard Error.

Each byte in the block of data that is written is read back and verified against what was written. This requires large amounts of processor overhead and reduces I/O throughput, but improves error detection.

File Exerciser Summary Display

In addition to general information, the file exerciser summary displays the following:

- Iterations completed
- Total write operations
- Total read operations
- Total seek operation operations
- Total bytes written: (total write operations]) * (I/O size)
- Total bytes read: (total read operations) * (I/O size)
- Average I/O rate: (total I/O operations) / (elapsed system time)
- Average data rate: (total bytes transferred) / (elapsed system time)

Tape Exerciser Information

Q-VET contains two standard tape test groups---*exer* and *tape_all*. Group *exer* contains a records test (test 1) and a file test (test 2); group *tape_all*, which takes considerably more time to run, includes an EOT (end-of-tape) test. The options that the groups take are identical (see Tape Exerciser Options.

Issue the **show groups all for tape** command to learn which groups are supported for the tape device, for example:

```
VET_setup> show groups all for tape
           Group: exer
         Devices: tape
            File: vet_exer_tape.exe
           Tests: 1 - Tape Write/verify/space records Test
                  2 - Tape Write/verify/space file test

           Group: tape_all
         Devices: tape
            File: vet_exer_tape.exe
           Tests: 1 - Tape Write/verify/space records Test
                  2 - Tape Write/verify/space file test
                  3 - Physical EOT test

VET_setup>
```

The following sections list the tape exerciser options, test patterns, and error-check levels, and describe its summary display.

Tape Exerciser Options

Table B-5 lists generic tape exerciser options in the order in which each exerciser prompts you for them.

Table B-5 Tape Exerciser Options

Option Name	Description
`start_at_bot`	Starts each test at the beginning-of-tape mark (BOT) rather than where the last test ended.
`number_of_records`	Selects the number of records to write for tests 1 and 2.
`record_size`	Sets the record size for tests 1 and 2. Setting this option to -1 allows writing random length records.
`number_of_files`	Sets the number of files to write in test 2.
`pattern`	Sets a data pattern to be used for writing. Appendix F lists supported tape exerciser test patterns for tests 1 and 2.
`delay`	Specifies the delay between any two consecutive read or write operations, as measured in {CLK_TCK}ths of a second.
`enable_rewind`	When clear, this option disables the final rewind command issued when the exerciser completes.
`warning_prompt`	Indicates whether the user will be alerted or not with a verification message before performing destructive operations. If set to "no" the warning is disabled; "yes" enables it.
`cpu_affinity`	ID number of the cpu which the created subprocess is to run on.
`ignore_features` check (Windows NT only)	Drive is checked for supported tape movement features. The default is no.

Tape Exerciser Test Patterns

Appendix F lists supported tape exerciser test patterns for tests 1 and 2.

The written test data includes a header which contains the following:

- File number
- Record number

If you request a record size that is too small to contain the header, the header is not included in the test data.

Tape Exerciser Summary Display

In addition to general information, the tape exerciser summary displays the following:

- Total records written (excluding file marks)
- Total records read (excluding file marks)
- Total file marks written
- Total file marks read
- Total bytes transferred: sum of records written and read
- Average data rate: (total bytes transferred]) / (elapsed system time)

Network Exerciser Information

The following sections list the network exerciser options, test patterns, and error-check levels, and describe its summary display.

Network Exerciser Options

Table B-6 lists generic network exerciser optionsin the order in which the exerciser prompts you for them.

Table B-6 Network Exerciser Options

Option Name	Description
`transport`	Sets the connection transport to use (TCP, UDP, DECnet).
`nodes_list`	Specifies one or more destination nodes. Spaces separate the node names. You may include ACL information if required, for example, node1"my-name my-password."The default is to use the local node.
`single_step`	Enables single-step mode (TCP or DECnet only). Default is stream mode.
`packet_size`	Sets packet size. Setting to 0 disables write and read operations. The test consists of creating and closing network connections only.

`total_bytes`	Total bytes to transfer without reestablishing connection. Transfer bytes include write and read operations. Setting to 0 keeps the connection for the life of the exerciser.
`total_writes`	Number of write operations to perform before attempting any read operations (UDP only).
`packets`	Total packets to transfer before terminating. Total packets transferred is the sum of all connections' successful write and read operations. Specifying 0 packets causes the test to run forever. An exerciser running forever can still end if the run time (set by the user) is exceeded. If a packet size of 0 is selected, this option is interpreted as the total number of connections to establish.
`pattern`	Sets a data pattern to be used for writing. A data header precedes the test data block, regardless of the pattern. If the I/O size is less than the header size, the header is excluded and some error-check ability is disabled. Appendix F lists supported network exerciser test patterns.
`delay`	Specifies the delay between any two consecutive read or write operations, as measured in {CLK_TCK}ths of a second.
`error_check_level`	Error-check level, which may be set to 1, 2, or 3.
`cpu_affinity`	ID number of the cpu which the created subprocess is to run on.

