Sep 1, 2003

Software Write Block Tool Specification & Test Plan

Version 3.0 September 1, 2003

National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

Abstract[1]

This document defines requirements for *hard drive software write block* (SWB) tools used in computer forensics investigations and the test methods used to ascertain whether a specific tool meets the requirements.

The requirements are used to derive assertions that will be tested. The assertions are described as general statements of conditions that can be checked after a test is executed. Each assertion generates one or more test cases consisting of a test protocol and the expected test results. The test protocol specifies detailed procedures for setting up the test, executing the test, and measuring the test results.

The requirements and test methods were developed by a focus group of individuals who are expert in the use of hard drive software write block tools and have performed investigations that have depended on the results of these tools. As this document evolves through comments from the focus group and others, new versions will be posted to the web site at http://www.cftt.nist.gov.

[1] **Certain trade names and company products are mentioned in the text or identified. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the products are necessarily the best available for the purpose.**

Contents

List of Tables

List of Figures

1 Introduction

There is a critical need in the law enforcement community to ensure the reliability of computer forensic tools. A capability is required to ensure that forensic software tools consistently produce accurate and objective test results. The goal of the Computer Forensic Tool Testing (CFTT) project at the National Institute of Standards and Technology (NIST) is to establish a methodology for testing computer forensic software tools by development of general tool specifications, test procedures, test criteria, test sets, and test hardware. The results provide the information necessary for toolmakers to improve tools, for users to make informed choices about acquiring and using computer forensics tools, and for interested parties to understand the tools capabilities. Our approach for testing computer forensic tools is based on well-recognized international methodologies for conformance testing and quality testing. This project is further described at http://www.cftt.nist.gov/.

The CFTT is a joint project of the National Institute of Justice, the research and development organization of the U.S. Department of Justice; the National Institute of Standards and Technology Office of Law Enforcement Standards and Information Technology Laboratory; and other agencies, such as the Technical Support Working Group. The entire computer forensics community participates in the development of the specifications and test methods by commenting on drafts as they are published on the website.

The central requirement for a sound forensic examination of digital evidence is that the original evidence must not be modified, i.e., the examination or capture of digital data from the hard drives of a seized computer must be performed so that the drive contents are not changed. The investigator follows a set of procedures designed to avoid any modification of original data, including prevention of the execution of any program that might modify the drive contents, using a software tool to block modification of a drive, or using a hardware device to block modification of a drive.

2 Purpose

This document defines requirements for *hard drive software write block* (SWB) tools used in computer forensics investigations and the test methods used to ascertain whether a specific tool meets the requirements.

The requirements are used to derive assertions that will be tested. The assertions are described as general statements of conditions that can be checked after a test is executed. Each assertion generates one or more test cases consisting of a test protocol and the expected test results. The test protocol specifies detailed procedures for setting up the test, executing the test, and measuring the test results.

The requirements and test methods were developed by a focus group of individuals who are expert in the use of hard drive software write block tools and have performed investigations that have depended on the results of these tools. As this document evolves through comments from the focus group and others, new versions will be posted to our web site at http://www.cftt.nist.gov.

3 Scope

The scope of this specification is limited to software tools that protect a hard drive attached to a PC from unintended modification. The test plan is currently confined to tools that protect drive access through the interrupt 0x13 BIOS interface of a PC. This interface is also listed as *INT 13h* and is defined as "A BIOS interrupt service that provides a protocol independent method for addressing floppy, hard drive, and other storage devices[NCITS 347:2001] Not included are tools that protect a hard drive from modification through other mechanisms such as hardware write blocking devices. However, the specifications are general and could be adapted to other types of software write blocking tools. If software write blocking tools other than interrupt 0x13 based tools are tested (e.g., windows device driver based tools), an additional test plan, specific to the type of tool tested will be developed and published as an addendum to this document. Definitions for hard drive related terms can be found in NCITS 347:2001 "American National Standard for Information Technology – BIOS Enhanced Disk Drive Services."

4 References

[NCITS 347:2001] ANSI NCITS 347-2001 BIOS Enhanced Disk Drive Services**,** ANSI 11 West 42nd Street, New York, NY 10036.

[Gilluwe] F. van Gilluwe, The Undocumented PC, Second ed. Addison—Wesley, New York (1997).

[Micro] The Hard Disk Technical Guide, 12th edition, MicroHouse, Micro House International, Inc. Boulder, CO, 1996.

[Norton] P. Norton, The Peter Norton Programmer's Guide to the IBM PC, Microsoft Press, Bellevue, Washington, 1985.

[Phoenix] PhoenixBIOS 4.0 Revision 6 User's Manual, Phoenix Technologies Ltd., 411 E. Plumeria, San Jose, CA 95134 (2000), http://www.phoenix.com/resources/userman.pdf.

5 Technical Background

This section provides technical background for a discussion of write blocking technology. The first subsection gives definitions and references for several physical drive interfaces. The second subsection presents an overview of how hard drives are physically attached to a computer and then accessed by programs running on the computer. The third subsection defines terminology related to software write block tools. An example illustrating how the terminology relates to an actual PC is presented in Appendix B.

5.1 Definitions

Firewire: colloquial term referring to an external bus standard that supports data transfer rates of up to 400Mbps (IEEE Standard 1394a) and 800Mbps (IEEE Standard 1394b). The term ‘FireWire’ was trademarked by Apple.

Integrated Drive Electronics/AT Attachment (IDE/AT) Interface: A colloquial term for interface standards developed by T13. Technical Committee T13 is responsible for all interface standards relating to the AT Attachment (ATA) storage interface utilized as the disk drive interface on personal and mobile computers. T13 is a Technical Committee for the InterNational Committee on Information Technology Standards (INCITS) [http://www.incits.org/]. INCITS is accredited by, and operates under rules approved by, the American National Standards Institute (ANSI) [http://www.ansi.org/]. [see Addendum for ATA interface commands]

Interface: A shared boundary defined by the characteristics of that boundary. The interface may be described at the physical level, at the software level, or as purely logic operations. For example, characteristics of the boundary may include the identification of any physical interconnections, description of signal exchanges across the boundary, or specification of functions performed on each side of the boundary.

