NIST

National Institute of Standards and Technology
Computer Systems Laboratory

NIST-PCTS:151-2
Installation and Testing Guide

October 1, 1995

NIST POSIX Certification Authority
National Institute of Standards and Technology
Computer Systems Laboratory
Gaithersburg, MD 20899

ABSTRACT

The NIST-PCTS:151-2, is a POSIX conformance test suite (PCTS) written by the National Institute of Standards and Technology. It tests system implementations for conformance to FIPS 151-2. FIPS 151-2 is based on the ISO/IEC 9945-1:1990 (Portable Operating Systems Interface(POSIX)— Part 1: System Application Program Interface (API) [C Language]) and the additional requirements restated in Appendix A of this document. The assertions tested by this PCTS are only those of the IEEE Std 2003.1-1992 (Tests Methods for Measuring Conformance to POSIX.1) that relate to FIPS 151-2.

This Installation Guide specifies the implementation resources needed and documents the procedures to be followed to install and execute this PCTS.

1. Introduction

The NIST-PCTS:151-2 is a POSIX.1 conformance test suite (PCTS) that verifies the compliance of an implementation to FIPS 151-2. It is based on the requirements of the IEEE Std 1003.3-1991 (Tests Methods for Measuring Conformance to POSIX) and IEEE Std 2003.1-1992 (Tests Methods for Measuring Conformance to POSIX.1). The NIST-PCTS:151-2 only tests the IEEE Std 2003.1-1992 assertions that are applicable to the requirements of FIPS 151-2.

To fully understand the limitations and requirements imposed on the NIST-PCTS:151-2, the documents IEEE Std 1003.3-1991, IEEE Std 2003.1-1992, ISO/IEC 9945-1:1990, and the FIPS 151-2 should be studied. This document assumes that the tester is knowledgeable with the terms and concepts used in these documents.

If you have questions, suggestions, suggested code, please send electronic mail to *151-2@sst.ncsl.nist.gov*. Other correspondence may be U.S. mailed to:

National Institute of Standards and Technology
Attn: NIST-PCTS:151-2
NIST/CSL (Code 872)
Bldg 225, Room B266
Gaithersburg, MD 20899

2. General

2.1 Notations

The specification of pathnames as used by this document are relative to the directory name NIST-PCTS. Therefore, a pathname of ./src/driver refers to the relative pathname NIST-PCTS/src/driver. The header files for this PCTS are denoted as <pathname_hdr>. The pathname of these files is NIST-PCTS/include/pathname_hdr.

The IEEE Std 2003.1-1992 will be referenced in this document as POSIX.3.1.

2.2 General Terms and Abbreviations

assertion
: A statement of functionality or behavior for a POSIX.1 *element* that is derived from the POSIX.3.1 standard being tested and that is true for a conforming POSIX.1 implementation.

assertion test
: The software or procedural methods that ascertain the conformance of a POSIX.1 implementation to an *assertion.*

C Standard
: FIPS 160. The reference text for FIPS 160 is ISO/IEC 9899, *Information technology—Programming languages—C.*

element
: A functional interface or a namespace allocation. Examples of elements are C functions or utility programs. Examples of namespace allocation include headers or error return value constants.

extended assertion
: An *assertion* that is not required to be tested.

GTI — general terminal interface

IUT — Implementation Under Test

PCD.1 — The POSIX conformance document required by the POSIX.1 standard.

POSIX Conformance Test Procedure (PCTP)
: The nonsoftware procedures used in conjunction with the PCTS to measure conformance.

POSIX Conformance Test Suite (PCTS)
: The collection of software possibly used in conjunction with other test methods to measure conformance.

ROFS — read-only file system

test method
: The software, procedures, or other means specified by a POSIX.1 standard to measure conformance.

: Test methods may include a PCTS, PCTP, or an audit of a PCD.

test result code

A value that describes the result of an *assertion test.*

2.3 Users

This PCTS can be used by a vendor or a user to test conformance of an operating system to the FIPS 151-2. This PCTS assumes that the environment under test is a POSIX.1 compatible environment.

2.3.1 Minimum Implementation Requirements

This PCTS requires the following:

- FIPS 151-2
  The implementation must support FIPS 160 (ISO/IEC 9899: Information technology—Programming languages—C). Implementations that do not support FIPS 160 may not attain successful compilation of this PCTS.
- {CHILD_MAX}
  This PCTS requires a value of {CHILD_MAX} of at least 10 to successfully run a number of the assertion tests. FIPS 151-2 requires a minimum value of {CHILD_MAX} of 25.
- headers
  Must be POSIX.1 compatible. The tester, for checking only, not for verification, may copy the system's include files to `NIST-PCTS/include` and then alter as desired to obtain POSIX.1 compatibility.
- functions/macros
  This PCTS will test that the required POSIX.1 functional interfaces and optionally-provided macros conform to FIPS 151-2 requirements.
- appropriate privileges
  This PCTS requires the implementation to provide the appropriate privileges synonymous with the super-user concept.

3. Contents

The documentation for this release is this Installation Guide.

This document specifies the procedures required to configure this PCTS and execute its tests. These procedures are numbered excerpts specified in italics.

This release contains no source code control identifiers.

3.1 POSIX.3.1 Assertion Changes

3.1.1 Assertion Updates

Assertions conditional on a conditional feature not being documented is not applicable to FIPS 151-2. FIPS 151-2 requires all supported conditional features to be documented (See §A item "p"). Therefore, a feature which is not documented is not supported by the implementation. Assertions of this nature are not tested and are assigned a test result code of *NOT_APPLICABLE*.

The following assertions are expected to be altered or should be altered in the ensuing IEEE 1003.3.1 standard (see comments preceding the instantiation of the ensuing test result code for the element stated).

- *sym_const*[06-08] — Replace with:
  06(C) If {_POSIX_CHOWN_RESTRICTED} is defined when <unistd.h> is included:
  The value of {_POSIX_CHOWN_RESTRICTED} is other than -1.

  07(C) If {_POSIX_NO_TRUNC} is defined when <unistd.h> is included:
  The value of {_POSIX_NO_TRUNC} is other than -1.

  08(C) If {_POSIX_VDISABLE} is defined when <unistd.h> is included:
  The value of {_POSIX_VDISABLE} is other than -1.
- *fork*[27] — Replace "27(PCTS_PROCESS_LIMIT?A:UNTESTED)" with "27(A)".
- *signal_con*[02] — Replace "void(*)()" with "void(*)(int)".
- *sigaction*[4] — Replace "void(*)()" with "void(*)(int)".
- *sigaction*[5-18] — Replace "Testing Requirement" with "Note".
- *link*[65] — Replace "(C)" with "(PCTS_FS?C:UNTESTED)".
- *rename*[68] — Replace "(C)" with "(PCTS_FS?C:UNTESTED)".
- *fpathconf*[28-29] — Replace "(C)" with "(PCTS_GTI_DEVICE?C:UNTESTED)".
- *settable*[53] — Replace "53(PCTS_GTI_DEVICE?C:UNTESTED)" with "R08".
  Append to assertion the text "See Assertion 69 in 7.1.1.11.)"
- *tcsetattr*[11] — Replace "A:UNTESTED)" with
  "C:UNTESTED) If the implementation supports input and output baud rates that differ:".
- *fgetc*[8] — Replace "08(C)" with "08(PCTS_GTI_DEVICE?C:UNTESTED)".
- *fgets*[8] — Replace "08(C)" with "08(PCTS_GTI_DEVICE?C:UNTESTED)".

- *fread*[8] — Replace “08(C)” with “08(PCTS_GTI_DEVICE?C:UNTESTED)”.
- *freopen*[34] — Replace “34(B) If {OPEN_MAX)≤{PCTS_OPEN_MAX}:” with “34({OPEN_MAX)≤{PCTS_OPEN_MAX}?B:A)”.
- *getc*[8] — Replace “08(C)” with “08(PCTS_GTI_DEVICE?C:UNTESTED)”.
- *getchar*[8] — Replace “08(C)” with “08(PCTS_GTI_DEVICE?C:UNTESTED)”.
- *gets*[8] — Replace “08(C)” with “08(PCTS_GTI_DEVICE?C:UNTESTED)”.
- *scanf*[8-9] — Replace “(C)” with “(PCTS_GTI_DEVICE?C:UNTESTED)”.
- *fscanf*[8-9] — Replace “(C)” with “(PCTS_GTI_DEVICE?C:UNTESTED)”.
- *sigsetjmp*[4] — Replace “04(A)” with “04(B)”.
- *tar*[20] — Replace “20(C)” with “20(D)”.
- *tar*[44] — Replace “44(C) If the ... ustar:” with “44(A)”.
- *cpio*[23] — Replace “23(A)” with “23(B)”.
- *cpio*[24] — Replace “24(C)” with “24(D)”.

3.1.2 Interpretations

No interpretations have been rendered at the time of the printing of this document.

3.2 Documents

3.2.1 NIST/CSL

This document and the “Certificate of Validation Requirements for FIPS 151-2” document are distributed as printed matter and are provided in *./DOCS*. These documents are formatted for use with `troff` and the memorandum macro package. To generate copies of these documents from the sources provided, the user is required to have the necessary expertise, software, and hardware to modify and execute the scripts provided in *./DOCS*.

3.2.2 Assertions

The assertions tested by the NIST-PCTS:151-2 are those in IEEE Std 2003.1-1992.

3.3 Source Controls

The complete set of files whose contents may be changed are:

- Header files in `./include` (See §6. and §B):
  <config>, <install.h>, <iso646_natv.h>, and <pcts_time.h>.
- Implementation dependent file for compiling and installing the `install` module. (See §6.1):
  `./src/install/s_install`.
- Compilation scripts:
  `./*/*/src/*/svcomp` —
  Changes restricted to library references only.
- Implementation dependent file for interfacing with *mtrwfs*(), *mtrofs*(), *unmtfs*(), *open_ctty*() (See §6.2):
  `./src/tools/tssf/tssf.c`

- Implementation dependent file for Data Interchange Format testing. (See §7.4):
  `./bin/dif_script`

A tester shall not change, replace, or alter any other files or permission modes of any file in this PCTS. The exception is those files in `./CERT_data` which are templates for the actual APTL generated files (see NIST POSIX Testing Policy— Certificate of Validation Requirements for FIPS 151-2).