Network Exerciser Test Patterns

Appendix F lists supported test patterns.

The written test data include a header which contains the following:

- PID of exerciser process---With the packet sequence number, verifies that the proper packet is returned
- Packet sequence number---Checks for lost or out-of-order return packets
- Pattern ID---Pattern number for this packet; verifies pattern bytes
- Send time stamp---Used to calculate round trip (echo]) time

If you request a packet size that is too small to contain the header, the header is not included in the test data.

Network Exerciser Error-Check Levels

Network exerciser error-check levels are as follows:

- Level 1---The following are checked and reported as a Device Fatal Error:

  Inability to map a specified node name to a valid address.

  Bad return status from any service call while trying to establish a network connection.

  Bad return status from any service call while trying to write or read to a network connection.

- Level 2---Same as level 1, plus the following are checked and reported as a Device Hard Error:

  Unexpected or unreadable packet header.

  The packet echo times out, if a packet does not echo within the timeout period (TCP and DECnet transports only). For each packet written, the echo packet is expected to return within a fixed amount of time.

  Duplicate packet echoed

- Level 3---Same as level 2, plus full data comparisons. A bad data comparison is reported as a Device Hard Error.

Network Exerciser Summary Display

In addition to general information, the network exerciser summary displays the following:

- Total number of network connections made
- Connection rate: (total connections) / (elapsed system time)
- Total write and read operations for each node
- I/O rate for each node: (total I/O operations) / (elapsed system time)
- Bandwidth for each node: (total bytes transferred) / (elapsed system time)
- Efficiency: (total bytes transferred for all nodes) / (total CPU time used)

Printer Exerciser Information

The following sections list the printer exerciser options and describe its summary display.

Printer Exerciser Options

Table B-7 lists generic printer exerciser options in the order in which the exerciser prompts you for them.

Table B-7 Printer Exerciser Options

Option Name	Description
`queue_name`	Specifies the name of the printer queue where the file is sent.
`queue_type`	Specifies a valid printer type of ASCII, PS, or OTHER. The default value is ASCII.
`page_width`	Specifies the width of the printout in characters. The default value is 80. This option only applies to the ASCII queue type with the default file.
`line_limit`	Specifies the number of lines that will be printed. The default value is 160. This option only applies to the ASCII queue type with the default file.
`file_name`	Sets the name of a file that contains data to be printed to the specified queue.
`cpu_affinity`	ID number of the cpu which the created subprocess is to run on.

Printer Exerciser Summary Display

In addition to general information, the printer exerciser summary displays the following:

- Queue name:
- Queue type:
- Page width:
- Line limit:
- File name:
- Time Data:
- Exerciser start-up time:

- Summary update time:
- Elapsed real system time: (in seconds)
- CPU Times - user and kernel: (in seconds)

Terminal Exerciser Information

The following sections list the terminal exerciser options, and describe its summary display.

Terminal Exerciser Options

Table B-8 lists generic terminal exerciser options in the order in which the exerciser prompts you for them.

Table B-8 Terminal Exerciser Options

Option Name	Description
`term_name_list`	Specifies one or more destination displays in a terminal list. Each terminal in the list should be separated by a space character.
`line_limit`	Specifies the number of rows the ripple pattern will be displayed.
`file_name`	Sets the name of a file that contains data to be displayed to the display. If a file is not specified, then the ripple pattern is displayed.
`display_width`	Specifies the number of characters to be displayed on each row.
`pause_screen`	Causes the exerciser to pause on each page waiting for an input. A "yes" or carriage-return input causes the next page to be displayed, otherwise the exerciser is terminated.
`iterations_per_test`	Number of times to repeat lines per pass.
`cpu_affinity`	ID number of the cpu which the created subprocess is to run on.

Terminal Exerciser Summary Display

In addition to general information, the terminal exerciser summary displays the following:

- Terminal list:
- Line limit:
- File name:
- Display width per term.:
- Pause screen:
- Time Data:
- Exerciser start-up time:
- Summary update time:
- Elapsed real system time: (in seconds)
- CPU Times - user, kernel: (in seconds)
- Terminal type(s):
- Number of lines displayed:
- Chars' displayed per term.:
- Avg. data rate per term.: (bytes/sec)

Video Exerciser Information

The following sections list the video exerciser options, and describe its summary display.