Small Computer System Interface (SCSI): A colloquial term for interface standards developed by T10. Technical Committee T10 is responsible for SCSI Storage Interfaces and SCSI architecture standards (SAM, SAM-2, and SAM-3), which are used by SCSI, SAS, Fibre Channel, SSA, IEEE 1394, USB, and ATAPI. T10 is a Technical Committee of the InterNational Committee on Information Technology Standards (INCITS) [http://www.incits.org]. INCITS is accredited by, and operates under rules that are approved by, the American National Standards Institute (ANSI) [http://www.ansi.org]. [see Addendum for SCSI interface commands]

Universal Serial Bus (USB): A colloquial term referring to external bus standards that support data transfer rates of up to 480Mbps for high-speed connection of peripheral equipment to microcomputers.

ASPI: Short for Advanced SCSI Programming Interface, an interface specification developed by Adaptec, Inc. for sending commands to a SCSI host adapter. ASPI has become a de facto standard that enables programmers to develop applications and drivers that work with all ASPI-compatible SCSI adapters.

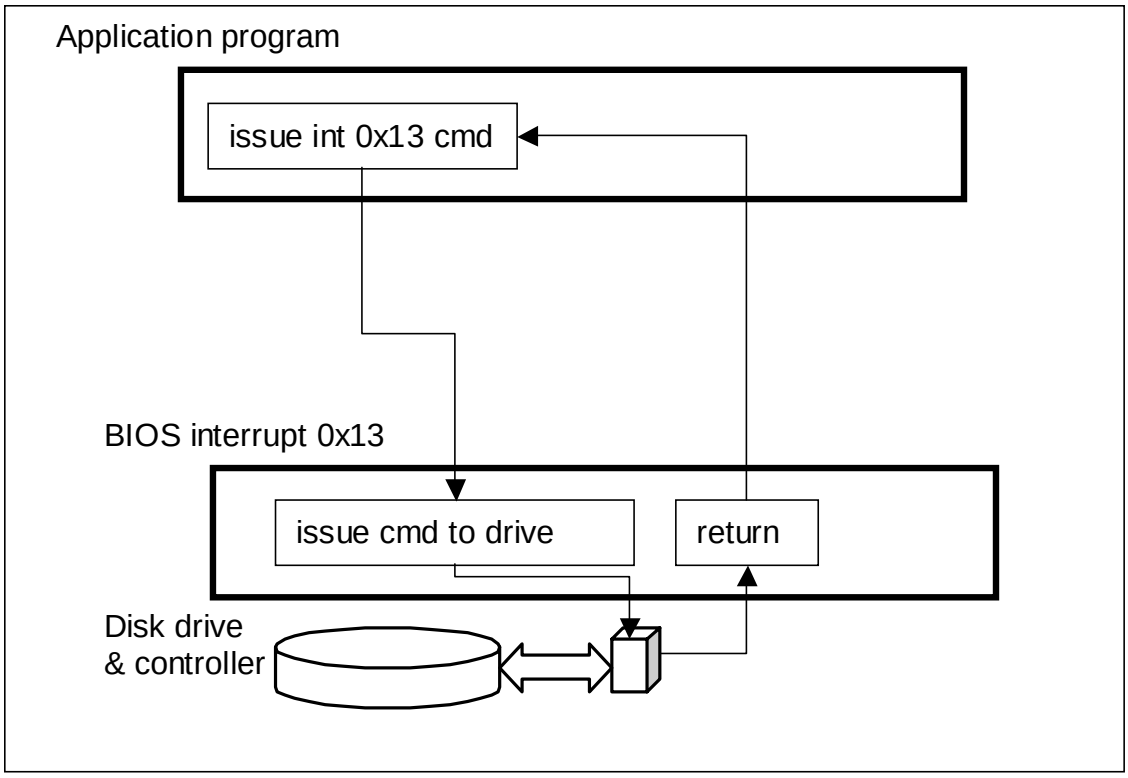
5.2 Hard Drive Attachment and Access

Before a hard drive can be used it must be physically attached to a computer. A hard drive is attached to a computer by one of several available physical interfaces. A drive is usually connected by a cable to an interface controller located either on the mother board or on a separate adapter card. The most common physical interface is the ATA (AT Attachment) or IDE (Integrated Drive Electronics) interface. This includes variants such as EIDE (Enhanced IDE) or ATA-2, ATA-3, etc. Some other physical interfaces include, but are not limited to SCSI (Small Computer System Interface), IEEE 1394 (also known as FireWire or i-Link), and USB (Universal Serial Bus).

All access to a drive is accomplished by commands sent from a computer to a drive through the interface controller. However, since the low level programming required for direct access through the interface controller is difficult and tedious, each operating system usually provides other access interfaces. For example, programs running in the DOS environment can, in addition to direct access via the drive controller, use two other interfaces: DOS service interface (interrupt 0x21) or BIOS service interface (interrupt 0x13). The DOS service operates at the logical level of files and records while the BIOS service operates at the physical drive sector level. More complex operating systems, for example Windows XP or a UNIX variant (e.g., Linux), may disallow any low level interface (through the BIOS or the controller) and only allow user programs access to a hard drive through a device driver, a component of the operating system that manages all access to a device.

Using the interrupt 0x13 interface for hard drive access is illustrated in Figure 5-1. An application program issues an interrupt 0x13 command. The interrupt transfers control to the interrupt 0x13 routine in the BIOS. The BIOS routine issues commands, ATA or SCSI as appropriate, directly to the hard drive controller. The device does the requested operation and returns the result to the BIOS and then to the application program.

Figure 5-1 Drive Access Through the 0x13 BIOS Interface



5.3 Technical Terminology

A hard drive software write block tool replaces or monitors a hard drive *access interface* on a general purpose host computer with hard drives attached by a physical interface. A hard drive access interface is defined as a method used by a program to access a hard drive. For a program to access a drive, the program issues a high level command to the access interface that is translated by the access interface into the corresponding low level command that is sent to the drive through the physical interface controller. For each command issued, the access interface indicates command results (e.g., command completion, error status) by a *return value*. A hard drive software write block tool operates by monitoring drive I/O commands sent from the PC through a given access interface. Any commands that could modify a hard drive are intercepted (i.e., blocked) and not passed on to the hard drive controller. The most common access interfaces currently found are as follows: hard drive device driver, interrupt 0x13 BIOS (Basic Input Output Services), ATA (AT Attachment) direct controller, ASPI (Advanced SCSI Programming Interface), USB (Universal Serial Bus) and IEEE 1394 (also known as Firewire). Each interface has its own command set and access protocol. The command set for a given interface can be partitioned into the following categories:

- **Write:** commands that transfer data from the computer memory to the drive.
- **Configuration:** commands that change how the drive is presented to the host computer. These commands often destroy data on the drive or make data inaccessible.
- **Read:** commands that transfer data from the drive to the computer memory.
- **Control:** commands that request the drive to do a nondestructive operation, for example: *reset* or *seek*.
- **Information:** commands that return information about the drive.
- **Miscellaneous:** commands that do not fit into the other categories.