4. Resources

You will need the following resources to successfully install this PCTS on a development or target system:

- **User Account** - A new user account should be made available for loading and testing this PCTS. See §5.1.
- **Time** - The time required to install and execute the NIST-PCTS:151-2 depends on the IUT's speed and peculiarities. On our in-house 25 SPECmark system, we installed the entire test suite in less than three hours and executed in less than two hours.
- **Disk Space** - The disk storage required when reading this distribution is approximately 8 MB. The DOCS and ass_NIST-PCTS directories, 1MB, may be deleted. Disk storage for execution modules will vary from one implementation to another. Implementations we have tested total storage requirements varied from 75MB to 200MB.

  The Certification Report can be generated on a POSIX.1 Section basis when the IUT has inadequate disk space to process all sections contiguously.
- **Utilities** - NIST-PCTS:151-2 requires the following utilities:
  - `ar` or a compatible library archiver;
  - `grep` or a compatible pattern retriever,
  - `cc -E` or a compatible macro preprocessor,
  - extended `cpio` creating and reading utility to read the NIST-PCTS:151-2 distribution and create an extended `cpio` archive;
  - extended `tar` format creating and reading utility to read and create an extended `tar` archive;
  - `mkdir` or a compatible directory maker;
  - `mknod` or a compatible special file builder;
  - `mv` or a compatible file mover;
  - `ranlib`, or the System V `ar/ld` combination;
  - `rm` or a compatible file remover;
  - `rm -r -f` or a compatible directory tree remover;
- **Printer** - for documents and possibly source listings;
- **Terminal Ports** - The NIST-PCTS:151-2 requires two terminal ports connected closed-loop (§6.2.1) for performing general terminal interface testing.
- **POSIX Headers** - The installation modules require those elements and structures defined in POSIX.1 headers to be present for compilation to be successful.
- **POSIX Functionality** - The library module `sv_sys/sv_setup.c` used in every test component module requires *sysconf*(). The modules in the `z_base` library require a functional interface to most of the POSIX.1 functions.

- **File Systems** - This PCTS requires an implementation to provide an additional file system to perform [EROFS] and [ENOSPC] testing (see §6.2.2).
- **Idle System** - This PCTS should be run on an idle system.

5. Getting Started

5.1 User Accounts

This PCTS should be installed into a newly created test user account. The sole purpose of this account should be to run the PCTS test suite.

The user-ID and group-ID for the test user account should contain different values for user-ID and group-ID. The test directory where the PCTS is to be loaded must be set to the test user's user-ID and group-ID. The user account login name must agree with the `config` (§B) required parameter **LOGNAME**.

5.2 Environment

This PCTS requires that the tester execute with no restrictions on created files.

<1>

• *Execute:* *umask 0*

5.3 Reading the Distribution

The NIST-PCTS:151-2 is distributed in extended `cpio` data interchange format.

Login as the tester and transfer the PCTS to the implementation to be tested. Modification time of files on the NIST-PCTS:151-2 distribution must be preserved or failures will be reported during verification.

<2>

• *Execute:* *cd tester_login_directory*
cpio -icBdmu[v] < distribution_file

Note1: The actual command to read the NIST-PCTS:151-2 *distribution may vary on your implementation.*

Note2: We read the NIST-PCTS:151-2 *distribution from a networked remote tape drive with the command:*
rsh remote_host dd if=/dev/rmt0 ibs=5120 | cpio -icBdumv

Note3: The NIST-PCTS:151-2 *distribution contains 7360 blocks.*

5.4 Apply Applicable Patch

Apply the latest FIXES Patch and BETA Patch if applicable. These patches are distrbuted by e-mail to those who have purchased the NIST-PCTS:151-2 and have registered with NIST/CSL. To register with NIST/CSL send an e-mail message to 151-2@sst.ncsl.nist.gov. (See Appendix G for procedures for installing patches).

<3>

• *Strip off e-mail header info from FIXES_PATCH_mm_dd_yy.*

• *Execute:* *sh FIXES_PATCH_mm_dd_yy 2>FP*

Ensure no problems were encountered.

<Φ> This step installs the active BETA Patch. This step should only be performed for BETA testing.

• *Execute:* *sh BETA_PATCH_mm_dd_yy 2>BP*

Ensure no problems were encountered.

5.5 Owner and Group Adjustment

The files read must be provided with the proper user ID and group ID of the tester. The procedure for accomplishing this can vary among implementations.

<4>

• *Ensure:* *The PCTS files are owned by the tester and have the tester's group ID.*

Note: On our implementation we perform the following commands.

chown -R tester NIST-PCTS
chgrp -R tester NIST-PCTS

5.6 Tape Contents

The NIST-PCTS:151-2 distribution contains the following directory trees:

CERT_data	- certification modules
DOCS	- documents (written for `troff` and mm macros)
STD	- POSIX.1 source test modules
STD/DEF	- POSIX.1 Chapter 2 (Definitions & General Requirements)
STD/PP	- POSIX.1 Chapter 3 (Process Primitives)
STD/PE	- POSIX.1 Chapter 4 (Process Environment)
STD/FD	- POSIX.1 Chapter 5 (Files & Directories)
STD/IO	- POSIX.1 Chapter 6 (Input & Output)
STD/DC	- POSIX.1 Chapter 7 (Device & Class-Specific Functions)
STD/LS	- POSIX.1 Chapter 8 (C Programming Language)
STD/PW	- POSIX.1 Chapter 9 (Passwords)
STD/DIF	- POSIX.1 Chapter 10 (Data Interchange Format)
bin	- executable file depository
data	- used by utility `verify`
include	- headers
lib	- archive library files depository
src	- commands, libraries, etc
src/Aids	- configuration files to test IUT for specific operating system
src/tools	- libraries
scripts	- used by utility `install`
tmp	- used by utility `verify`

5.7 NIST-PCTS:151-2 **Modules**

The NIST-PCTS:151-2 provides the following utilities:

- `install`
  The `install` command builds the executable modules of the NIST-PCTS:151-2.

- `verify`
  The `verify` command is the NIST-PCTS:151-2 driver. It can execute test components on an individual or on a SECTION-by-SECTION basis. The `verify` command has some options, when

executed on a SECTION-by-SECTION basis only, which control the amount of data to be retained from running a test component. The -d option retains the temporary files. The -k option retains the Raw Journal Report files for each test component in the associated `adm` directory under the name `.component_test_name`. The -K option retains the Raw Journal Report only if an error in testing was detected. The default is no options. On completion of each test component, `verify` will produce an output report.

When `verify` is executed on a component basis the Raw Journal Report is the only journal provided, and this is sent to the standard output device.

- `setaccess`
  The `setaccess` module traverses the entire NIST-PCTS:151-2 directory tree and assigns the proper ownership and protection modes to selected component test modules, data files and directories. Module `setaccess` touches each data file and will report all missing data files.
- `clraccess`
  The `clraccess` module traverses the entire NIST-PCTS:151-2 directory tree and resets the ownership and protection modes set by `setaccess` to their original modes.
- `validate`
  The `validate` module compares the tests result codes from the output journals with the expected test result codes and reports all discrepancies to `stdout`.

6. NIST-PCTS:151-2 Configuration Procedures

When the NIST-PCTS:151-2 is to be rerun for an IUT, then skip to §6.4 to reinstall the configuration files.

6.1 `install`

The `install` utility builds the executable modules of the NIST-PCTS:151-2.

The NIST-PCTS:151-2 support utilities are Strictly Conforming Applications. These utilities will compile and execute on all POSIX.1 Conforming Implementations. To allow an implementation's utilities to interface with the NIST-PCTS:151-2 support utilities, this PCTS allows the tester to define the command syntax for the implementations utilities. The `<install.h>` header file specifies the syntax for the implementation utilities.

The symbol `PATH_CMD` will be prefixed to commands which do not begin with a "/". This allows commands to be specified without a common absolute pathname. Commands must be regular, executable files, not *shell* scripts.

<5>

- *Edit* `<install.h>` *to define* `PATH_CMD` *and the command syntax for all the commands specified in* `struct cmd_imp`.

  NOTE: Some of the assertion tests rely on the `'C'` *compiler to provide the required testing. The APTL must exert proper judgement to ensure that if compilation flags are used for compilation commands they do not suppress warnings that are actually* FIPS 151-2 *errors.*

- *If your implementation supports the System V* `ar/ld` *combination, then the* `ranlib` *command is not required.*

The module `./src/install/s_install` is a script file that contains the commands for compiling and loading the `install` program.

<6>

- *Modify* `s_install` *as needed by the IUT to compile and install the* `install` *program..*

6.2 Target System Support Library

The NIST-PCTS:151-2 uses a library to interface required *target system support facilities.* File *tssf.c* in directory `./src/tools/tssf` is provided as an example. The target system support routine will be archived in library `./lib/tssf.a`.

6.2.1 Terminal Device Configuration

Many tests in this PCTS require two interconnected terminal device files. This section specifies the configuration and what is required from the installed configuration.

Two interconnected terminal device ports on the same machine, with each port simulating the presence of a terminal provides for error-free and repeatable testing. Currently, only those implementations which have a hardware configuration with at least two asynchronous communications ports available for use by this PCTS will be able to show compliance with those assertions that require this configuration.

The terminal special files corresponding to PORT_TTY and PORT_LOOP in "./include/config" should be configured on the system with modem control enabled. The special files' modes should be set to give read and write permission to the test user account.

Terminal devices are required to test those assertion numbers listed in the tables of Section 12 of this document, under the column marked GTI_dev.

When enabling asynchronous ports with a RS232 loop-back cable connection, GTI_dev marked assertion tests expect to open each side on the loop-back connector without blocking, such that a blocking open() would succeed. Data read/written on one port can be written/read on the other port respectively. Also a last close() on one side (with HUPCL set) would de-assert the line such that the other side detects it. This implies that the system asserts a RS232 line with RTS/CTS or DTR/DSR on open(), but also that any form of hardware handshaking will not de-assert the opened line.