Video Exerciser Options

Table B-9 lists generic video exerciser.

Table B-9 Video Exerciser Options

Option Name	Description
`test_time`	The duration of each test can be set by the user. The minimum time is 2 seconds. The duration value is in seconds.
`top_window`	Specifies where to position the top edge of the window on the video screen. The position value is specified in pixels.
`left_window`	Specifies where to position the left edge of the window on the video screen. The position value is specified in pixels.

`height_window`	Specifies the vertical size of the window in pixels.
`width_window`	Specifies the horizontal size of the window in pixels.
`Log_color soft errors`	Y or N.
`mpc_bitmap`[12]	Y or N.
`mpc_stipple`	Y or N.
`mpc_lines`	Y or N.
`mpc_arc`	Y or N.
`mpc_segments`	Y or N.
`mpc_rectangle`	Y or N.
`mpc_points`	Y or N.
`mpc_pattern`	Y or N.
`mpc_detached`	Y or N.
`mpc_error_limit`	Number of errors before reporting/stopping.
`screen_number`	0-6.
`cpu_affinity`	ID number of the cpu which the created subprocess is to run on.

Note

Mpc cannot be run with the X11perf or Do_it exercises.

Video Exerciser Summary Display

In addition to general information, the video exerciser summary displays the following:

- Window position from top and left side:
- Window size for width and height:
- Time Data:
- Exerciser start-up time:
- Summary update time:

[12] mcp_ options are available for DIGITAL UNIX and OpenVMS only.

- Elapsed real system time: (in seconds)
- CPU Times - user, kernel: (in seconds)
- Error Data:
- Video Test Errors:

C
Menu Entries and Command Equivalents

This Appendix shows menu entries and their corresponding Command interface equivalents. On Windows NT systems, there is also an equivalent toolbar icon for some pull-down menu selections.

Table C-1 File Menu

Menu Entry	**Function**	**Command**

Set Path	Displays the current Q-VET path list.
	
	Modifies the path list.
		`show path`
		
		`set path`

Write Error Log	Writes a text string to the operating system error log file.	**`write errorlog`**
Foreign	Selects for execution a program that was not designed specifically to run under Q-VET.	**`foreign`**
Exit	Terminates the Q-VET session on all nodes and returns control to the operating system.	**`exit or quit`**

Table C-2 Setup Menu

Menu Entry	Function	Command

Select Devices	Selects devices for testing.	`select devices`
		`select groups`

Connect Node	Connects a node.	`connect node`
Connect Device	Adds a device's definition to the configuration database.	`connect devices`

Disconnect	Removes a node or device from the configuration database.	`disconnect devices`
		`disconnect nodes`

Duplicate	Creates one or more duplicate copies of processes.	`duplicate process`
Deselect	Deselects a process or processes.	`deselect processes`
Drop	Drops processes from the current run.	`drop processes`
Add	Adds processes to the current run.	`add processes`
Change Group	Changes the group for a process.	`select groups`

Show/Modify Parameters	Allows you to examine and change the tests, run time, pass count, and other options of a selected process.	`select options`
		`select tests`
		`set passcount`
		`set runtime`

Load	Creates a default system load.	`load`
Save Setup	Saves the current configuration displayed in Q-VET's Device Work Areas and the settings of Q-VET's process options.	no equivalent

Setup Menu (cont.)

Menu Entry	Function	Command
Restore Setup	Restores a Q-VET setup state that you have saved with Save Setup.	no equivalent
Clear Setup	Clears the contents of the Device Work Area and the Process Work Area. All process definitions are removed.	`deselect devices all` `deselect processes all` `disconnect nodes all`

Table C-3 Windows Menu

Menu Entry	Function	Command
Cascade	Arranges all MDI Child Windows in a cascade format.	no equivalent
Tile Horizontally	Arranges all MDI Child Windows in a horizontal format (they are wide rather than tall).	no equivalent
Tile Vertically	Arranges all MDI Child Windows in a vertical format (they are tall rather than wide).	no equivalent
Arrange Icons	Arranges all minimized MDI Child Windows.	no equivalent
Close Active	Closes the currently selected MDI Child Windows.	no equivalent
Device Work Area	Displays the nodes and devices in the configuration database in an MDI Child Window.	`show devices`
Process Work Area	Displays the current subprocesses that have been defined in an MDI Child Window.	`show processes`

Windows Menu (cont.)