Appendix A presents two tables: categorizations of the typical interrupt 0x13 BIOS commands, and a catalog of commands and widely known extensions to the typical command set.

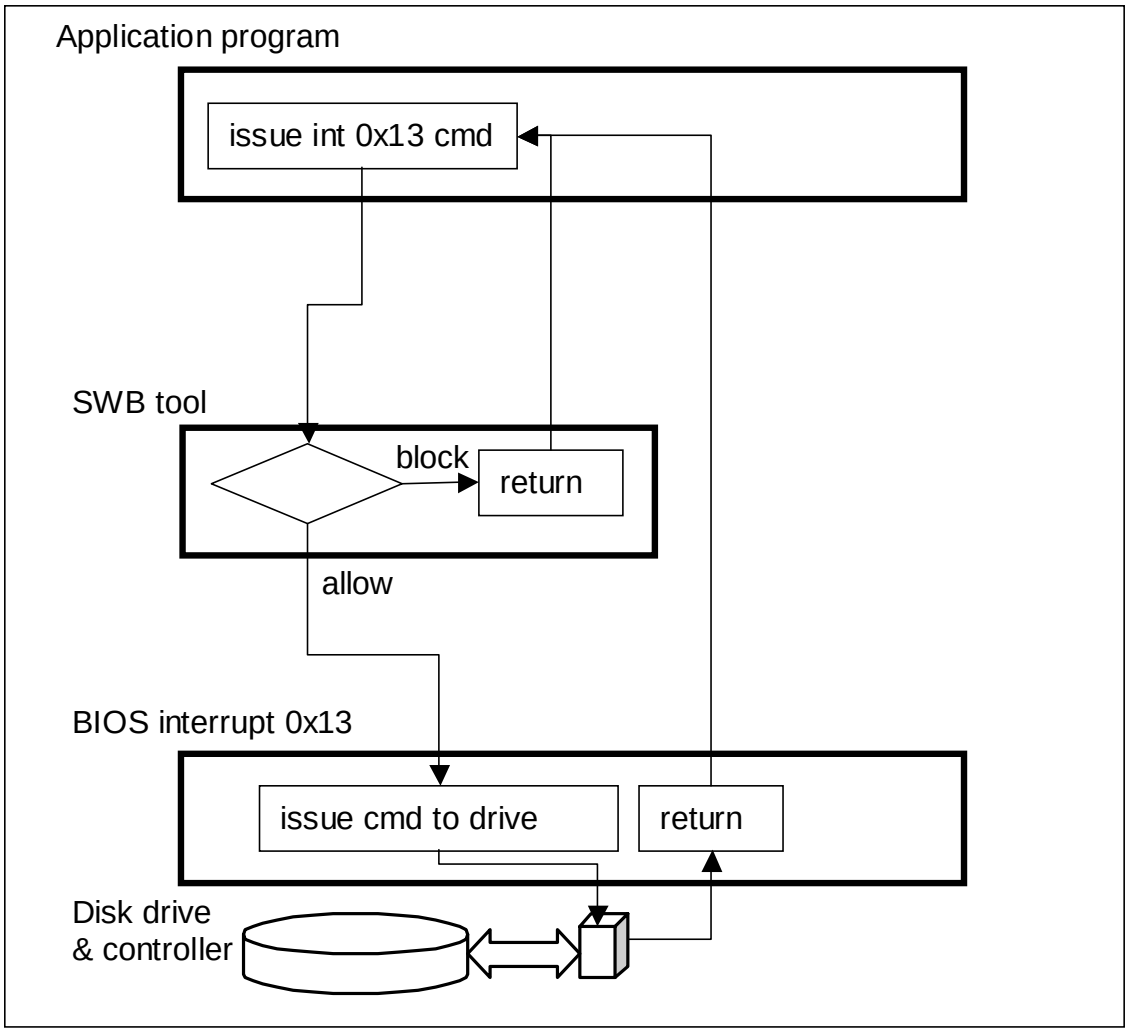
The following terms are defined for convenience in specifying the tool requirements.

- **Covered interface:** a drive access interface that is controlled by the tool.
- **Covered drive**: a drive attached to a covered interface.
- **Protected drive:** a drive designated for protection from modification when accessed by a covered interface.
- **Unprotected drive:** a drive that is not protected from modification through a specified access interface.
- **Blocked command to a drive:** a command issued by an application program that is intercepted by a SWB tool such that neither the issued command nor some other command is sent to the interface used to access the drive. A command that is not blocked is not altered in any way.

Use of a SWB tool changes the normal operation of the interrupt 0x13 interface. Figure 5-2 illustrates SWB tool operation.

1. The SWB tool is executed. The SWB tool saves the current interrupt 0x13 routine entry address and installs a new interrupt 0x13 routine.
2. The application program initiates a drive I/O operation by invoking interrupt 0x13. The replacement routine installed by the SWB tool intercepts the command.
3. The SWB tool determines if the requested command should be blocked or if the command should be allowed.
4. If a command is blocked, the SWB tool returns to the application program without passing any command to the BIOS I/O routines. Depending on SWB tool configuration either **success** or **fail** is returned for the command status.
5. If the command is allowed (not blocked), the command is passed to the BIOS and the BIOS I/O routine issues required I/O commands (ATA, SCSI or other) to the drive controller so that the desired I/O operation occurs on the hard drive.
6. Results are returned to the application program.

Figure 5-2 SWB Tool Operation



6 Requirements

This section presents mandatory requirements for all software write block tools and a list of optional requirements that some software write block tools may provide.