<7>

- *Connect two terminal device ports back-to-back (*closed-loop*).*

6.2.2 Mounting and Unmounting File Systems

Ability to mount and unmount file systems is an implementation feature not provided by POSIX.1. POSIX.1 does require a conforming implementation to support read-only file systems and read-write file systems.

To test these file systems the NIST-PCTS:151-2 requires the tester to provide the interface routines *mtrwfs*(), *mtrofs*(), and *unmtfs*(). This interface must successfully perform the functionality as described below. The functionality can be performed programmatically or semi-manually.

Modules that use these interfaces are provided appropriate privileges.

6.2.2.1 Mount with Read/Write Capabilities

6.2.2.1.1 Synopsis

int mtrwfs(dirname, fsname)

*char *dirname;*

*char *fsname;*

6.2.2.1.2 Description

The *mtrwfs*() function will mount the file system specified by `fsname` on the mount point specified by `dirname`. The file system mounted should be of size 10 kilobytes or larger. A mountable filesystem much larger than 10 kilobytes will only result in a larger setup time when preparing to test assertions which depend on a full file system. The mount point `dirname` will be specified by NIST-PCTS:151-2. The file system must be mounted in a manner that allows both read and write access to the file system.

6.2.2.1.3 Returns

Upon successful completion, *mtrwfs*() shall return a value of zero. Otherwise, a value of -1 is returned.

6.2.2.2 Mount with Read Capabilities

6.2.2.2.1 Synopsis

int mtrofs(dirname, fsname)

*char *dirname;*

*char *fsname;*

6.2.2.2.2 Description

The *mtrofs*() function will mount the file system specified by `fsname` on the mount point specified by `dirname`. The mount point `dirname` will be specified by NIST-PCTS:151-2. The file system must be mounted in a manner that allows only read access to the file system.

6.2.2.2.3 Returns

Upon successful completion, *mtrofs*() shall return a value of zero. Otherwise, a value of -1 is returned.

6.2.2.3 Unmount a File System

6.2.2.3.1 Synopsis

int unmtfs(dirname, fsname)

*char *dirname;*

*char *fsname;*

6.2.2.3.2 Description

The unmtfs() function will unmount the file system specified by `fsname` which is currently mounted on the mount point specified by `dirname`. On completion the specified directory `dirname` may be used as the mount point for another file system.

Both the directory name and the file system are passed to this function as it is unknown which of these will be used to effect the unmount.

6.2.2.3.3 Returns

Upon successful completion, *unmtfs()* shall return a value of zero. Otherwise, a value of -1 is returned.

6.2.3 Establish a Controlling Terminal

Ability to establish a controlling terminal is an implementation feature not provided by POSIX.1. POSIX.1 does allow a conforming implementation to support a general terminal interface for controlling asynchronous communications ports.

A conforming implementation that supports asynchronous communications ports must interface to the routine *open_ctty*(). This interface must successfully perform the functionality as described below.

6.2.3.1 Establish a Controlling Terminal

6.2.3.1.1 Synopsis

int open_ctty(path, flags, mode)

*char *path;*

int flags;

mode_t mode;

6.2.3.1.2 Description

The *open_ctty*() function will establish the terminal device specified by `path` as the controlling terminal with permissions of `mode`.

6.2.3.1.3 Returns

Upon successful completion, *open_ctty*() shall open the controlling terminal device and shall return a non-negative integer representing the file descriptor for the controlling terminal device. Otherwise, a value of -1 is returned.

6.2.4 Configuring Target System Support Facilities

<8>

• *Edit file* `tssf.c` *in* `./src/tools/tssf` *as required for your implementation.*

6.3 config

The NIST-PCTS:151-2 uses the file `<config>` to specify all the installation parameters required for testing POSIX.1 conformance. Appendix B provides the explanation of the symbols used in `<config>` and the details on the specifications of what is required of the tester to define the IUT.

NOTE: APTL's should complete the template in Appendix A of *NIST POSIX Testing Policy— Certificate of Validation Requirements for FIPS 151-2* before tackling the configuration of `<config>`. By following this systematic approach, much of the input required by `<config>` will have been derived from the PCD and the data reported by the APTL, for this template, will agree with that used by `<config>`.

<9>

• *Edit* <config> *to define all required parameters.*

The `./scripts` directory contains script files which are used by `install` to compile and install the NIST-PCTS:151-2 modules. Scripts exist for compiling and installing this PCTS.

6.4 Reinstall of Configuration Files

Once the IUT has been configured, steps <5> thru <9> completed, all the configuration files required for the IUT should be placed into a non-hierarchial test directory.

When the configuration files to be modified for the IUT are available in a single directory, the script file, `./bin/config_iut`, is available for the tester to use. When this script is executed the NIST-PCTS:151-2 configuration files present in the directory specified will be installed. The NIST-PCTS:151-2 distribution configuration files will be retained with a trailor label name of ‘‘_DISTR’’.

<5-9>

• *Example:* *./bin/config_iut* *./src/Aids/oper_system*

This script file may be executed repeatedly during configuration procedures.

A tester's procedure should be to copy the changed configuration files to a test directory once the configuration files are initially established. If configuration are to be modified, then only perform the changes in the test directory and execute *./bin/config_iut* to reinstall all configuration files needed. This procedure simplifies the task of updating and maintenance of configuration files for an IUT.

The directory `./src/Aids` contains anonymous contributions of configuration files that have been used on specific implementations. No guarantees or claims are associated with these files that when used on the hardware and operating system stated that the IUT will sucessfully run the NIST-PCTS:151-2. Further more, no guarantees or claims are associated with these files that they were configured properly for the NIST-PCTS:151-2 nor the IUT tested. These configuration files are provided as examples of what a tester did use to run a version of the NIST-PCTS:151-2 during its development. A tester may be able to use one of the sets of configuration files provided to test an IUT.

7. **Installing** NIST-PCTS:151-2

7.1 Install Modules

Install the `install` program.

<10>

- *Execute:* *cd ./src/install*
- *Execute:* *./s_install*
- *Execute:* *cd ../..*

The NIST-PCTS:151-2 libraries, `verify`, `setaccess` and `clraccess` modules can now be installed. The time required on a 20 SPECmark system was approximately 20 minutes (120 SPECmark system, 8 minutes).

The output from `install`, Installation Report, is generated as the filename `journal`. To ensure a prior Installation Report is retained, rename the output filename. If a *Certificate of Validation* is to be obtained, the Installation Report obtained from the execution of `./scripts/utilities` must be renamed to *util_inst_rep*.

<11>

- *Execute:* *./bin/install ./scripts/utilities*
- *Correct any Installation Report errors and warnings that violate* FIPS 151-2 *conformance requirements before continuing.*
- *Execute:* *mv journal util_inst_rep*

7.2 Install Test Components

The test components can now be installed. The time required on a 20 SPECmark system was approximately 2.25 hours (120 SPECmark system, 35 minutes).

<12>

- *Execute:* *./bin/install ./scripts/NIST-PCTS*
- Correct any Installation Report errors and warnings that violate FIPS 151-2 conformance requirements before continuing.

NOTE: Test load module names have a length less than {POSIX_NAME_MAX}. Each test module's name is derived from the POSIX.1 subclause designation and the corresponding POSIX.3.1 starting assertion number being tested. (Ex. The name for the module that tests assertion #4 of *fork*() would be 3.1.1_04.) When multiple functions have the same POSIX.1 subclause designation, then an additional alphabetic letter follows the POSIX.1 subclause designation. (Ex. The name for the module that starts with code to test assertion #4 of *wait*() and *waitpid*() would be 3.2.1a_04 and 3.2.1b_04 respectively.) When a test requires multiple load modules then succeeding modules are designated with a trailing alphabetic letter starting with "x".

NOTE: When multiple POSIX.1 functions have similar requirements for testing. Instead of the assertion number, a name designation is provided. The complete set of tests that have this designation are:

syn	synopsis section testing
pth	pathname testing
acc	access testing

7.3 Special Files

<13>

- *If IUT supports character special devices, then create a character special file with pathname of*
  `./STD/DIF/data/dif.d/type.d/mode_cks.d/S_ISCHR`
  with the user ID and group ID of the tester.
  Example:
  mknod ./STD/DIF/data/dif.d/type.d/mode_cks.d/S_ISCHR c 77 77
  chown tester ./STD/DIF/data/dif.d/type.d/mode_cks.d/S_ISCHR
  chgrp tester ./STD/DIF/data/dif.d/type.d/mode_cks.d/S_ISCHR

- *If IUT supports block special devices, then create a block special file with pathname of*
  `./STD/DIF/data/dif.d/type.d/mode_cks.d/S_ISBLK`
  with the user ID and group ID of the tester.
  Example:
  mknod ./STD/DIF/data/dif.d/type.d/mode_cks.d/S_ISBLK b 77 77
  chown tester ./STD/DIF/data/dif.d/type.d/mode_cks.d/S_ISBLK
  chgrp tester ./STD/DIF/data/dif.d/type.d/mode_cks.d/S_ISBLK

7.4 Special Permissions

The NIST-PCTS:151-2 requires many of the modules in the test suite to execute as a specific user and many of the data files and directories to possess special ownership and/or permissions. The `setaccess` module provides these attributes. This module must be setup to execute as the super-user.

<14>

- *As the super-user, set the owner of* `./bin/setaccess` *to be super-user and set the mode bits S_ISUID, S_IXUSR and S_IXGRP.*

<15>

- *Execute: ./bin/setaccess*

7.5 Data Interchange Format

A conforming implementation must be able to read and write both `cpio` and extended `tar` data interchange formats. The NIST-PCTS:151-2 uses a set of data files to create `cpio` and extended `tar` archives. A Bourne shell script is used to establish test files. This script creates the `cpio` and extended `tar` archives with its header and source data using characters as represented in ISO/IEC 646 IRV, and extracts data files from the PCTS distributed `cpio` and extended `tar` archives.