Menu Entry	Function	Command
Run Display	Displays output of current run in an MDI Child Window.	Standard Q-VET output after the **`start`** command is issued
Summary Display	Displays a summary of the current or most recent Q-VET run in an MDI Child Window.	**`show summary`**
Summary Filter...	Limits the Summary Display to a specified device or process list.	**`show summary for..`**
Groups...	Displays all tests currently installed in the Q-VET Database.	**`show groups all`**
Command Line...	Allows you to enter Q-VET commands.	Q-VET prompt in command line mode
Active Windows List	Displays a list of active windows. The window that has focus is displayed with a check mark. You can change the focus to another window by clicking on that window in the list.	no equivalent

Table C-4 Run Menu

Menu Entry	Function	Command
Start All	Starts a Q-VET run.	**`start`**
Stop All	Temporarily stops the execution of Q-VET processes.	**`stop`** [Ctrl/C]
Continue All	Resumes execution of suspended processes.	**`continue`**
Terminate All	Permanently terminates suspended processes.	**`terminate`**

Continue	Resumes execution of specified processes.	**`continue processes`**
		`continue devices`

Run Menu (cont.)

Menu Entry	Function	Command

Stop	Temporarily stops the execution of specified processes.	`stop processes`
		`stop devices`

Terminate	Permanently terminates specified processes.	`terminate processes`
		`terminate devices`

Table C-5 Scripts Menu

Menu Entry	Function	Command
Execute Script Buffer	Executes the contents of the script buffer.	**`execute`**
Execute File...	Executes the contents of a script file.	**`execute`** *filename*
Edit Script Buffer...	Opens the script buffer for editing.	**`edit`**
Edit File...	Opens a new or existing script file for editing.	**`edit`** *filename*

Table C-6 Options Menu

Menu Entry	Function	Command
Set Run Time...	Sets the run time of all Q-VET processes.	**`set runtime`**
Set Pass Count	Sets the pass count for all Q-VET processes.	**`set passcount`**
Set Timeout	Sets the time interval Q-VET uses to determine if processes are “hung”.	**`enable timeout`** **`set timeout`**
Set Affinity...	Sets the default processor affinity to be used when defining a subprocess.	**`set affinity cpu_id`**
Set Error_Threshold	Sets the number of errors that are allowed for the overall run or for particular process(s).	**`set error_threshold`**
Other Options...	Sets parameters that control a Q-VET run.	See following table
The following options, displayed in the Other Options dialog box, are included for completeness:		
Error Display	Enables a level of error-reporting detail.	**`enable error_display`**
Trace Display	Enables a type of trace display.	**`enable trace`**

Options Menu (cont.)

Menu Entry	Function	Command
Stop on...	Stops on an error type.	`enable stop`
Loop on...	Creates an execution loop on an error.	`enable loop`
Mode	Selects any combination of run-time modes.	`enable mode`
Execution	Controls execution type - serial or parallel.	`set execution`
Error Logging	Enables or disables Q-VET's writing messages to the system event log, during execution of Q-VET runs.	`enable/disable logging error`

Table C-7 Maintenance Menu

Menu Entry	Function	Command

Install/Remove Files...	Displays the contents of Q-VET's program database.
	
	Allows you to add new files to the database and delete existing ones.
		`show installed`
		
		`install`
		
		`remove`

Set Default Group...	Designates a test group as the default of a device type.	`select default`
Accept	Runs acceptance procedures for new systems and devices.	`accept`

D
Available Device Tests

The process tests available for each device under the DIGITAL UNIX, OpenVMS, and Windows NT operating systens are described in Table D-1.

Table D-1 Available Tests for Devices

Device	Available Tests	DIGITAL UNIX OpenVMS	Windows NT
CPU	CPU Binary Operations Test	x	x
	CPU Scalar Integer Computations Test	x	x
	CPU Scalar Floating Point Computations Test	x	x
	CPU Scalar Data Conversions Test	x	x
	CPU Whetstone Synthetic Benchmark	x	x
	CPU Gausian Emination Matrix	x	x
	CPU Floating Point Screening	x	x
	CPU Double Precision Matrix Inversion	x	x
	CPU Prime Numbers Count/Check	x	x
	CPU CoProcessor (HCT)		x
Disk	Fixed Block Size Test	x	x