The informal hard drive software write block tool requirements are the following:

- The tool shall not allow a protected drive to be changed.
- The tool shall not prevent obtaining any information from or about any drive.
- The tool shall not prevent any operations to a drive that is not protected.

The three informal requirements are the essence of a software write blocking tool: protect the evidence from alteration while allowing a complete examination of the evidence. It should be noted that it may be acceptable to not block a given command in one environment, but vital to block the same command in another environment. In other words, a SWB tool may be safe to use in one environment but not in other environments. For example, an int 13 extended write command (43h) must be blocked on contemporary computers, but a SWB tool that does not block this command might be acceptable on a computer with a legacy BIOS (i.e., a BIOS that does not implement command 43h). The

requirements stated in this document are intended for a general use tool that allows safe access to a hard drive in almost all environments.

6.1 Requirements for Mandatory Features

A formal statement of these requirements follows:

SWB-RM-01. The tool shall block any commands to a protected drive in the write, configuration, or miscellaneous categories.
SWB-RM-02. The tool shall not block any commands to a protected drive in the read, control or information categories.
SWB-RM-03. The tool shall give an indication to the user that the tool is active.
SWB-RM-04. The tool shall report all drives accessible by the covered interfaces.
SWB-RM-05. The tool shall report the protection status of all drives.
SWB-RM-06. The tool shall, if so configured, adjust the return value of any blocked commands to indicate that the operation was carried out successfully even though the operation was blocked.
SWB-RM-07. The tool shall, if so configured, adjust the return value of any blocked commands to indicate that the operation failed.
SWB-RM-08. The tool shall not block any commands to an unprotected drive.

6.2 Requirements for Optional Features

The following requirements are defined for tool features that might be implemented for some SWB tools. If a tool provides the optional feature, the tool is tested as if the requirement were mandatory. If the tool does not provide the capability defined, the requirement does not apply.

The following optional features are identified:

- The SWB tool can indicate to the user with an audio or visual signal that a command has been blocked.
- The SWB tool can be deactivated after the tool has been executed.
- The SWB tool can be configured to be active during the system boot and shutdown.
- The SWB tool allows the user to select an arbitrary subset of the available hard drives for protection.

A formal statement of these requirements follows:

SWB-RO-01. The tool shall alert the user when a command is blocked, either by an audio or a visual signal.
SWB-RO-02. If after the tool is running and is then successfully deactivated, any protected drives become unprotected, otherwise there is no change to the protection status of the covered drives.
SWB-RO-03. The user shall be able to specify each of the covered drives as either protected or unprotected.
SWB-RO-04. If the tool is active during the operating system boot and shutdown processes, then the tool shall block any commands to the protected drives in the write, configuration, or miscellaneous categories during the boot process and during the shutdown.
SWB-RO-05. If a subset of the covered drives is designated for protection, the tool shall not block commands from any category to the unprotected drives.

7 Assertions

Each assertion specifies a set of conditions that can be tested and the expected results. A traceability matrix relating requirements and assertions is presented in Table C-1.

7.1 Assertions for Mandatory Features

This section lists assertions that all SWB tools should meet.

SWB-AM-01. If a drive is protected and a command from the write category is issued for the protected drive then the tool shall block the command.
SWB-AM-02. If a drive is protected and a command from the configuration category is issued for the protected drive then the tool shall block the command.
SWB-AM-03. If a drive is protected and a command from the miscellaneous category is issued for the protected drive then the tool shall block the command.
SWB-AM-04. If a drive is protected and a command from the read category is issued for the protected drive then the tool shall not block the command.
SWB-AM-05. If a drive is protected and a command from the control category is issued for the protected drive then the tool shall not block the command.
SWB-AM-06. If a drive is protected and a command from the information category is issued for the protected drive then the tool shall not block the command.
SWB-AM-07. If the tool is executed then the tool shall issue a message indicating that the tool is active.
SWB-AM-08. If the tool is executed then the tool shall issue a message indicating all drives accessible by the covered interfaces.
SWB-AM-09. If the tool is executed then the tool shall issue a message indicating the protection status of each drive attached to a covered interface.
SWB-AM-10. If the tool is configured to return *success* on blocked commands and the tool blocks a command then the return code shall indicate successful command execution.
SWB-AM-11. If the tool is configured to return *fail* on blocked commands and the tool blocks a command then the return code shall indicate unsuccessful command execution.

7.2 Assertions for Optional Features

This section lists assertions that are tested if the tool supports the relevant feature or option.

SWB-AO-01. If a subset of all covered drives is specified for protection, then commands from the write category shall be blocked for drives in the selected subset.
SWB-AO-02. If a subset of all covered drives is specified for protection, then commands from the configuration category shall be blocked for drives in the selected subset.
SWB-AO-03. If a subset of all covered drives is specified for protection, then commands from the miscellaneous category shall be blocked for drives in the selected subset.
SWB-AO-04. If a subset of all covered drives is specified for protection, then commands from the read category shall not be blocked for drives in the selected subset.
SWB-AO-05. If a subset of all covered drives is specified for protection, then commands from the control category shall not be blocked for drives in the selected subset.
SWB-AO-06. If a subset of all covered drives is specified for protection, then commands from the information category shall not be blocked for drives in the selected subset.
SWB-AO-07. If a subset of all covered drives is specified for protection, then no commands from any category shall be blocked for drives not in the selected subset.
SWB-AO-08. If the tool is active during the operating system boot and shutdown processes then no changes are made to any protected drives.
SWB-AO-09. If the tool is active and the tool is then deactivated then no commands to any drive shall be blocked.
SWB-AO-10. If the tool blocks a command then the tool shall issue either an audio or a visual signal.

8 Test Methodology

This section describes the methodology that has been developed to test interrupt 0x13 based SWB tools. First several issues are identified and then an approach is described that addresses the issues.