<16>

- *Header file <iso646_natv.h> specifies a mapping between ISO/IEC 646 IRV and the IUT's local codeset.*
  Edit <iso646_natv.h> if needed.

<17>

- *Edit the script* `./bin/dif_script` *for proper functioning on the implementation to be tested.*
- *Execute script file: ./bin/dif_script*

8. **Testing** NIST-PCTS:151-2

8.1 Functional Interfaces

The NIST-PCTS:151-2 provides for the *tester* to sequentially run all the test components by executing a single command (`verify`). The Raw Journal Report for each component tested is placed in the file `./STD/POSIX_CHAPTER/adm/.test_component` and the Output Report is placed in the file `./STD/POSIX_CHAPTER/adm/journal`. The POSIX_CHAPTER designation is the appropriate directory DEF, PP, PE, FD, IO, DC, LS, PW, DIF corresponding to the POSIX chapter where the test component is specified. The time required to execute Step <18> on a 20 SPECmark system was approximately 90 minutes.

<18>

- *Execute: ./bin/setaccess (Additional insurance)*
- *Execute: ./bin/verify IUT*

NOTE: A report submitted to NIST/CSL for FIPS 151-2 Validation, when adequate storage exists, must have been generated by the preceding command. When the implementation has limited storage and can only compile and execute the PCTS in sections, then the sections of the report must be installed and verified in the following order: STD/DEF, STD/PP, STD/PE, STD/FD, STD/IO, STD/DC, STD/LS, STD/PW, STD/DIF (See *NIST POSIX Testing Policy— Certificate of Validation Requirements for FIPS 151-2*).

Test components may be run a setion or multiple section at a time by specifying the appropriate test directory to `verify`.

<Ex.>

- *Execute section "Files and Directories" tests:*
  ./bin/verify -k STD/FD
- *Execute multiple section tests:*
  ./bin/verify -k STD/DEF STD/PP STD/PE STD/FD

Test components may be run a component at a time by specifying the test component to `verify`. When `verify` is executed by test component the Raw Journal Report is only outputted to the standard output device.

<Ex.>

- *Execute component: ./bin/verify STD/FD/link*

9. Reports

The NIST-PCTS:151-2 produces three reports.

9.1 Installation Report

The Installation Report captures all the commands that are used to generate the NIST-PCTS:151-2 utilities and test modules and all output results reported by the implementations utilites used. Installation reports are placed in the file `journal`. Move any journals you want to save to another file before running the `install` command again.

An example of the Installation Report is provided in Appendix C.

9.2 Raw Journal Report

The Raw Journal Report provides a step-by-step log of what the NIST-PCTS:151-2 requires for each test, the preparations taken for each test, and the results of each test. This log is in sufficient detail such that any failure can be quickly pointed to a specific line of the actual source code where the test failed.

Raw journal reports, if the -k or -K option (See §5.7, `verify`) is used, are placed in the `./STD/Section/adm` directories. Where `Section` is the name of a particular section, e.g., `FD`. A separate Raw Journal Report is generated for each component tested. The name of each Raw Journal Report is the name of the component preceded by a '.'. (e.g., The Raw Journal Report for *link*() is *.link*.

A Raw Journal Report for *cfgetispeed*() is provided in Appendix E.

9.3 Output Report

The NIST-PCTS:151-2 output report is based on the IEEE Std 1003.3-1991. The report consists of a test result code for every assertion of each element. The test result code is one of the following:

- *PASS* - successful test of a base assertion.
- *FAIL* - unsuccessful test of a base assertion.
- *PASS_EXTENDED* - successful test of an extended assertion.
- *FAIL_EXTENDED* - unsuccessful test of an extended assertion.
- *UNTESTED* - the test for this extended assertion does not exist for the target system.
- *UNSUPPORTED* - the conditional feature was not tested because it was not implemented.
- *NOT_APPLICABLE* - The assertion is not applicable to the profile as required by FIPS 151-2.
- *NOT_TESTABLE* - The implementation lacks a PCTS required testing constraint (GTI devices are not provided) or a PCTS testing parameter's value ({PCTS_NAME_MAX}) is not applicable to the value required by the assertion's testing constraint.

  NOTE: When the PCTS determines that the testing constraint has not been satisfied, the PCTS uses *NOT_TESTABLE* instead of the POSIX.3.1 specified test result code of *UNTESTED*. When a POSIX.3.1 testing constraint assertion is extended and the code to test the assertion is provided, the test result code is *UNTESTED*.

- *UNRESOLVED* - the test for the assertion could not be completed. Two examples are: 1) a directory was required before the assertion could be tested, and mkdir() failed; 2)An unexpected signal was sent to the test driver, causing the test to abort. An UNRESOLVED test result code is an intermediate result that is required to be resolved to one of the other test result codes.
- *NOT_INITIATED* - No test was run for this assertion number. This test result code is normally the result of an earlier failure in a test module. See the Raw Journal Report to determine the reason for failure.

The NIST-PCTS:151-2 shall output *UNUSED* instead of a test result code for assertion numbers in the *UNUSED* list of each element. *UNUSED* means there is no assertion associated with the number.

Output reports are placed in the `./STD/Section/adm` directories. Where `Section` is the name of a particular section, e.g., `FD`. The name of the output reports are `journal`.

An example of the Output Report is provided in Appendix F.

A list of potential problems and possible solutions is provided in Appendix H.

10. FIPS 151-2 **PCTS Conformance Testing**

10.1 Type Checking

The detection of type checking non-conformity relies upon the ANSI C compiler to generate an error or a warning message.

<1>

- *Ensure no errors or warnings , which imply errors, are reported in the compilation phase of the PCTS.*
  When such warning or errors occur, the corresponding final test result codes are FAIL *even if the* NIST-PCTS:151-2 *reports them as* PASS.

10.2 Required Test Result Codes

The following tables provide the acceptable test result codes other than PASS for each assertion in IEEE Std 2003.1-1992.

Terminology and General Requirements

Test Code Class Element	#	UNSUPPORTED Conditional other	UNTESTED Extended
gen_terms	7		1-2,4-7
gen_concept	1		1
err_numbs	3		3
psd_types	5		
env_descr	2		
c_lang	5		5
num_lmts	17	4-15	16
sym_const	8	6-8	

Additional test result codes:
PASS_EXTENDED — *psd_types*[5]

Process Primitives

Test Code Class Element	#	UNSUPPORTED Conditional macro	other	NOT_TESTABLE Testing Constraint GTI_dev	other	UNTESTED Extended
fork	28	2-3		20		5-6,26,28
execl	60	2-3	38		38,44-46,49,54-55	22-23,26-27,52,60
execv	60	2-3	38		38,44-46,49,54-55	22-23,26-27,52,60
execle	60	2-3	38		38,44-46,49,54-55	22-23,26-27,52,60
execve	60	2-3	38		38,44-46,49,54-55	22-23,26-27,52,60
execlp	60	2-3	38		38,44-46,49,54-55	22-23,26-27,52,60
execvp	60	2-3	38		38,44-46,49,54-55	22-23,26-27,52,60
wait	15	2-3				
waitpid	27	2-3				
_exit	13	2-3		11-12		6
signal_con	61			14-15,46 53,56,59,61	46,53,56	3,9,39,47-48,54,57
kill	20	2-3			16	
sigemptyset	4	2-3				
sigfillset	4	2-3				
sigaddset	6	2-3			5	6
sigdelset	6	2-3			5	6
sigismember	6	2-3			5	6
sigaction	21	2-3				
sigprocmask	14	2-3				
sigpending	5	2-3				5
sigsuspend	9	2-3				
alarm	9	2-3				
pause	7	2-3				
sleep	6	2-3				

Additional test result codes:

NOT_APPLICABLE — *execl*[32], *execv*[32], *exele*[32], *execve*[32], *execlp*[32], *execvp*[32], *kill*[19]

UNUSED — *execl*[9-15], *execv*[9-15], *exele*[9-15], *execve*[9-15], *execlp*[59], *execvp*[59]

Process Environments

Test Code Class Element	#	UNSUPPORTED Conditional macro	other	NOT_TESTABLE Constraint GTI_dev	other	UNTESTED Extended
getpid	4	2-3				
getppid	4	2-3				
getuid	4	2-3				
geteuid	4	2-3				
getgid	4	2-3				
getegid	4	2-3				
setuid	9	2-3		6-7	8	
setgid	9	2-3		6-7	8	
getgroups	7	2-3				
getlogin	5	2-3				5
getpgrp	4	2-3				
setsid	9	2-3		6,8-9		9
setpgid	16	2-3			11	
uname	7	2-3				7
time	6	2-3				6
times	11	2-3				11
getenv	8	2-3				8
ctermid	8	2-3		4-5,7		8
ttyname	6	2-3		4		6
isatty	5	2-3		4		
sysconf	19	2-3		15-16		17

Additional test result codes:

NOT_APPLICABLE — *setuid*[6-7], *setgid*[6-7], *setpgid*[12], *sysconf*[15-16]

Files and Directories

Test Code Class Element	#	UNSUPPORTED Conditional macro	ROFS	modem	other	NOT_TESTABLE Testing Constraint GTI_dev	other	UNTESTED Extended
dirent.h	3							3
opendir	35	2-3			31		24-25,30-31	33-35
readdir	13	2-3					12	7,13
rewinddir	5	2-3						2
closedir	7	2-3			5-6			7
chdir	27	2-3					21-22	
getcwd	8	2-3						
open	67	2-3	65-67			44	56-57	23,34,38-41, 51,59,62
creat	53	2-3	52-53			36	45-46	5-6,22, 32-33,48,50
umask	6	2-3						
link	66	2-3	64		9-10,23-24, 37,40, 61-62,65		5-6,46-52,65	5-6,19-20,35, 44,58
mkdir	36	2-3	36				27-30	5-6,9-10,34
mkfifo	33	2-3	33				26-27	5-6,9-10,31
unlink	39	2-3	39		9-10,23,37		30-31	5-6,20-21, 26,29,38
rmdir	36	2-3	36				29-30	5-6,20,22, 24,27
rename	69	2-3	67		40,42,46, 61-62,68		52-55,61-62,68	5-6,19-20,35, 38,41,48,64
stat.h	19				4-5			19
stat	37	2-3					31-32	
fstat	14	2-3						
access	37	2-3	35		23,24,36,37		28-29	37
chmod	34	2-3	34				5-6,21-23,26-27	
chown	37	2-3	35		19		5-6,22,25-26,36	37
utime	36	2-3	36		22-23		5-6,28-29	
pathconf	57	2-3			21,23,25, 27,30,33 36,38,40	40-41	22-25,51-52	4-14
fpathconf	25	2-3			5,7,9,11,14 17,20, 22,24	6-9, 24-25, 28-29		