Device	Available Tests	DIGITAL UNIX OpenVMS	Windows NT
	Variable Block Size (512-64k)Test	x	x
	Sequential Diskx Test	x	
	64k Quadword Marker Test	x	x
	Disk Stress (HCT)		x
Tape	Tape Write/Verify/space records Test	x	x
	Tape Write/Verify/space file Test	x	x
	Tapex and mt Tests	x	
Video	Demo - Solid Rectangles	x	x
	Demo - Lines	x	x
	Demo - Open Rectangles	x	x
	Text Test	x	x
	Dot Matrix Test	x	x
	Line Matrix Test	x	x
	Bars Step Up/Down Test	x	x
	Multiburst Test		x
	Shaded Boxes Test	x	x
	Graphics Line Test	x	x
	Gray Level Test	x	x
	Bit Map Test	x	x
	Palette Test	x	x
	MPC Test	x	
File	File Unit/Read Test	x	x
Memory	Virtual Memory Test (Q-VET)	x	x
	MVB_loop Test	x	x
	Grey Code Tests	x	x
	memx Tests	x	
	XMA2(SITP)	x	x
	Virtual Memory Test (HCT)		x
Network	Network Echo Stress Test	x	x

Device	Available Tests	DIGITAL UNIX OpenVMS	Windows NT
x11exer	x11perf-all	x	
	ico	x	
	mpc - no data check	x	
Do_it		x	
Crash		x	x
Boot_loop		x	x
Jacketed		x	x

E
The Jacketed_Test Device

Q-Vet runs system and device tests as well as 'external' tests such as UNIX diskx, tapex, memx, x11perf, SITP xma2, and some of the HCTs. Other tests which are not a part of Q-VET can be run using the 'jacketed test control procedure' by selecting the Jacketed Test Device which has been included in the Device Work Area. The Jacketed Test is a pseudo-device based on the 'jacket test control' procedures that Q-Vet uses to launch and monitor the current 'external' tests.

The Jacketed Test allows other qual groups to take advantage of Q-Vet features and incorporate their own tests without the need to change Q-Vet source code. The Foreign command in the Q-VET Setup Menu also allows you to run non-Q-VET tests, but does not provide as many control features, error checking, or summary logging.

The 'device' will control the external test with options passed in to it via the standard device Q-Vet options parameters GUI or from the Q-Vet command line inteface (Q-Vet scripts). The test, of course, can not prompt for responses, it must be runable by a command string (the usual method for most tests).

Advantages

Using the Jacketed Test Device offers the following advantages.

- You have full control of the test and how its run, because you are supplying the command string that you would use to run the test from a console window or a batch script. You don't have to accept a default setup from Q-Vet.
- There is no Q-Vet source code to worry about. You make any changes you want to your test image or script and Q-Vet will do whatever you want, by supplying the appropriate options to the jacketed_test device(s).

Running a Jacketed Test

To run a Jacketed Test, select the jacketed_test device in the Device Work Area. Select the jacketed_test process which appears in the Process Work Area, and choose Show/Modify Parameters from the Setup Menu. The Show/Modify Parameters screen appears.

You may choose the following options from the Show/Modify Parameters screen. A minimum of only 1 argument is nneded for some tests. You can use the standard Q-Vet options to control run time, pass count, cpu affinity, error threshold, etc.

Option	**Description**
image	Name of test image or shell script. (w/o path)
cmd	The argument string that is used by the image; includes test options or whetever. example: -s -e -b 1024 (can be blank)
path	Directory where test resides. If not specified, uses Q-Vet default.
Log	Name of test output file that Q-Vet should check. If this option is blank, Q-Vet will redirect stderr and stdout to a log file which it will created, identify and check.
OK_check	A text string that Q-Vet should check for to determine (from the log file) that the test completed OK. If this option is blank Q-Vet will ignore it.
Bad_Check	A Text string for Q-Vet to check the log for which indicates a error or failure. If blank, Q-Vet will ignore it.
go_delay	Delay before start of test. Default 0 seconds. Use -1 for random of 0-30; max 1000 seconds. This will allow tests to run staggered or random if so needed.
halt_error	The default is Yes
delete_tmp_log	The default is to delete run logs on success (standard Q-Vet option)
affinity	The default is none (standard Q-Vet option)

Q-Vet will run the device (like all other devices) as follows.

1. Q-Vet processes the options, making sure they are valid.
2. Q-Vet sets up the device, in this case, by making sure that the image file exists in the path specified and that it is executable. It also creates a log file if necessary.
3. The 'image' is started with the supplied 'cmd' string (after the go_delay, if not zero). If a Logfile was not specified, Q-Vet will redirect the tests output to a file. This file name will be based on the image name, the path, and a process id (supplied by Q-Vet). A message will display the log name.
4. Q-Vet will then ensure that the test started ok and will monitor it in the system process table.
5. When the test completes, (or bombs out) Q-Vet will check the log file (based on the options) and report/log errors to the system log and the Q-Vet summary, just like all the other Q-Vet devices.
6. The Q-Vet summary will show the number of passes completed, time run, number of errors, etc.