One simple strategy to determine the effectiveness of a SWB tool would be to install the tool, attempt an operation that should change the hard drive contents, and then examine the drive for any changes. This has the limitation that only effects for the selected command parameters are measured, not the actual commands blocked. If there is no change to the hard drive it is not clear if the command is actually blocked or if the selected command parameters did not cause a

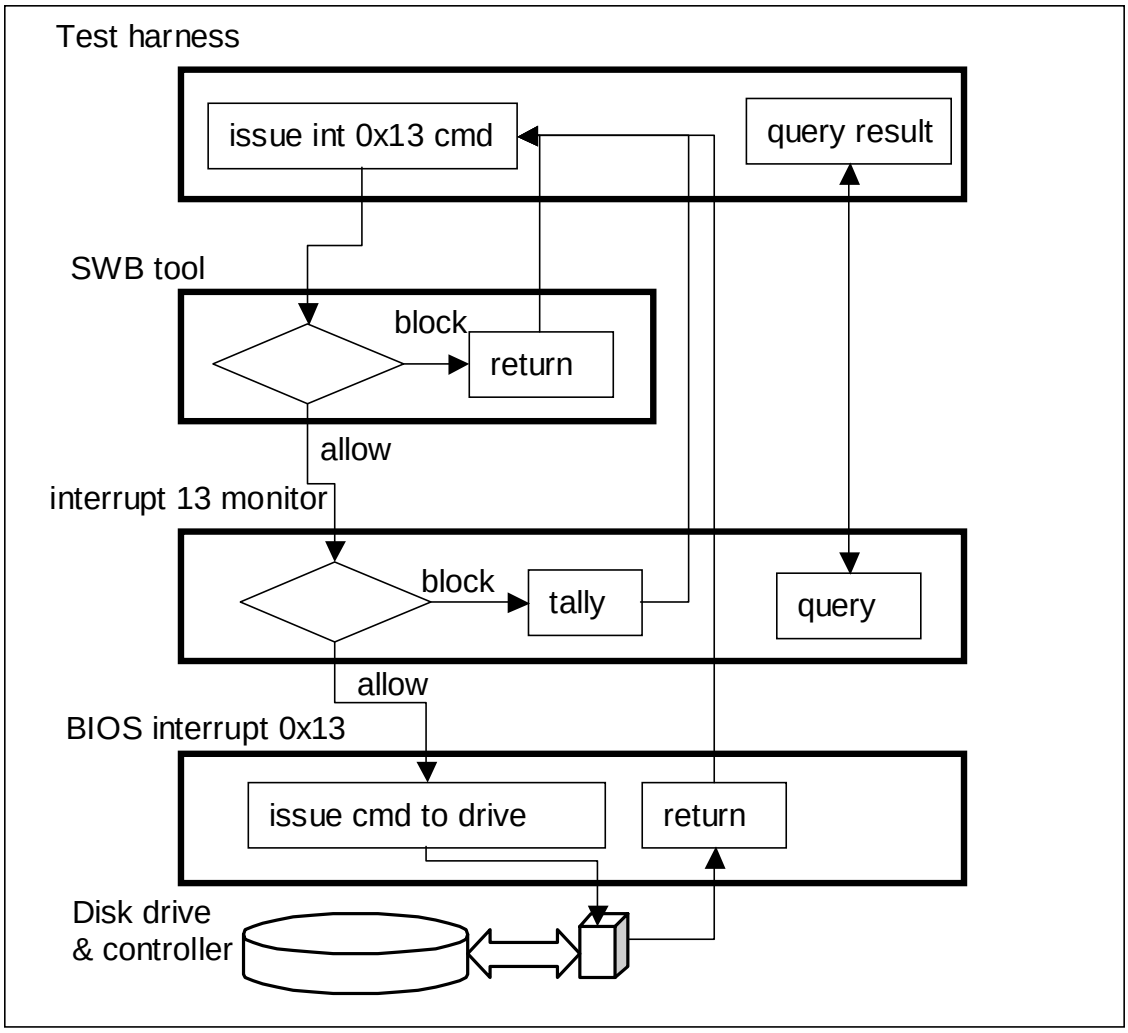
change to the hard drive. A second limitation is that some commands, should only be executed in a *factory setting*. These are commands such as *low level formatting* or *diagnostic* commands that may have subtle and unexpected results. The proper parameter values for some commands are propriety and not easily determined. The user is cautioned that improper parameters may render a drive unusable. The problem is how to safely test if a SWB tool blocks such potentially destructive commands.

A more direct methodology that avoids these limitations has been developed. The normal interrupt 0x13 BIOS routine is replaced with a software monitor that counts the number of times each I/O function is called. The monitor blocks all commands so that any command can be safely issued to a SWB tool. The monitor has a secondary interface to allow a test harness to query the monitor to determine the command functions blocked or allowed by the SWB tool.

Every possible command can be tried and the results observed. First, the monitor is installed to replace the usual interrupt 0x13 processing. The monitor operates in two states: *allow command* or *block/tally*. In the *allow command* state all commands are passed to the hard drive. The *allow command* state permits the SWB tool to initialize during installation. After the SWB tool is installed the monitor state is switched to *block/tally*. In this state, all commands passed by the SWB tool are blocked by the monitor and a tally is kept of all commands seen by the monitor. The monitor query interface allows the test harness to determine which commands are passed by the SWB tool.

Figure 8-1 illustrates the command flow during a test case. After the interrupt 0x13 monitor and the SWB tool are installed, the test harness is executed. The test harness issues every command for a given category. For example, a test of the read command category would issue each command defined for the read command category: 0x02 (read), 0x10 (read long) and 0x42 (extended read). As each command is issued, the SWB tool intercepts the command and either blocks (return with no action) or allows the command (passes on to the interrupt 0x13 monitor). If the command gets to the interrupt 0x13 monitor the tally for the received command is incremented and the monitor returns to the caller (the test harness). After control returns to the test harness, the test harness queries the interrupt 0x13 monitor to get a count of the number of times the issued command has been intercepted by the monitor. If the count is zero, then the SWB tool has blocked the command. Otherwise, a non-zero count indicates that the command was not blocked.

Figure 8-1 Test Harness and Interrupt Monitor Operation



8.1 Test Case Parameters

Each test case is specified in two parts, the settings of conditions (parameters) for the test and the expected results after the test run completes. Test parameters derived from the requirements and assertions are presented in Table 8-1.
The BIOS type (legacy or extended) was considered as a possible parameter but rejected since there is no interaction with the BIOS when the SWB tool decides on blocking or allowing a command. The only interaction between the BIOS and a SWB tool is when the tool is installed to determine the number of attached drives.