Additional test result codes:

NOT_APPLICABLE — *opendir*[16,32], *chdir*[15], *open*[15], *creat*[15], *link*[15,29,38,63,66], *mkdir*[15], *mkfifo*[15], *unlink*[15], *rmdir*[15], *rename*[15,29,47,63,69], *stat*[15], *access*[15], *chmod*[15], *chown*[15,21], *utime*[15], *pathconf*[15]

PASS_EXTENDED — *creat*[9-10], *pathconf*[17]

Input and Output Primitives

Test Code Class Element	#	UNSUPPORTED Conditional macro	UNSUPPORTED Conditional other	NOT_TESTABLE Testing Constraint GTI_dev	NOT_TESTABLE Testing Constraint other	UNTESTED Extended
pipe	13	2-3				13
dup	6	2-3				
dup2	9	2-3				9
close	13	2-3				6,8-10,13
read	33	2-3		11		14-17,21-22,28
write	42	2-3		12,40	27-29 41	11,15-17,20,22-24, 30-33,38,41
fcntl.h	7					
fcntl	52	2-3	46,51			50,52
lseek	13	2-3				

Additional test result codes:

NOT_APPLICABLE — *read*[12-13,16], *write*[13-14], *fcntl*[47]

PASS_EXTENDED — *write*[11,22,41]

Device- and Class- Specific Functions

Test Code Class Element	#	UNSUPPORTED Conditional macro	UNSUPPORTED Conditional modem	UNSUPPORTED Conditional other	NOT_TESTABLE Testing Constraint GTI_dev	NOT_TESTABLE Testing Constraint other	UNTESTED Extended
interface	69		64-66,69	9,59,63	1-69		2,68
settable	71		54, 56-58	22-23,39-40, 47,48	8-34,36,39-58, 60-68,71	33-34	5,13,15,18 43-44,49,51
cfgetospeed	6	2-3					
cfsetospeed	5	2-3			5		
cfgetispeed	4	2-3					
cfsetispeed	5	2-3			5		
tcgetattr	8	2-3		5,6	4-6		
tcsetattr	21	2-3	10	11-12,20	5-21		6,9
tcsendbreak	11	2-3			4-9		
tcdrain	11	2-3			4-8,11	11	
tcflow	16	2-3			5-13,15		
tcflush	14	2-3			5-11,13		
tcgetpgrp	9	2-3			4-5,8		
tcsetpgrp	11	2-3		6	4,6,8,10-11		

Additional test result codes:

NOT_APPLICABLE — *tcgetpgrp*[7], *tcsetpgrp*[7]

PASS_EXTENDED — *interface*[2]

UNUSED — *settable*[53]

Language-Specific Services for C

Test Code Class		UNSUPPORTED Conditional			NOT_TESTABLE Testing Constraint		UNTESTED Extended
Element	#	macro	ROFS	other	GTI_dev	other	
assert	2						2
isalnum	4	2-3					4
isalpha	4	2-3					4
iscntrl	4	2-3					4
isdigit	4	2-3					4
isgraph	4	2-3					4
islower	4	2-3					4
isprint	4	2-3					4
ispunct	4	2-3					4
isspace	4	2-3					4
isupper	4	2-3					4
isxdigit	4	2-3					4
tolower	4	2-3					4
toupper	4	2-3					4
acos	4	2-3					4
asin	4	2-3					4
atan	4	2-3					4
cos	4	2-3					4
sin	4	2-3					4
tan	4	2-3					4
cosh	4	2-3					4
sinh	4	2-3					4
tanh	4	2-3					4
exp	4	2-3					4
log	4	2-3					4
log10	4	2-3					4
sqrt	4	2-3					4
ceil	4	2-3					4
fabs	4	2-3					4
floor	4	2-3					4
atan2	4	2-3					4
pow	4	2-3					4
fmod	4	2-3					4
frexp	4	2-3					4
ldexp	4	2-3					4
modf	4	2-3					4
setjmp	4	1-3					4
longjmp	4	2-3					2,4
clearerr	4	2-3					4
fclose	19	2-3			19		6,8-10,16-17
feof	5	2-3					
ferror	5	2-3					

Language-Specific Services for C (Cont'd)

Test Code Class Element	#	UNSUPPORTED Conditional macro	ROFS	other	NOT_TESTABLE Testing Constraint GTI_dev	other	UNTESTED Extended
fflush	11	2-3			11		8-9
fgetc	9	2-3			8-9		
fgets	9	2-3			8-9		
fopen	34	2-3	34			25-28	29,32
fputc	11	2-3			11		8-9
fputs	11	2-3			11		8-9
fread	9	2-3			8-9		
freopen	35	2-3	43			34-37	21,23-25,38,41
fseek	11	2-3					8-10
ftell	6	2-3					
fwrite	12	2-3			12	5	9-10
getc	9	2-3			8-9		
getchar	9	2-3			8-9		
gets	9	2-3			8-9		
perror	4	2-3					
printf	11	2-3			11		8-9
fprintf	11	2-3			11		8-9
sprintf	4	2-3					4
vprintf	12	3-4			12		9-10
vfprintf	12	3-4			12		9-10
putc	11	2-3			11		8-9
putchar	11	2-3			11		8-9
puts	11	2-3			11		8-9
remove	19	2-3					5-6
rewind	4	2-3					
scanf	9	2-3			8-9		
fscanf	9	2-3			8-9		
sscanf	4	2-3					4
setbuf	4	2-3					4
tmpfile	7	2-3					5,7
tmpnam	4	2-3					4
ungetc	4	2-3					4
abs	4	2-3					4
atof	4	2-3					4
atoi	4	2-3					4
atol	4	2-3					4

Additional test result codes:

NOT_APPLICABLE — *fopen*[15], *remove*[15], *freopen*[15], *vprintf*[2], *vfprintf*[2]

PASS_EXTENDED — *freopen*[34]

Language-Specific Services for C (Cont'd)

Test Code Class Element	#	UNSUPPORTED Conditional macro	ROFS	other	NOT_TESTABLE Testing Constraint GTI_dev	other	UNTESTED Extended
rand	4	2-3					4
srand	4	2-3					4
calloc	4	2-3					4
free	4	2-3					4
malloc	4	2-3					4
realloc	4	2-3					4
abort	15	2			12-13		7
exit	17	2-3			15-16		2,9
bsearch	4	2-3					4
qsort	4	2-3					4
strcpy	4	2-3					4
strcat	4	2-3					4
strtok	4	2-3					4
strncpy	4	2-3					4
strncat	4	2-3					4
strcmp	4	2-3					4
strncmp	4	2-3					4
strchr	4	2-3					4
strrchr	4	2-3					4
strcspn	4	2-3					4
strspn	4	2-3					4
strpbrk	4	2-3					4
strstr	4	2-3					4
strlen	4	2-3					4
ctime	15	2-3					
asctime	4	2-3					
gmtime	4	2-3					
localtime	16	2-3					
mktime	4	2-3					16
strftime	8	2-3					
setlocale	16	2-3					
fileno	10	2-3					10
fdopen	13	2-3					13
sigsetjmp	6	1-3					4
siglongjmp	6	2-3					
tzset	5	2					2

Additional test result codes:

NOT_APPLICABLE — *exit*[14,18]

System Database

Test Code Class Element	#	UNSUPPORTED Conditional macro	ROFS	NOT_TESTABLE Testing Constraint GTI_dev	other	UNTESTED Extended
getgrgid	6	2-3				7
getgrnam	1	2-3				6
getpwuid	3	2-3				7
getpwnam	5	2-3				6

Data Interchange Format

Test Code Class Element	#	UNSUPPORTED Conditional Char Sp	Block Sp	UNTESTED Extended
dif	11			4-11
tar	45	25,33	26,34	4,8,11,20,30,37,42-43,45
cpio	25			9,16,22-25

10.3 Validation Utility

The test result codes from the output journal can be compared with the expected result codes from the tables provided above. The utility `validate` will report all discrepancies to `stdout`.

<1>

- *Execute:* *./bin/validate*

11. FIPS 151-2 PCD Conformance Testing

11.1 PCD.1 Audit

POSIX.1 specifies requirements for the PCD.1 in POSIX.1, 1.3.1.2. These requirements are detailed by the specifications of the documentation assertions in POSIX.3.1. A PCD.1 audit had been required for FIPS 151-2 conformance. IEEE Interpretations stated in August 94, that a conflict exists between the requirements of POSIX.1 and the documentation assertions of POSIX.3.1. Since NIST/CSL no longer has a document to cite for PCD.1 requirements the FIPS 151-2 PCD.1 audit requiremnt was dropped on 8/29/94 (see *Certificate of Validation Requirements for FIPS 151-2*, 7.6 PCD.1 Requirement Dropped).

11.1.1 PCD.1 versus <config>

The tester must ensure that the parameters specified for `<config>` are consistent with the specifications of the IUT's PCD.1.

11.1.2 POSIX.1 Documentation Requirements

The tester should ensure that the implementation's PCD.1 complies with the "current understanding" of the documentation requirements of POSIX.1.

11.1.3 Additional FIPS 151-2 Documentation Requirements

The tester must ensure that the additional documentation requirement as specified in FIPS 151-2 (See §A.p) is met. See *NIST POSIX Testing Policy— Certificate of Validation Requirements for FIPS 151-2* for the specific set of requirements.