Productivity Tips

The following are suggestions and tips for use of the Jacketed Test Device.

- You can have any number of Q-Vet 'jacketed_test' devices running at the same time, with or without the standard Q-Vet devices. This is of course limited by the test image itself, what resorces does it use, system tuning, etc. (these are things that you should consider for any test you are running anyway).
- Q-Vet will include the 'image' name in all reports and messages, so you can identify tests.
- You can run a 'jacked_test' device' and change it's run parameters from the Q-Vet GUI. It is not necessary to write any code or scripts.
- You can use the standard Q-Vet script editor from the Q-Vet GUI to create custom ways to start/stop your tests. You can also use any other editor to create a script. The command syntax to Q-Vet is exactly the same as for the other Q-Vet devices.
  script entry example:
  select group exer with options **`image "diskx" cmd "-b 1204"`**
- You can setup all your 'jacketed_test' devices, and then save the completed Q-Vet setup to a file that can be reused; just like all Q-Vet devices.

- You can write a simple 'setup' procedure which could build a Q-Vet script based on the system configuration, then run this Q-Vet script from the Q-Vet GUI or from the command line. This is what Q-Vet does for it's own internal and external tests (Qual Scripts).
- Another 'automated' feature that you could supply, would be to have a script start Q-Vet, and then allow Q-Vet to 'size' the system and build its scripts. Your procedure can then insert your "jacketed_test" devices into the scripts that Q-Vet creates. Thus you can use the existing Q-Vet 'Qual Scripts' menu to run your tests even more easily. Your script can also 'unload' one or more of the standard Q-Vet devices so that they could not be run.

F
Data Patterns

Data Patterns for Q-Vet Disk Test 1 and Test 2

Number	Name	Pattern
0	pattern_cycle_all = 0	
1	pattern_all_ones	/* 0xFF */
2	pattern_all_zeros	/* 00 */
3	pattern_checker	/* AA55 */
4	pattern_alt_checker	/* 55AA */
5	pattern_all_three	/* 33 (cont_freq_wc) */
6	pattern_all_a	/* AA */
7	pattern_all_five	/* 55 */
8	pattern_three_c	/* 33CC */
9	pattern_c_three	/* CC33 */
10	pattern_alt_one_zero	/* FF00 */
11	pattern_alt_zero_one	/* 00FF */
12	pattern_scan	/* 39C39C39 */
13	pattern_freq_burst_wc1	/* 2667 3333 */
14	pattern_freq_burst_wc2	/* 6667 3326 */
15	pattern_zero_three	/* 03 */
16	pattern_three_zero	/* 30 */
17	pattern_f_e	/* FEEF */
18	pattern_e_f	/* EFFE */
19	pattern_dibit_wc	/* (71C7 C71C 1C71) */

20	pattern_random	/* random byte data */

Data Format for Q-VET "64k Block Quadword Marker Test"

The following are features of the 64k Block Quadword Marker Test.

- 65536 byte xfers to random blocks.
- Unique data - assuming 64k xfer and "block number" based on xfer size.

Each Quadword in a transfer looks like this: [13]

0 0xFF FF 15 16 0xFF FF FF 39 40 0xFF FF 55 56 0xFF 63

qw_offset	block number	iteration	drive ID

Bits	**Description**
0:15	quadword offset into the 65535 byte block
16:39	block number (of size 65536)
40:55	iteration or I/O operation of current pass.
56:63	drive ID

For Windows NT the drive ID is base 36 and will look like this in the miscompare printout:

ascii drive name:	A	B	C	...	G	H	...	X	Y	Z
hex value:	A	B	C	...	10	11	...	21	22	23

For DIGITAL UNIX and OpenVMS, the ID is just Hex value of drive number. For Windows NT only, the floppy device is F0.

Use the DumpMedia program to read the actual data on the failing drive.

[13] Shown this way because miscompares are printed low to high byte

Data Header for Q-Vet Disk Test

The following data header applies to Q-Vet Disk Tests.

```
u32bit_int    exerciser_id;     /* process ID       */
s64bit_int    current_block;
u32bit_int    pattern;          /* current pattern   */
u32bit_int    random_offset;    /* within random pool */
u32bit_int    block_size;       /* current xfer size  */
```

Data Patterns for Q-Vet Memory Test (Test 1) .

