Test Environment Toolkit Release Notes

Implementation for POSIX.1 Systems in ‘C’ and XPG3 Shell
by OSF, UNIX International and X/Open

Release 1.9 (03/09/92)

Preface

These release notes briefly describe Release 1.9 of the *Test Environment Toolkit* implementation for POSIX.1 systems in 'C' and XPG3 shell by OSF, UI and X/Open. They describe the prerequisites for running this toolkit, what is supplied in the toolkit, and how to build the toolkit. The release notes also list the operating system platforms on which the Test Environment Toolkit has been built and run.

Audience

This document is written for software engineers who will be porting their validation test suites into the Test Environment Toolkit.

Document Usage

This document is organised into the following chapters.

- Chapter 1 gives introductory information about the Test Environment Toolkit.
- Chapter 2 describes how to build the Test Environment Toolkit.
- Chapter 3 describes the changes made since the previous release.
- Chapter 4 describes extensions to the specification.

Related Documents

For additional information about the Test Environment Toolkit, refer to the following document:

- *Test Environment Toolkit: Architectural, Functional, and Interface Specification, Version 4.3*

Typographic and Keying Conventions

This document uses the following typographic conventions:

Bold	In formats and command descriptions, **bold** words or characters represent commands or keywords that you must use literally, including pathnames. In examples, information that you enter appears in **bold**.
Italics	*Italic words or characters in formats and command descriptions represent values that you must supply.*
`Typewriter font`	`Information that the system displays appears in this typeface. Examples also appear in this typeface.`

Problem Reporting

If you have problems with the TET send electronic mail to:

tet_support@xopen.co.uk

Include the following information in your mailing:

- Your name, company or organisation, and email address
- Hardware you are running on
- Operating system version TET is installed on

- TET version number
- Description of encountered problem, including how the problem occurred and how it can be reproduced
- A copy of the TET configuration file(s) for the affected modes
- A copy of the TET journal file
- A suggested fix for the problem, if possible

A template to use for reporting problems by email is provided in the file **doc/posix_c/err.template**.

1. Introducing the Test Environment Toolkit

The purpose of the Test Environment Toolkit (TET) is to provide a uniform framework, or test scaffold, into which test suites can be incorporated. By providing such a scaffold, test suites from different vendors can share a common interface, allowing for, among other things, ease of portability.

This implementation of the Test Environment Toolkit currently consists of:

- A test case controller providing support for the building, execution, and clean-up of test scenarios (**tcc**);
- A test suite installation tool (**tet_ins**);
- A report generator which opens a specified journal file, reads its contents, and delivers them to a named report treatment filter (**rpt**);
- a ‘C’ Application Programming Interface; and
- a shell Application Programming Interface.

Each Application Programming Interface (API) consists of a test case manager and an API library, which are used in test cases to handle interfacing between the test case and the test case controller.

This release does not include the Programmers’ Guide or Users’ Guide. These documents are currently being worked on by UNIX International, and will be distributed separately when they are ready.

Reference manual pages are supplied for **tcc**, **rpt**, and **tet_ins** in the directory *tet_root*/**doc/posix_c**. Refer to Chapter 6 of the TET specification for information on the Application Programming Interfaces.

In addition to the TET, you will also receive a copy of a simple demonstration TET test suite.

2. Building the Test Environment Toolkit

This chapter describes how to set up your system and build the test environment. Prior to building the test environment, you should check that the following requirements are met. The test environment will not build without them.

2.1 Test Environment Toolkit Requirements

The Test Environment Toolkit requires that your system's operating system be compliant with the following:

- POSIX.1[1] with either standard 'C' or common 'C' support.
- XPG3, Vol. 1[2] (for the **shell, find, cp, rm,** and **make** commands)

As of this writing, the Test Environment Toolkit has only been installed and tested on UNIX[3] type of systems. Note that this does not preclude its installation on other types of operating systems.

The TET provides a series of makefiles that can be used to build the toolkit in a UNIX type of environment. If you are not running a UNIX type of operating system, you will have to develop your own equivalent method of building TET.

2.2 Installed Platforms

This release of the TET has been installed and tested on the following operating system platforms (UNIX type of operating systems).

- HP-UX[4] version 7.0.
- ULTRIX[5] version 4.0.
- UNIX System V release 4.0
- SunOS[6] release 4.1

2.3 What is in the Test Environment Toolkit

This document assumes that your TET root directory is *tet_root*. When you specify your TET pathname, replace *tet_root* with your equivalent pathname. After you load the TET, you should have the following subdirectories on your system:

tet_root/**inc/posix_c**	POSIX 'C' header files for use by test suites, currently only **tet_api.h**.
tet_root/**src/posix_c**	Source directory for the TET tools and API, for POSIX 'C'.
tet_root/**src/posix_c/tools**	Source directory for the TET tools **tcc**, **rpt**, and **tet_ins**.