12. Collecting Data for an NIST/CSL Certification Report

12.1 Generating `CV_report`

When the APTL desires a Certification Report to request a Validation Certificate for FIPS 151-2, the script `./bin/gen_cert_rept` will generate the required report. The *NIST POSIX Testing Policy—Certificate of Validation Requirements for FIPS 151-2*, §6.*.* provides specific information on the contents of the required files that constitute the Certification Report.

The file `./NIST-PCTS/CERT_data/LIST_FILES` lists all the files to be included in the Certification Report. The APTL must as a minimum collect, from the IUT, all the required files via the extended `cpio` format creating utility once testing is completed. Once this data is collected, only the files:

```
./CERT_data/NPTP_appendixA
./CERT_data/NPTP_appendixB
./CERT_data/alt_svcomp
./CERT_data/resolved_trc
./CERT_data/imp_features
./CERT_data/config_IUT
```

may be altered by the APTL. All files configured for the execution of the NIST-PCTS:151-2, and those created on execution and then archived by the APTL for submission to NIST/CSL as part of the Certification Report, must retain the times at which they were last modified by the PCTS.

APPENDIX A

FIPS PUB 151-2 Specifications

The FIPS 151-2 specifications are the specifications contained in the International Standard ISO/IEC 9945-1, "Information Technology — Portable Operating System Interface (POSIX) — Part 1: System Application Program Interface (API) [C Language]" with the modifications specified below. These modifications are required for implementations of POSIX.1 that are acquired by Federal agencies.

These modifications ensure that applications, which choose to use those optional features specified in POSIX.1 and mandated below, are strictly conforming FIPS 151-2 applications (portable to all conforming FIPS 151-2 implementations). For each modification a reference to the associated POSIX.1 text is provided.

a. Implementations claiming conformance to FIPS 151-2 shall provide the functionality specified in FIPS 160 and provide C Standard Language-Dependent System Support. (The reference text for FIPS 160 is ISO/IEC 9899: Information technology—Programming languages—C) [See POSIX.1 Subclause 1.3.3-1.3.3.3 lines 143-188].

b. Implementations claiming conformance to FIPS 151-2 shall define the POSIX.1 environment variable, **HOME**, in the environment for the login shell. [See POSIX.1 Subclause 2.6 lines 698-699].

c. Implementations claiming conformance to FIPS 151-2 shall define the POSIX.1 environment variable, **LOGNAME**, in the environment for the login shell. [See POSIX.1 Subclause 2.6 lines 698-699].

d. Implementations claiming conformance to FIPS 151-2 shall support the POSIX.1 option {NGROUPS_MAX} such that the value of {NGROUPS_MAX} is greater than or equal to eight (8). [See POSIX.1 Subclause 2.8.3 lines 1013-1015].

e. Implementations claiming conformance to FIPS 151-2 shall support a minimum value of 25 for the POSIX.1 variable {CHILD_MAX}. [See POSIX.1 Subclause 2.8.4 lines 1029-1030].

f. Implementations claiming conformance to FIPS 151-2 shall support a minimum value of 20 for the POSIX.1 variable {OPEN_MAX}. [See POSIX.1 Subclause 2.8.4 lines 1031-1032].

g. Implementations claiming conformance to FIPS 151-2 shall support the functionality associated with {_POSIX_JOB_CONTROL} being defined in <*unistd.h*>. [See POSIX.1 Subclause 2.9.3 lines 1117-1118].

h. Implementations claiming conformance to FIPS 151-2 shall support the functionality associated with {_POSIX_SAVED_IDS} being defined in <*unistd.h*>. [See POSIX.1 Subclause 2.9.3 lines 1119-1120].

i. Implementations claiming conformance to FIPS 151-2 shall support the functionality associated with {_POSIX_CHOWN_RESTRICTED} being defined in <*unistd.h*> with value other than -1. [See POSIX.1 Subclause 2.9.4 lines 1136-1139].

j. Implementations claiming conformance to FIPS 151-2 shall support the functionality associated with {_POSIX_NO_TRUNC} being defined in <*unistd.h*> with value other than -1. [See POSIX.1 Subclause 2.9.4 lines 1140-1141].

k. Implementations claiming conformance to FIPS 151-2 shall support the functionality associated with {_POSIX_VDISABLE} being defined in <*unistd.h*> with value other than -1. [See POSIX.1 Subclause 2.9.4 lines 1142-1144].

l. Implementations claiming conformance to FIPS 151-2 shall support the functionality associated with the setting of the group ID of a file (when it is created) to that of its parent directory. [See POSIX.1 Subclause 5.3.1.2, 5.4.1.2, and 5.4.2.2 lines 188-192, 384-385, and 431-432].

m. Implementations claiming conformance to FIPS 151-2 shall support, for terminal devices, the functionality associated with an interrupted *read*() such that the return from *read*() when interrupted by a signal after successfully reading some data returns the number of bytes the system has read. [See POSIX.1 Subclause 6.4.1.2 lines 132-134].

n. Implementations claiming conformance to FIPS 151-2 shall support, for terminal devices, the functionality associated with an interrupted *write*() such that the return from *write*() when interrupted by a signal after successfully writing some data returns the number of bytes the system has written. [See POSIX.1 Subclause 6.4.2.2 lines 214-216].

o. Implementations claiming conformance to FIPS 151-2 shall support the functionality associated with the symbols CS7, CS8, CSTOPB, PARODD, and PARENB defined in <termios.h> for asynchronous general terminal interface devices. [See POSIX.1 Subclause 7.1.2.4 lines 383-387].

p. Implementations claiming conformance to FIPS 151-2 shall document in the POSIX Conformance Document the FIPS 151-2 conditional features implemented (see ‘‘NIST POSIX Testing Policy—Certificate of Validation Requirements for FIPS 151-2’’ for documentation details). (The term conditional features are the features or behaviors referred to in FIPS 151-2 that need not be present on all conforming implementations. IEEE Std 2003.1-1992 lists the documentation assertions for POSIX.1)

APPENDIX B

NIST-PCTS:151-2 **Configuration Parameters**

The following parameters (listed in the order as specified in `./include/config`) are used by the NIST-PCTS:151-2 to define the environment of the implementation to be tested. Each parameter, when appropriate, is provided a reference to the applicable documentation assertion in IEEE Std 2003.1-1992. This reference, when used with the PCD for the implementation, should provide the needed information to properly configure these parameters for the NIST-PCTS:151-2.

The NIST-PCTS:151-2 and its documentation audit will verify, whenever possible, the installation parameters specified for the IUT.

PCTS_PATH
: Absolute pathname of the target directory where the PCTS is to reside.

UID, GID & LOGNAME
: User and group identification numbers and login name associated with the tester on the target system. This tester will be the only user who may run the tests reliably, since certain tests test results against this value. Assign UID and GID with different values.

ROOT_UID & ROOT_GID
: The user and group identification numbers associated with the super-user on the target system.

BIN_UID & BIN_GID
: User and group identification numbers for another user. This PCTS requires that no other process is running as this user.

ANOTHER_UID
: One more user identification number. This parameter is used exclusively for *kill*() testing.

SUPPLEMENTARY_GROUPS
: To validate values of the supplementary groups returned by *getgroups*(), the PCTS installer is required to provide the values of those group ID numbers that have been assigned as supplementary group IDs to the test user. The number of supplementary groups required to be assigned to the test user (UID) is NGROUPS_MAX. The value BIN_GID must not be a parameter of SUPPLEMENTARY_GROUPS.

The tester must ensure that the groups specified are defined for the system being tested. This groups definition mechanism is implementation specific. The values of the supplementary groups are specified as a string separated by a comma.

UNUSED_UID, UNUSED_GID, UNUSED_LOGNAME, & UNUSED_GROUP_NAME
: User and group identification numbers and login and group names not associated with any user in the password and group databases.

GDB_ENTRY
: An actual entry in the group database. The format is: “group_name,group_ID,group_member,group_member,...”.

PDB_ENTRY
: An actual entry in the password database. The format is: “login_name,user-

ID,group_ID,home_directory,default_shell".

CHAR_SPECIAL (2.2.2.9[02])
If the IUT provides support for character special devices, then specify `'1'` (see §8.1, step <16> for character special file creation requirement) otherwise specify `'0'`.

BLOCK_SPECIAL
If the IUT provides support for block special devices, then specify `'1'` (see §8.1, step <16> for block special file creation requirement) otherwise specify `'0'`.

PCD_READ_ONLY_FILE_SYSTEM
If the IUT supports read-only file systems, then specify `'1'` (and specify a file system for PCTS_FS) otherwise specify `'0'`.

PCTS_FS
This parameter specifies the device on which the test suite mounts and unmounts a file system. This file system is used to test assertions conditional on read-only file systems being supported and to test assertions which attempt to write more data than the file system can accomodate.

When the implementation under test does not provide another file system to test with, then specify the parameter for PCTS_FS as

PCTS_FS ""

and the file system used to test the required assertions will be the one available to the PCTS.

The mounting and unmounting is performed by the implementation supplied interface functions *mtrwfs*(), *mtrofs*(), and *unmtfs*(). The implementation provides these function interfaces. How it is performed is determined by the implementation. See the installation manual for documentation on these required interface functions. The file system mounted should be configured with 20 - 100 free blocks. The implementation must support the generation of a signal file with length corresponding to the number of free blocks on this mounted file system. This should be a *clean* spare file system.

The time required to perform the required testing on PCTS_FS may vary significantly from one implementation to another. Depending on the procedures required for mounting and unmounting file systems and the amount of free space provided on PCTS_FS.

TIME_PCTS_FS_TESTING
This parameter specifies the minimum time the PCTS waits for the tests associated with [EROFS] and [ENOSPC] to complete. This time is specified in seconds.

UNAME_INFO
The PCTS will test the strings returned by *uname*() against the parameters provided for this item. Each parameter is specified as one or more characters separated by a comma. The ordered sequence of the paramters are:

- SYSNAME — name of the implementation being tested
- NODENAME — target system node name as returned by *uname*()
- RELEASE — target operating system release number as returned by *uname*()
- VERSION — target operating system version as returned by *uname*()
- MACHINE — target hardware name as returned by *uname*()

A `','` can be included in any parameter by deliminating the `','` with two `'\'` characters. e.g., `UNAME_INFO NIST-SYS,nemo.ncsl.nist.gov,3\\,1,0,ABCD` specifies the expected RELEASE string as `3,1`.