Table 8-1 SWB Test Parameters

Parameter Name	Description
Category	The category of command function to be tested. Possible values are: **all, read, write, information, control, configuration, miscellaneous** and **Return-Status**. A value of **all** indicates that all command functions should be tested.
Return	The value returned to the application program from the SWB tool if a command is blocked. Possible values are: **success** and **fail**.
N Drives	The number of hard drives attached to the **covered interface** (interrupt 0x13). Possible values (**one, two, three** and **all**) are explained below.

Parameter Name	Description
Protected	Specification of protected and unprotected hard drives Possible values are explained below in section 8.2.3.

8.2 Parameter Values

This section discusses the possible parameter values for each parameter.

8.2.1 Category

There is no definitive set of interrupt 0x13 commands. There are 256 possible command codes. For a given BIOS, only a few of all possible commands are implemented. There are slight variations for which commands are implemented among BIOS implementations. The listed commands in Table 8-2 are derived from a variety of sources including BIOS user guides [Phoenix], PC architecture books [Gilluwe], [Micro], {Norton], and BIOS related standards [NCITS 347:2001].

The **category** parameter indicates the set of commands to be tested. Each of the 256 possible *int 0x13* command codes, from 0x00 through 0xFF inclusive, are assigned to a unique category so that all possible command codes, including unimplemented commands, are tested. For testing a selected category, all commands in the category are tried. The assignment of 0x13 BIOS drive access commands to categories is presented below (Table 8-2). Any command code not in the table is assigned to the miscellaneous category.

The **Prime Source** column indicates at least one reference to documentation for the given command. Note that not all commands are defined for a typical BIOS [Phoenix]. For example, the majority of command codes in the miscellaneous category are usually undefined and do not represent valid command functions.

It should be noted that the assignment of commands to categories that should be blocked is conservative. While it would be safer to block all unimplemented commands, it might be acceptable to use a tool that does not block a given command (even a write command) if it is known that the command is not implemented in the BIOS environment where the SWB tool is used. For example, a legacy BIOS does not implement the *extended write* command (43h). Therefore, a SWB tool that does not block the extended write command can be safely used with a legacy BIOS. However, other factors may complicate matters. If a SCSI adaptor card and SCSI drive is added, the adaptor card may modify the BIOS environment and add an implementation of the extended write command to the BIOS. With such a SCSI adaptor card installed it would now be unsafe to use a SWB tool that does not block the extended write command.

Table 8-2 Interrupt 0x13 Command Category Assignments

Categorization of Interrupt 0x13 BIOS Commands			
Command	**Code**	**Category**	**Prime Sources**
Format Track	05h	Configuration	[Gilluwe]
Format Track With Bad Sectors	06h	Configuration	[Gilluwe]
Format Cylinder	07h	Configuration	[Phoenix]
Initialize Drive Parameters	09h	Configuration	[Phoenix], [Gilluwe]
ESDI Diagnostic (PS/2)	0Eh	Configuration	[Gilluwe]
ESDI Diagnostic (PS/2)	0Fh	Configuration	[Gilluwe]
Controller RAM Diagnostic	12h	Configuration	[Gilluwe]
Drive Diagnostic	13h	Configuration	[Gilluwe]
Controller Diagnostic	14h	Configuration	[Phoenix], [Gilluwe]
Reset	00h	Control	[Phoenix], [Gilluwe]
Seek Drive	0Ch	Control	[Phoenix], [Gilluwe]
Alternate Drive Reset	0Dh	Control	[Phoenix], [Gilluwe]
Recalibrate Drive	11h	Control	[Phoenix], [Gilluwe]
Extended Seek	47h	Control	[Phoenix], [NCITS 347:2001], [Gilluwe]
Get Last Status	01h	Information	[Phoenix], [Gilluwe]
Verify Sectors	04h	Information	[Phoenix], [Gilluwe]

Categorization of Interrupt 0x13 BIOS Commands			
Command	**Code**	**Category**	**Prime Sources**
Read Drive Parameters	08h	Information	[Phoenix], [Gilluwe]
Test Drive Ready	10h	Information	[Phoenix], [Gilluwe]
Read Drive Type	15h	Information	[Phoenix], [Gilluwe]
Check Extensions Present	41h	Information	[Phoenix], [NCITS 347:2001], [Gilluwe]
Verify Sectors	44h	Information	[Phoenix], [NCITS 347:2001], [Gilluwe]
Get Drive Parameters	48h	Information	[Phoenix], [NCITS 347:2001], [Gilluwe]
Read Sectors	02h	Read	[Phoenix], [Gilluwe]
Read Long Sector	0Ah	Read	[Phoenix], [Gilluwe]
Extended Read	42h	Read	[Phoenix], [NCITS 347:2001], [Gilluwe]
Write Sectors	03h	Write	[Phoenix], [Gilluwe]
Write Long Sector	0Bh	Write	[Phoenix], [Gilluwe]
Extended Write	43h	Write	[Phoenix], [NCITS 347:2001], [Gilluwe]
Undefined & Unimplemented	16h-40h, 49h-FFh	Miscellaneous	

8.2.2 N Drives

A typical computer motherboard allows connection of up to four ATA devices. A typical SCSI adapter card allows connection of at least seven devices and up to fifteen. A computer could easily be configured with up to ten hard drives plus a CD-ROM drive. While a typical computer forensics usage would be for a single protected (source) drive and a single unprotected (destination) drive, using all attached drives would be possible.

The challenge for selecting parameter values for **N Drives** is to select usable values for a variety of possible but unknown test environments. The number of drives that can be used on a test machine cannot be known in advance and depends on the test machine available for testing. However, a reasonable estimate can be derived from the following scenarios:

- An examiner boots a PC from a floppy and then images a single drive to removable media such as tape or CD.
- An examiner boots a PC from a floppy and then images or copies a single source drive to another drive.
- An examiner boots from a hard drive and then images or copies a source drive to another drive.
- An examiner uses all drives attached to a system.

These scenarios are not the only ones possible, but they are typical and suggest using the values **one**, **two**, **three** and **all** as possible values to use for **N Drives**, the number of drives to install for a test. With the value **all** representing *install as many drives as possible* in the test machine.