Number	Name	Pattern
0	pattern_cycle_all = 0,	
1	pattern_all_FF,	/* 0xFF */
2	pattern_all_zeros,	/* 00 */
3	pattern_checker,	/* AA55 */
4	pattern_alt_checker,	/* 55AA */
5	pattern_all_three,	/* 3333 */
6	pattern_all_a,	/* AA */
7	pattern_all_five,	/* 55 */
8	pattern_three_c,	/* 33CC */
9	pattern_c_three,	/* CC33 */
10	pattern_alt_zero_one,	/* 00FF */
11	pattern_alt_one_zero,	/* FF00 */
12	pattern_two_zero_four,	/* 2004 */
13	pattern_a_zero_six,	/* A006 */
14	pattern_scan,	/* 39C3 */
15	pattern_zero_three,	/* 03 */
16	pattern_three_zero,	/* 30 */
17	pattern_f_e,	/* FEEF */
18	pattern_e_f,	/* EFFE */

19	pattern_all_eight,	/* 8888 */
20	pattern_all_four,	/* 4444 */
21	pattern_all_two,	/* 2222 */
22	pattern_all_one,	/* 1111 */
23	pattern_all_six,	/* 6666 */
24	pattern_all_nine,	/* 9999 */
25	pattern_random	/* random bytes

Data Patterns for Q-Vet Network Test.

Number	Name	Pattern
0	pattern_cycle_all = 0,	
1	pattern_all_ones,	/* 0xFF */
2	pattern_all_zeros,	/* 00 */
3	pattern_checker,	/* AA55 */
4	pattern_alt_checker,	/* 55AA */
5	pattern_all_three,	/* 33 */
6	pattern_all_a,	/* AA */
7	pattern_all_five,	/* 55 */
8	pattern_three_c,	/* 33CC */
9	pattern_c_three,	/* CC33 */
10	pattern_alt_one_zero,	/* FF00 */
11	pattern_alt_zero_one,	/* 00FF */
12	pattern_scan,	/* 39C3 */
13	pattern_freq_burst_wc1,	/* 2667 3333 */
14	pattern_freq_burst_wc2,	/* 6667 3326 */
15	pattern_random	/* random bytes */

Data Patterns for Q-Vet File Test .

Number	Name	Pattern
0	pattern_cycle_all = 0,	
1	pattern_random,	/* random bytes */
2	pattern_all_ones,	/* 0xFF */
3	pattern_all_zeros,	/* 00 */
4	pattern_cont_freq_wc,	/* 0x3333 */
5	pattern_freq_burst_wc1,	/* 2667 3333 */
6	pattern_freq_burst_wc2,	/* 6667 3326 */
7	pattern_dibit_wc,	/* 71C7 C71C 1C71 */
8	pattern_checker,	/* 0xAA55 */
9	pattern_alt_checker,	/* 0x55AA */
10	pattern_all_a,	/* 0xAA */
11	pattern_all_five,	/* 0x55 */
12	pattern_three_c,	/* 0x33CC */
13	pattern_c_three,	/* 0xCC33 */
14	pattern_scan	/* 0x39C3 */

Data Patterns for Tape Exerciser Tests .

Number	Name	Pattern
0	pattern_cycle_all	Cycle through all the following patterns in sequence (default)
1	pattern_random	Random data
2	pattern_all ones	All ones pattern: FFFF
3	pattern_all_zeros	All zeros pattern: 0000
4	pattern_alt_one_zero	Alternating bytes of all zeros and all ones: FF00

5	pattern_two_one_zero	Alternating two bytes of ones and two bytes of zeros: FFFF0000
6	pattern_four_one_zero	Alternating four bytes of ones and four bytes of zeros: FFFFFFFF00000000
7	pattern_three_one_zero	Alternating three bytes of ones and one byte of zeros: FFFFFF00

Glossary

Automated Mode

The mode that permits Q-VET to run under the control of a higher-level automated environment, such as an automated testing system used in the manufacturing process. Disabled by default.

Basic-level error report

A short statement identifying the kind of error being reported, such as a write-check error for a disk. Every error report contains a basic error message that it displays if an error is encountered.

Command file

See *Script file*.

Configuration database

Q-VET’s database of testable devices.

Device

For testing purposes, any subset of the system under test. It can be a particular device unit, a controller, an adapter, or a processor. It can also be a software-defined system subset, such as the "file system."

Device fatal errors

Errors that are so serious that it is impossible to continue testing. After the Q-VET program reports a device fatal error, it stops testing.