FILENAME_OF_TERMINAL_FILE
The pathname of a terminal file.

PORT_TTY and PORT_LOOP
Two general terminal interface asynchronous devices are required to test most of the assertions associated with the general terminal interface requirements. If these devices are not available, then these parameters must be specified as null strings ("") and asociated assertions will be resolved to UNSUPPORTED (See Appendix XXX). Implementations that do not support these devices can receive an NIST Certificate which will specify that the POSIX General Terminal Interface was not tested.

PORT_TTY and PORT_LOOP must be connected as specified in this manual. This connection allows this PCTS to direct output to one of these ports and read the data generated by reading the the other port. Conversely, input on one ports is output on the other port. This closed-loop connection allows the PCTS to simulate a keyboard, controlling terminal, and terminal display to test the general terminal interface assertions.

GTI_MODEM_CONTROL
If the implementations supports modem control on the asynchronous serial terminal ports PORT_TTY and PORT_LOOP, then specify `'1'` otherwise specify `'0'`. This PCTS assumes that support for modem control implies support for CLOCAL clear (see POSIX.1, line 416), CLOCAL set, HUPCL set (see POSIX.1, lines 409-411), and HUPCL clear.

PCTS_GTI_BUFFERS_OUTPUT (7.1.1.8[11])
If the implementation provides buffered output for general terminal devices, then specify `'1'` otherwise specify `'0'`.

GTI_LOCKS (6.5.2.2[02])
If the implementation supports locks on PORT_TTY and PORT_LOOP, then specify `'1'` otherwise specify `'0'`.

GTI_BAUD_RATES
This parameter specifies the input/output baud rates that your implementation will support for the PORT_TTY and PORT_LOOP ports. Tests are performed to ensure this data is accurate and complete.

Do not specify B0 as a GTI_BAUD_RATES parameter.

TIME_ACQUIRE_XMITTED_DATA
Implementations vary as to the time required for data to arrive at the input queue via the loop back cable after being transmitted. This parameter is the delay in tenths of seconds to be used when needed to ensure the data has arrived.

GTI_LINE_CLOSE_DELAY
Telephone equipment may not immediately terminate a connection when a hangup occurs. Implementations may disconnect the TTY line and inhibit another *open*() on that TTY line for an implementation defined period. This period is made long enough to guarantee that the connection has been terminated by the telephone switching equipment. If this period is not long enough, the telephone connection may not be broken and a successive *open*() may complete with the old connection.

The GTI_LINE_CLOSE_DELAY parameter allows an implementation to define a delay time in seconds that will be honored for those instances in this PCTS where a *close*() on the GTI line is performed.

GTI_START_STOP (7.1.1.9[12])
: If the implementation supports changing the START and STOP special characters on PORT_TTY and PORT_LOOP, then specify '1' otherwise specify '0'.

PCTS_ECHOE
: Control sequence to erase terminal's previous character.

PCTS_ECHOK
: Control sequence to erase terminal's current line.

A_D_TO_P_WHEN_C_P_T (7.1.1.3[06])
: If after a controlling process terminates, access to the terminal by processes in the terminated process's session is denied, then specify '1' otherwise specify '0'.

ANOTHER_LOCALE (8.1.3.2[01])
: To adequately test a number of the assertions for *setlocale*() when an implementation does support additional locales, another *locale* is needed other than "C" and "POSIX". Subclause 8.1.2.2 of the implementations PCD specifies any additional locales that are supported. If no additional locales are supported specify "" otherwise specify the name of one of the additional locales supported.

APP_ACCESS (5.6.3.2[01])
: If the implementation provides a method for associating with a process the appropriate privileges to override the file access control mechanism, then specify '1' otherwise specify '0'.

APP_CHOWN (5.6.5.2[01])
: If the implementation provides a method for associating with a process the appropriate privileges to change file owner, then specify '1' otherwise specify '0'.

APP_UTIME (2.2.2.4[01])
: If the implementation permits appropriate privilege processes to change file times, then specify '1' otherwise specify '0'.

GROUP_ID_OF_PARENT_DIR (5.3.1.2[02])
: This parameter informs the PCTS how to test that the group ID of newly created files and directories are set to the group owner of its parent directory. If the implementation requires the S_ISGID mode set on the parent directory, then specify GROUP_ID_OF_PARENT_DIR as '1' otherwise specify as '0'.

EINVAL_ACCESS (5.6.3.4[3])
: If the implementation supports [EINVAL] for *access*(), then specify the value for the *amode* argument of *access*() otherwise specify '0'.

EINVAL_SETPGID
: If the implementation supports [EINVAL] for *setpgid*() when the *pgid* argument is invalid, then specify a value for the *pgid* argument otherwise specify '0'.

EINVAL_SYSCONF
: Specify a value for the *name* argument to *sysconf*() to generate [EINVAL] when *sysconf*() is called.

EINVAL_WAITPID
: Specify the value for the *options* argument to *waitpid*() to generate [EINVAL] when *waitpid*() is called.

INVALID_GID_VALUE, INVALID_PID_VALUE,

INVALID_PGRP_ID_VALUE, & INVALID_UID_VALUE
If the implementation supports an invalid value for the process atribute stated, then specify an invalid value for that process attribute otherwise specify `'0'`.

EBADF_CLOSEDIR (5.1.2.4.4[03]),

EBADF_READDIR (5.1.2.2.4[06]),

EDEADLK_FCNTL (6.5.2.4[04]), and

EMFILE_OPENDIR (5.1.2.1.4[02])
If the implementation supports the detection of the errno (leading part of parameter's name) for the function (trailing part of parameter's name), then specify `'1'` otherwise specify `'0'`.

PCD_CREAT_LINK_COUNT (5.1.1[01]),

PCD_DIR_TYPE (5.1.1[01]),

PCD_LINK_FILE_SYSTEM (5.3.4.2[01]),

PCD_LINK_TO_DIRECTORY (5.3.4.2[03]),

PCD_NO_LOCK_FILE_TYPE, (6.5.2.2[02]), and

PCD_WRITE_PERM_TO_RENAME (5.5.3.2[01])
These items are specified in IEEE Std 2003.1-1992 as PCD.1 determinable conditional features and are required by FIPS 151-2 (item `'p'`) to be documented if supported. If the PCD.1 documents the conditional feature as supported, then specify `'1'` otherwise specify `'0'`.

PCD_UNLINK_DIRECTORY (5.5.1.2[02])
If the implementation documents support for unlinking a directory, then specify `'1'` otherwise specify `'0'`.

PCTS_CHMOD_SET_IDS (5.6.4.2[02])
If the implementation supports the setting of S_ISUID and S_ISGID by *chmod*(), then specify `'1'` otherwise specify `'0'`.

PCTS_INVALID_OWNER
If the implementation supports an invalid group or owner ID, then specify `'1'` otherwise specify `'0'`.

PCTS_INVALID_SIGNAL
If the implementation supports detection of an invalid or unsupported signal number, then specify `'1'` otherwise specify `'0'`.

PCTS_INVALID_SIGNAL_NUMBER
If the specification for PCTS_INVALID_SIGNAL was `'1'`, then specify a value for an invalid or unsupported signal number.

APPENDIX C

NIST-PCTS:151-2 **Configured Parameters**

When the POSIX.1 limit values exceed practical testing constraints, the following ‘PCTS_’ limit values are used for testing as recommended by the IEEE Std 2003.1-1992.

PCTS Configuration Variables

PCTS Limit Name	POSIX.1 Limit	Test Value
PCTS_ARG_MAX	{ARG_MAX}	16384
PCTS_CHILD_MAX	{CHILD_MAX}	256
PCTS_LINK_MAX	{LINK_MAX}	256
PCTS_LOCK_MAX	---	100
PCTS_MAX_CANON	{MAX_CANON}	1020
PCTS_MAX_INPUT	{MAX_INPUT}	1020
PCTS_NAME_MAX	{NAME_MAX}	2048
PCTS_OPEN_MAX	{OPEN_MAX}	256
PCTS_PATH_MAX	{PATH_MAX}	4096
PCTS_PIPE_BUF	{PIPE_BUF}	32767
PCTS_TZNAME_MAX	{TZNAME_MAX}	256

APPENDIX D

Sample Installation Report

```
          NIST-PCTS:151-2, Version 1.0, 4/15/93


STD/DC     Apr 15 12:12:23 1993
  ./src/cfgetispeed
     c89 -c -I/u/login_dir/NIST-PCTS/include cfgetispeed.c
     c89 -o ../../bin/cfgetispeed cfgetispeed.o
          /u/login_dir/NIST-PCTS/lib/sv_tools.a
          /u/login_dir/NIST-PCTS/lib/sv_base.a
          /u/login_dir/NIST-PCTS/lib/sv_sys.a
          /u/login_dir/NIST-PCTS/lib/tssf.a
     rm -f cfgetispeed.o
     c89 -c -I/u/login_dir/NIST-PCTS/include 7.1.3c_syn.c
     c89 -o ../../bin/7.1.3c_syn 7.1.3c_syn.o
          /u/login_dir/NIST-PCTS/lib/sv_tools.a
          /u/login_dir/NIST-PCTS/lib/sv_base.a
          /u/login_dir/NIST-PCTS/lib/sv_sys.a
          /u/login_dir/NIST-PCTS/lib/tssf.a
     rm -f 7.1.3c_syn.o
     c89 -c -I/u/login_dir/NIST-PCTS/include 7.1.3c_04.c
     c89 -o ../../bin/7.1.3c_04 7.1.3c_04.o
          /u/login_dir/NIST-PCTS/lib/sv_tools.a
          /u/login_dir/NIST-PCTS/lib/sv_base.a
          /u/login_dir/NIST-PCTS/lib/sv_sys.a
          /u/login_dir/NIST-PCTS/lib/tssf.a

     Create load module using sv_base_M.a library.
     c89 -o ../../bin/7.1.3c_04_M 7.1.3c_04.o
          /u/login_dir/NIST-PCTS/lib/sv_tools.a
          /u/login_dir/NIST-PCTS/lib/sv_base_M.a
          /u/login_dir/NIST-PCTS/lib/sv_sys.a
          /u/login_dir/NIST-PCTS/lib/tssf.a
     rm -f 7.1.3c_04.o
```