8.2.3 Protected

The optional feature allowing specification of a subset of installed drives for protection specified in assertions SWB-AO-01 through SWB-AO-05 is challenging to test thoroughly. As the number of installed hard drives increases, the number of possible combinations of protected and unprotected drives increases exponentially (2^n–1 combinations for *n* drives). Exhaustive testing of all possible combinations of up to ten drives would require at least 2,000 test cases. However, three protection scenarios that cover a wide range of situations are the following: protect a single drive while imaging the drive to tape, protect a single drive while imaging to a second unprotected hard drive, and imaging drives from a system with several drives requiring protection. This suggests the following heuristics for selecting the number of drives and patterns of protecting drives.

- Test one or two drives with at least one drive protected.
- Test as many drives attached as possible with about the same number of drives protected as unprotected.
- Test as many drives attached as possible with only one drive unprotected.

- Exhaustive testing of all protection patterns for a small (three) number of drives.

The protection patterns in Table 8-3 are designed to yield a variety of patterns of either one drive protected, one drive unprotected, approximately an equal number of protected and unprotected drives, or an arbitrary pattern of protection.

Table 8-3 Possible Values for the *Protected* Parameter

Value Name	Description
odd	Protect the odd numbered drives: 0x81, 0x83, 0x85, etc
even	Protect the even numbered drives: 0x80, 0x82, 0x84, etc
low	Protect the low numbered drives: 0x80, 0x81, etc. Given *n* drives, the first unprotected drive is 0x80 + *n*/2 (using integer division discarding any fraction)
high	Protect the high numbered drives. Given *n* drives, the first protected drive is 0x80 + (*n*/2)
random protected	Select at random (die throw) one drive that has not been used as a single protected drive. If there are no unused drives, select any drive at random.
random unprotected	Select at random (die throw) one drive that has not been used as a single unprotected drive. If there are no unused drives, select any drive at random.
first	Protect drive 0x80.
mid	Given *n* drives protect drive 0x80 + *n*/2 (discard any fraction).
last	Given *n* drives protect drive 0x80 + *n* - 1.
not first	Protect all drives except for 0x80.
not mid	Given *n* drives, protect all drives except for drive 0x80 + *n*/2 (discard any fraction).
not last	Given *n* drives, protect all drives except for drive 0x80 + *n* - 1.
XX…X	Where X is either U or P, explicit specification of protected (P) or unprotected (U) drives. The first U or P indicates the state of drive 0x80, the second letter indicates the state of drive 0x81, etc.

8.2.4 Return

The **return** parameter is specified to the SWB tool on execution to select the status code return value that the SWB tool should cause to be returned to the calling program if a command is blocked by the SWB tool. This parameter is meaningful only if the SWB tool allows specification of a return status code. A value of **success** indicates that the tool is configured to return a value indicating a successful I/O operation even though the I/O command is actually blocked. A value of **fail** indicates that either the tool is configured to return **fail** on blocked commands. If the tool does not allow return code configuration, then this parameter is meaningless and should be ignored.

9 Abstract Test Cases

Abstract test cases describe the combinations of test parameters required to fully test each assertion and the results expected for the given combination of test parameters. The test cases are abstract in that they do not prescribe the exact environment in which the tests are to be performed. They are written at the next level above the environment. This allows different environments to be substituted under the test cases for testing different products and options.

A set of test parameters is chosen to cover the assertions from various aspects. Not all possible tests will be specified since this number could run into the hundreds or thousands based on the combinations of parameters that could be used. Exhaustive testing, in most cases, is not economically feasible. Instead, a subset of parameter values will be used to define the set of test cases needed to evaluate tools against the requirements.

The test cases are described in Table 9-2. The primary goal of test cases 01-06 is to identify the commands blocked by the SWB tool for each installed drive. A secondary goal of cases 01-06 is to test the **return value** feature of the SWB tool for blocked commands.

The primary goal of test cases 07-12 is to investigate the behavior of the SWB tool for commands that should not be blocked.

Test cases 13-36 investigate the behavior of the SWB tool given a protected subset of covered drives. These cases use two different approaches. Cases 13-24 are focused on testing a sample of likely protection patterns for a relatively large number (4 or more) of installed drives. Not all possible combinations of protected and unprotected drives are used since this would create more cases than would be feasible to execute. Test cases 25-36 focus on executing tests on the **read** and **write** command categories for all possible combinations of protecting a subset of three drives. Three drives were chosen as a likely upper limit for the minimum number of drives that must be installed.

Test cases 37 and 38 investigate the capabilities of the tool to protect a drive during the boot and shutdown processes.

Test cases 39 and 40 investigate the ability of the SWB tool to cleanly uninstall.

9.1 Test Case Summaries

The following table gives a summary for each test case.

Table 9-1 Test Case Summaries

SWB-01	Install all drives, configure return code to fail, protect all drives, execute write commands.
SWB-02	Install two drives, configure return code to success, protect all drives, execute write commands.
SWB-03	Install one drive, configure return code to fail, protect all drives, execute configuration commands.
SWB-04	Install all drives, configure return code to success, protect all drives, execute configuration commands.
SWB-05	Install two drives, configure return code to fail, protect all drives, execute miscellaneous commands.
SWB-06	Install one drive, configure return code to success, protect all drives, execute miscellaneous commands.
SWB-07	Install all drives, configure return code to fail, protect all drives, execute read commands.
SWB-08	Install two drives, configure return code to success, protect all drives, execute read commands.
SWB-09	Install one drive, configure return code to fail, protect all drives, execute information commands.
SWB-10	Install all drives, configure return code to success, protect all drives, execute information commands.
SWB-11	Install two drives, configure return code to fail, protect all drives, execute control commands.
SWB-12	Install one drive, configure return code to success, protect all drives, execute control commands.
SWB-13	Install all drives, configure return code to fail, protect with pattern odd, execute write commands.
SWB-14	Install all drives, configure return code to success, protect with pattern low, execute write commands.
SWB-15	Install all drives, configure return code to fail, protect with pattern first, execute configuration commands.
SWB-16	Install all drives, configure return code to success, protect with pattern mid, execute configuration commands.
SWB-17	Install all drives, configure return code to fail, protect with pattern random_p, execute miscellaneous commands.