Device hard errors

Errors from which it is impossible for a Q-VET program to recover, such as a disk-seek error. Generally, repeated attempts to complete an operation that continually generates a soft error ends in the Q-VET program reporting a hard error.

Device setup errors

Occur before a test runs if the Q-VET program finds that the hardware is improperly set up, for example, when a device is off line or a loopback connector has not been attached.

Device soft errors

Occur when the Q-VET program encounters device errors from which it can recover.

Device under test

The portion of the system that is the target of the tests being run. The device under test need not be local to the system running Q-VET nor a piece of hardware.

Error reporting

Three levels of detail are reported:

Basic (see Basic-level error report)

Full (see Full-level error report)

Extended (see Extended-level error report)

Error types

Six types of errors are reported:

Device fatal errors (see Device fatal errors)

Device hard errors (see Device hard errors)

Device setup errors (see Device setup errors)

Device soft errors (see Device soft errors)

Software fatal errors (see Software fatal errors)

User errors (see User errors)

Execution state

The state in which the Q-VET operates; one of the following:

Active state

Setup state

Suspend state

Exerciser

A Q-VET program that is designed to generate large amounts of activity in a piece of system hardware or on the operating system.

Extended-level error report

Displays data when large amounts are available, such as the contents of a large data buffer. An error report need not contain this level of detail.

Full-level error report

Contains more information about an error than a *basic-level* message, such as the contents of all relevant device registers. An error report need not contain this level of detail.

Group

See *Test group*.

Interactive mode

The mode that tells Q-VET that a user is monitoring the Q-VET run and is available to respond to prompting messages. Enabled by default.

Loop

The repeated execution of a single subtest. If you enable "looping-on-error" and an error occurs, Q-VET repeatedly executes the subtest that has reported the error, but only up to the point at which the error is reported.

Modes

One of a number of possible states of operation. See *Run-time modes*.

Organization of tests

See *Test organization*.

Pass

The execution of all selected tests on a particular device one time.

Process

The basic entity scheduled by the system software. Q-VET executes each test group selected for a device as a separate process.

Process definition

A description of a process that Q-VET will create during a Verification Software run. The process definition contains all process-specific run-time parameters that the process uses when it executes.

Process number

Number assigned to a process. Process numbers are used as arguments to some commands. Q-VET process numbers are not the same as system process numbers, which identify processes existing outside of Q-VET.

Program database

Q-VET's database of test programs. Contains information about each test program known to Q-VET including:

File name

Supported devices

List of options

List of groups

List of tests for each group

Quick mode

A mode that shortens the time it takes a test group to complete a pass. Not all test groups provide quick mode. Disabled by default.

Run

The execution of all selected test groups in all processes for the selected number of times. A Q-VET run may consist of several passes.

Run-time modes

One of five modes:

*Automated (*see *Automated mode)*

*Interactive (*see *Interactive mode)*

*Quick (*see *Quick mode)*

*Truncate (*see *Truncate mode)*

*Verify (*see *Verify mode)*

Script file

A file containing commands that Q-VET can execute. Each line of the script is either a Q-VET command or the response to a user prompting message. The **execute** command starts the execution of a script file.

Sizing process

The process that finds the devices that are connected to a system and obtains device information needed by the Q-VET programs. By default, the Q-VET runs the sizing process to size the local system when you initiate a Q-VET session.

Software fatal errors

Catastrophic errors that occur during a Q-VET program. These errors are unrelated to device testing but make further testing impossible.

Subtest

A set of one or more procedures that test a device. Subtests are written to be members of a test and are not necessarily independent of each other. There is no limit to the number of subtests in each test. Subtests are numbered sequentially from one.

Subtests are the parts of a test on which Q-VET can *loop* when it encounters an error.

Test

A set of one or more device testing procedures or *subtests* that have have been grouped together. Each test in a test group is independent of the others. Its execution never depends on the previous execution of another test.

Test group

A set of one or more tests that form a complete testing function; the primary Q-VET testing entity. One test group is always the default group for the device.

Test organization

Organized into the following groups:

*Test groups (*see *Test group)*

*Tests (*see *Test)*

*Subtests (*see *Subtest)*

Truncate mode

The mode in which, after Q-VET reports a device error, passes of that process are executed only up to the subtest before the failed one.

User errors

Errors resulting from users' entering invalid values for process options.

Verify mode

The mode in which the commands contained in a script file are displayed as Q-VET reads them from the file. Disabled by default.

Verification Software run

See *Run*.

Index