1. IEEE Std 1003.1-1990, *Portable Operating System Interface for Computer Environments*
2. *X/Open Portability Guide Issue 3, Volume 1: XSI Commands and Utilities*
3. UNIX is a registered trademark of UNIX System Laboratories, Inc. in the U.S. and other countries.
4. HP-UX is a registered trademark of Hewlett Packard Corporation.
5. ULTRIX, DEC, and DIGITAL are a registered trademarks of Digital Equipment Corporation.
6. SunOS is a trademark of SUN Microsystems, Inc.

tet_root/**src/posix_c/api**	Source directory for the TET Application Programming Interface for POSIX 'C'.
tet_root/**src/posix_c/inc**	Header files, for the TET POSIX 'C' sources, including the file **tet_jrnl.h**.
tet_root/**src/xpg3sh/api**	Source directory for the TET Application Programming Interface for XPG3 shell.
tet_root/**bin**	Executable tools for the TET, including **tcc**, **rpt**, and **tet_ins**.
tet_root/**lib/posix_c**	POSIX 'C' library files for use by test suites, including **libapi.a**, **tcm.o**, and **tcmchild.o**.
tet_root/**demo**	Directory containing the files that can generate a simple demonstration of a TET test suite.

2.4 Building the Environment

After the TET is loaded, you can build the Test Environment Toolkit, including the TET tools and the application programming interfaces, with the makefiles supplied. However, there is a makefile that needs to be edited before you can build the TET.

- *tet_root*/**src/posix_c/makefile**

This makefile calls other makefiles in *tet_root*/**src/posix_c/tools** and *tet_root*/**src/posix_c/api**. Edit the *tet_root*/**src/posix_c/makefile** and follow the instructions in the comments.

The TET 'C' source is written to POSIX.1 using common 'C' language, although prototypes for the API library functions are provided in a header file for use with ANSI 'C' compilers. One extension to POSIX.1 is required: the symbol NSIG. Many UNIX systems already define NSIG in **<signal.h>** as an extension to POSIX.1. In this case, NSIG can be made available by compiling with the appropriate feature test macro defined in addition to _POSIX_SOURCE.

The TCC catches all signals it can from 1 up to (NSIG-1). On some systems this can cause problems with non-standard signals, for example SIGCLD if it is supported separately from SIGCHLD. Also, with the default signal handling the TCC cannot be suspended and restarted using job control signals. Two variables can be set in **makefile** to alter the signal handling. One specifies a list of signal numbers to ignore, the other a list of signal numbers to leave alone. If you wish to use job control with the TCC, you should include the job control signals in SIG_LEAVE.

2.5 Building the Tools and 'C' API

1. To start, if you are not already there, change directory to *tet_root*/**src/posix_c**.

   cd *tet_root*/**src/posix_c**

2. Type **make**. This will build the three TET tools, **tcc, rpt**, and **tet_ins** as well as the 'C' API.

When you use **make**, an executable file for each TET tool component is placed in the directory *tet_root*/**bin**. The TET API files are generated and placed into the appropriate directories:

Name	Path	Description
tcm.o	*tet_root*/**lib/posix_c**	Test Case Manager **main()** routine. For processes to be executed by the **tcc**
tcmchild.o	*tet_root*/**lib/posix_c**	Executed processes **main()** routine. For processes to be executed by **tet_exec()**
libapi.a	*tet_root*/**lib/posix_c**	Library of API interfaces

In addition, the following files are also available for use with the Application Programming Interface.

Name	Path	Description
tet_api.h	*tet_root*/**inc/posix_c**	Header containing useful symbol definitions and declarations/prototypes for all API interfaces
llib-ltcm.c **llib-ltcmc.c**	*tet_root*/**src/posix_c/api**	Lint libraries defining the API interfaces. Use the one which corresponds to the object file in which **main()** is defined (**tcm.o** or **tcmchild.o**). They can be used directly or turned into libraries for use with **lint -l** by doing a **make LINTLIB** and installing the **.ln** files on the system.

2.6 Building the Shell API

1. To start, if you are not already there, change directory to *tet_root*/**src/xpg3sh/api**.

   cd *tet_root*/**src/xpg3sh/api**

2. Edit the **makefile** and change the lists of signal numbers to the correct values for your system.

3. Type **make**. This will install the shell API files **tcm.sh** and **tetapi.sh** in the directory *tet_root*/**lib/xpg3sh**.

2.7 Running the Test Environment Toolkit Demonstration

This section describes how to run the demonstration TET test suite in *tet_root*/**demo**. The given invocation will build the test suite, execute it, and clean it.