APPENDIX E

Sample Raw Journal Report

```
1. Function:
     cfgetispeed[004]

1.   Preparation[4000]
     Fork a subtest

1.<p1>.1.1. Function:
     cfgetispeed[004]

1.<p1>.1.1. Abstract
     2

1.<p1>.1.1.1. Testing[4002]
     Assign of "cfgetispeed(termios_p)" to variable of type "speed_t."

1.<p1>.1.1.2. Testing[4002]
     Compare "cfgetispeed(termios_p)" to variable of type "speed_t."

1.<p1>.1.1.3. Testing[4002]
     Negation of integral type "cfgetispeed(termios_p)" is acceptable.

1.<p1>.1.2. Abstract
     3

1.<p1>.1.2.1. Preparation[4003]
     Generate system command to extract macro.

1.<p1>.1.2.2. Preparation[4003]
     Locate text for macro cfgetispeed().

1.<p1>.1.2.3. Testing[4003]
     Macro cfgetispeed(), protects its result value with extra
     parentheses when necessary.
          cfgetispeed(): (((const struct termios *)->c_cflag&0x000f0000)>>16)

1.<p1>.1.3. Abstract
     1

1.<p1>.1.3.1. Testing[4001]
     Function prototype correct.
```

```
2.   Preparation[4000]
     Fork a subtest

2.<p1>.2.1. Function:
     cfgetispeed[004]

2.<p1>.2.1. Abstract
     4

2.<p1>.2.1.1. Preparation[4004]
     Establish PORT_TTY as the control terminal and PORT_LOOP.

2.<p1>.2.1.2. Testing[4004]
     Call to cfgetispeed() returns the input baud rate stored
     in the "termios" structure pointed to by the termios_p
     parameter.

2.<p1>.2.1.3. Cleanup
     Close terminal devices.

MACRO Processing

3.   Preparation[4000]
     Fork a subtest

3.<p1>.3.1. Function:
     cfgetispeed[004]

3.<p1>.3.1. Abstract
     4

3.<p1>.3.1.1. Preparation[4004]
     Establish PORT_TTY as the control terminal and PORT_LOOP.

3.<p1>.3.1.2. Testing[4004]
     Call to cfgetispeed() returns the input baud rate stored
     in the "termios" structure pointed to by the termios_p
     parameter.

3.<p1>.3.1.3. Cleanup
     Close terminal devices.
```

APPENDIX F

Sample Output Report

NIST-PCTS:151-2
Release: Version 1.0, 4/15/93
Based on: IEEE Std 2003.1-1992
Hardware Type: XYZ
Sysname: ABCD, Release: 1, Version: 3
Date: Apr 15 15:55:55 1993

cfgetispeed	(004)	
001	PASS	
002	PASS	
003	PASS	
004	PASS	PASS

APPENDIX G

NIST-PCTS:151-2 **Update Policy**

G.1 Code Changes

Procedures for handling user selected code changes are:

1. Send suggested code changes via e-mail to *151-2@sst.ncsl.nist.gov*. When a final disposition is determined a response will be provided to the submitter. Code changes must conform to FIPS 151-2 and must correct a NIST-PCTS:151-2 deficiency in existing code.
2. If deemed appropriate by the NIST/CSL review committee, suggested code changes will be integrated into the NIST-PCTS:151-2 and evaluated in-house on all NIST/CSL systems used for FIPS 151-2 testing.
3. When testing is completed, the changes will be queued for eventual distribution as a BETA Patch for the Official NIST-PCTS:151-2 to NIST-PCTS:151-2 owners who are e-mail accessible.

G.2 Patches

1. Updates are normally distributed as a shell script file which utilize the utility to automatically update source modules. The update will also generate a text file which specifies the changes performed. When appropriate, an update will be distributed as a complete redistribution of the NIST-PCTS:151-2.
2. All updates distributed to APTLs will be logged. Each APTL on receipt of an update must acknowledge receipt. NIST/CSL will maintain records on notification and acknowledgement.
3. NIST/CSL will provide two types of updates, the BETA Patch and the FIXES Patch. The BETA Patch is configured to be installed on the latest Official NIST-PCTS:151-2. The BETA Patch represents fixes that have been generated and tested by NIST/CSL. These fixes are made available, to the testing community, for final checkout before being integrated into a FIXES Patch.

   BETA Patches are short lived, with a target life of two to eight weeks. If no negative comments have been received, the current BETA Patch will be integrated into an updated FIXES Patch or a complete redistribution.
4. All BETA Patches and FIXES Patches generate an update that records the date the Patch was released.
5. At most, one BETA Patch will exist at any time. An updated BETA Patch, FIXES Patch, or NIST-PCTS:151-2 redistribution terminates the prior BETA Patch (i.e., if the date associated with a FIXES Patch is later than the date associated with a BETA Patch, the BETA Patch no longer applies).
6. An updated FIXES Patch, replaces the prior FIXES Patch and represents an official update of the NIST-PCTS:151-2. The official date of the updated NIST-PCTS:151-2 is automatically reported in the generated test results.

G.3 Installing Patches

Whenever a FIXES Patch is distributed by NIST/CSL, the NIST-PCTS:151-2 to be used is the combination of the last NIST-PCTS:151-2 distribution and the FIXES Patch file. The procedures for generating the Official NIST-PCTS:151-2 are as follows:

1. Load the last NIST-PCTS:151-2 distribution. The commands provided below perform this procedure on our implementations but may vary on your implementation.
   ```
    tester_login_directory
   cpio -icBdmu[v] <distribution_file
   ```
2. After removing any extraneous information added by the e-mail distribution, execute the FIXES Patch file. This file requires utility to produce the current Official NIST-PCTS:151-2. A text file will be generated which provides a description of the changes performed.

   When executing the Patch file, save the journal output produced by to ensure that all patches were performed. In the Bourne shell execute the following command.
   ```
   Patch_file 2>p_patch_rslts
   ```
   The sources for are distributed with the NIST-PCTS:151-2.

   The revision level of the NIST-PCTS:151-2 will be updated to specify the FIXES Patch used. This revised level is reported in the output reports generated.
3. Create the new Official NIST-PCTS:151-2. The commands provided below perform this procedure on our implementations but may vary on your implementation.
   ```
   NIST-PCTS -print | cpio -omcB > special_file
   ```
4. Use the newly created Official NIST-PCTS:151-2 for all future testing.

APPENDIX H

Testing Miscellany

- The absence of a required data file will be reported by `setaccess`.
- If you execute `verify` before **setaccess**, data files may be inadvertantly created, modified, or deleted. The tester should repeat the test procedures starting at Step <1>.
- When modules are recompiled after `./bin/setaccess` has been run, some implementations may fail to load specific modules. To avoid this possibility, execute `./bin/clraccess` prior to recompilation.
- Although the NIST-PCTS:151-2 is designed to continue if a test FAILs, a failure of a test may cause a later test in the same module to FAIL that should have PASSed. Tests are not necessarily executed in ascending order. The Raw Journal Report provides the order of testing. FAILures should be resolved in the order in which they were tested.
- A failure or an abort of an assertion which make use of the general terminal interface devices may cause the implementation's driver for that device to hang. If rerun of prior assertions, which use the general terminal interface devices, now FAIL where they were previously reporting PASS a reboot of the operating system may be necessary to clear the driver.
- If assertion results are grossly inconsistent with prior results, a process of a prior PCTS run may be still active. Perform a process report on currently active processes and delete any old PCTS active processes that are found.
- For failure of assertions associated with permission bits, ensure the command `umask 0` was in affect when the test was processed.

APPENDIX I

Sources of Documents

American National Standards Institute (ANSI), 1430 Broadway, New York, NY 10018, 212/354-3300, FAX 212/302-1286.

- ISO/IEC 9899: Information Technology—Programming Language—C.

Institute of Electrical and Electronics Engineers (IEEE), 345 East 47th Street, New York, NY 10017-2394, telephone 212/705-7900, Standards Orders 800/678-4333.

- International Standard ISO/IEC 9945-1, Information Technology— Portable Operating System Interface (POSIX)— Part 1: System Application Program Interface (API) [C Language]
- IEEE Standard for Information Technology—Test Methods for Measuring Conformance to POSIX, IEEE Std 1003.3-1991
- IEEE Standard for Information Technology—Test Methods for Measuring Conformance to POSIX.1, IEEE Std 2003.1-1992

National Institute of Standards and Technology (NIST), Computer Systems Laboratory, NIST POSIX Certification Authority, Bldg 225 Room B266, Gaithersburg, MD 20899, 301/975-3295, FAX 301/590-0932.

- NIST POSIX Testing Policy— General Information.
- NIST POSIX Testing Policy— Certificate of Validation Requirements for FIPS 151-2.
- NIST-PCTS:151-2 Distribution contains: (1) NIST POSIX Testing Policy— General Information, (2) NIST POSIX Testing Policy— Certificate of Validation Requirements for FIPS 151-2, (3) NIST-PCTS:151-2 — Installation and Testing Guide, and (4) NIST-PCTS:151-2.

National Technical Information Service (NTIS), 5285 Port Royal Road, Springfield, VA 22161, 703/487-4650, FAX 703/321-8547.

- Federal Information Processing Standards Publication 151-2 (FIPS PUB 151-2), Portable Operating System Interface (POSIX)— System Application Program Interface [C Language], 1993 May 12. (Supersedes FIPS PUB 151-1 — 1990, March 28)
- Federal Information Processing Standards Publication 160 (FIPS PUB 160), 'C', March 13, 1991 with change #1, August 24, 1992.

CONTENTS