1. To start, if you are not already there, change directory to *tet_root*.

   cd *tet_root*

2. Type the following command invocation (be sure that *tet_root*/**bin** is in your PATH).

   tcc -bec demo

   As the demonstration executes, it displays the following messages:

   ```
   Journal file is: tet_root/demo/results/0001bec/journal
   We have not set TET_OUTPUT_CAPTURE=True so all normal stdin/stdout/stderr
   files are available.
   If we had set output capture, the results logged by the API would not
   be in the journal.
   But these lines would.
   ```

   You can then look in *tet_root*/**demo/results/0001bec/journal** to see the results of the demonstration.

3. Changes Since the Previous Release

3.1 'C' API

- An intermittent memory leak in tet_putenv() was fixed.
- Dummy arguments were added to the signal catching functions alrm() and sig_term() in tet_fork.c.

3.2 Shell API

- The tet_setcontext() function was altered to use a new method of obtaining context numbers so that it now changes the context number on each call.
- Improvements were made to signal handling code to avoid repetitious **trap** commands and other processing of signal numbers.

3.3 tcc

- Use of an uninitialised variable in process_line(), which could result in further executions being skipped after an execution failure, was corrected.
- The function getopt() was renamed to avoid possible clashes with versions provided on some systems.
- The locking code has been changed to ensure a previously obtained source lock is not left in place when an execution lock cannot be obtained.
- Result codes of type **Abort** are now correctly handled by the **tcc** so that it shuts down at the end of the test case.
- Type mismatches using NULL to represent an integer zero have been corrected.
- The resume and rerun options can now be used properly with build and clean modes, as described in the revised TET specification (version 4.4).
- The limit of 133 characters for a configuration file line has been greatly increased. Long configuration value lines in the journal file are truncated to 512 characters, and are followed by a warning message.
- If the timeout option is not used and the **tcc** encounters a lock, an error message is now generated both on stderr and in the journal file.
- Further general improvements have been made to error messages.
- A simple memory debugging system has been added, as a compile time option. As a result several memory leaks have been plugged.
- The execution results file reordering functions have been reorganised in order to help resolve one of the above mentioned memory leaks.
- The TET_SAVE_FILES variable is now obtained from the correct configuration file when more than one mode is selected.
- The error cleanup code has been changed to avoid the possibility of infinite recursion.
- The handling of SIGHUP, SIGINT and SIGQUIT has been changed so that they are only set to be caught if not inherited as ignored. This is to prevent the TCC from catching signals not meant for it if run in the background or under *nohup*.

3.4 tet_ins

- If the file **tet_code** does not exist, **tet_ins** no longer complains. The format of the file is still checked if it exists.

3.5 rpt

No changes.

4. Extensions to the Specification

The following features are implemented in addition to those required by the TET specification.

4.1 Environment Variables

You can override the value of TET_ROOT compiled into **tcc** by setting TET_ROOT in your shell environment before executing **tcc**. For example

- For the C shell, type a line like the following at the shell prompt:

  `setenv TET_ROOT` <*your TET_ROOT path*>

- Or, for the Bourne shell, type the following line at the shell prompt:

  `TET_ROOT=`<*your TET_ROOT path*>`;export TET_ROOT`

  By doing this, you can then refer to <*your TET_ROOT path*> as $TET_ROOT in the shell.

The environment variable TET_EXECUTE can be set in a similar fashion to that of the TET_ROOT, and allows you to specify a default value for the alternate execution directory that is used by the **tcc**. Note however that a command line supplied value for the alternate execution directory has a higher precedence than that supplied by the TET_EXECUTE environment variable.

During the execution of the **tcc** temporary files are usually created in unique sub_directories under a directory called **tet_tmp_dir**, which is located in the *tet_root*. Whilst every effort has been made to ensure that each sub-directory is removed as the **tcc** finishes, it is possible that an abnormal termination will mean that a sub-directory remains. Such directories should be removed to conserve disk space.

The user may specify an alternative temporary directory for use by **tcc** by setting the environment variable TET_TMP_DIR. This variable should be set in a similar fashion to the TET_ROOT variable discussed above. Performance problems due to network file systems can be helped by setting this variable to a directory on a local file system.

4.2 Extra Options

The following **tcc** command line options are provided in addition to those in the specification:

-l may be used to supply a one-off scenario line for execution without the need to create a scenario file.

-p enables progress reporting to standard output.

-y -n may be used to select parts of a scenario for processing. Test case execution lines are processed only if they match the specified search string (**-y**) or don't match it (**-n**).

See the manual page in *tet_root*/**doc/posix_c/tcc.1** for more details.