SWB-18	Install all drives, configure return code to success, protect with pattern not_last, execute miscellaneous commands.
SWB-19	Install all drives, configure return code to fail, protect with pattern last, execute read commands.
SWB-20	Install all drives, configure return code to success, protect with pattern not_mid, execute read commands.
SWB-21	Install all drives, configure return code to fail, protect with pattern high, execute information commands.
SWB-22	Install all drives, configure return code to success, protect with pattern not_first, execute information commands.
SWB-23	Install all drives, configure return code to fail, protect with pattern random_u, execute control commands.
SWB-24	Install all drives, configure return code to success, protect with pattern even, execute control commands.
SWB-25	Install three drives, configure return code to fail, protect with pattern PUU, execute write commands.
SWB-26	Install three drives, configure return code to success, protect with pattern UPU, execute write commands.
SWB-27	Install three drives, configure return code to fail, protect with pattern UUP, execute write commands.
SWB-28	Install three drives, configure return code to success, protect with pattern UPP, execute write commands.
SWB-29	Install three drives, configure return code to fail, protect with pattern PUP, execute write commands.
SWB-30	Install three drives, configure return code to success, protect with pattern PPU, execute write commands.
SWB-31	Install three drives, configure return code to fail, protect with pattern PUU, execute read commands.
SWB-32	Install three drives, configure return code to success, protect with pattern UPU, execute read commands.
SWB-33	Install three drives, configure return code to fail, protect with pattern UUP, execute read commands.
SWB-34	Install three drives, configure return code to success, protect with pattern UPP, execute read commands.
SWB-35	Install three drives, configure return code to fail, protect with pattern PUP, execute read commands.
SWB-36	Install three drives, configure return code to success, protect with pattern PPU, execute read commands.
SWB-37	Install all drives, configure to be active at boot and shutdown, configure return code to fail, protect with pattern odd, execute write commands.
SWB-38	Install all drives, configure to be active at boot and shutdown, configure return code to success, protect with pattern even, execute write commands.
SWB-39	Install all drives, configure return code to fail, protect with pattern high, execute write commands, uninstall, execute all commands.
SWB-40	Install all drives, configure return code to success, protect with pattern low, execute write commands, uninstall, execute all commands.

9.2 Test Case Parameters

The test case parameters specified in Table 9-2 identify conditions that must be created for each test case. The **Criterion** column, explained in more detail in section 9.4, is not a test case parameter, but identifies a criterion for test case selection. A test case is executed if the tool under test implements the set of optional features tied to a given selection criterion.

Table 9-2 Test Case Parameters

Case	Category	N Drives	Protected	Return	Criterion
SWB-01	Write	all	all	fail	Base
SWB-02	Write	two	all	success	return value
SWB-03	Configure	one	all	fail	Base
SWB-04	Configure	all	all	success	return value
SWB-05	Miscellaneous	two	all	fail	Base
SWB-06	Miscellaneous	one	all	success	return value
SWB-07	Read	all	all	fail	Base
SWB-08	Read	two	all	success	return value
SWB-09	Information	one	all	fail	Base
SWB-10	Information	all	all	success	return value
SWB-11	Control	two	all	fail	Base
SWB-12	Control	one	all	success	return value
SWB-13	Write	all	odd	fail	Subset
SWB-14	Write	all	low	success	Subset, return value
SWB-15	Configure	all	first	fail	Subset
SWB-16	Configure	all	mid	success	Subset, return value
SWB-17	Miscellaneous	all	ran p	fail	Subset
SWB-18	Miscellaneous	all	not last	success	Subset, return value
SWB-19	Read	all	last	fail	Subset
SWB-20	Read	all	not mid	success	Subset, return value
SWB-21	Information	all	high	fail	Subset
SWB-22	Information	all	not first	success	Subset, return value
SWB-23	Control	all	ran u	fail	Subset
SWB-24	Control	all	even	success	Subset, return value
SWB-25	Write	three	PUU	fail	Subset
SWB-26	Write	three	UPU	success	Subset, return value
SWB-27	Write	three	UUP	fail	Subset
SWB-28	Write	three	UPP	success	Subset, return value
SWB-29	Write	three	PUP	fail	Subset
SWB-30	Write	three	PPU	success	Subset, return value
SWB-31	Read	three	PUU	fail	Subset
SWB-32	Read	three	UPU	success	Subset, return value
SWB-33	Read	three	UUP	fail	Subset
SWB-34	Read	three	UPP	success	Subset, return value
SWB-35	Read	three	PUP	fail	Subset
SWB-36	Read	three	PPU	success	Subset, return value
SWB-37	Write	all	odd	fail	Boot
SWB-38	Write	all	even	success	Boot, return value
SWB-39	all	all	high	fail	Uninstall
SWB-40	all	all	low	success	Uninstall, return value

9.3 Test Case Expected Results

Table 9-3 presents the expected results for each test assertion. The Expected Results column contains a message template for the related assertion. The Comments column presents additional information about the assertion. For assertions that have more than one possible expected result, the Comments column documents any conditions that select the particular expected result.

Table 9-3 Expected Results by Assertion

Assertion	Expected Result	Comments
AM-01	All cmds to drive XX blocked	Write commands
AM-02	All cmds to drive XX blocked	Configure commands
AM-03	All cmds to drive XX blocked	Miscellaneous commands
AM-04	No cmds to drive XX blocked	Read commands
AM-05	No cmds to drive XX blocked	Control commands
AM-06	No cmds to drive XX blocked	Information commands
AM-07	Tool Active message	Required message
AM-08	N drives identified	Required message
AM-09	Drive XX is protected	Protection status for protected drive
AM-09	Drive XX is unprotected	Protection status for unprotected drive
AM-10	N Commands return success	S option specified
AM-11	N Commands return fail	S option not specified
AO-01	All cmds to drive XX blocked	Write commands
AO-02	All cmds to drive XX blocked	Configure commands
AO-03	All cmds to drive XX blocked	Miscellaneous commands
AO-04	No cmds to drive XX blocked	Read commands
AO-05	No cmds to drive XX blocked	Control commands
AO-06	No cmds to drive XX blocked	Information commands
AO-07	No cmds to drive XX blocked	All commands to unprotected drives
AO-08	All cmds to drive XX blocked	Protected drive
AO-08	No cmds to drive XX blocked	Unprotected drive
AO-09	SWB not removed	S option specified
AO-09	SWB removed	S option not specified

9.4 Test Case Selection Guide

Not all test cases are appropriate for all tools. Each test case is assigned to a selection criterion based on optional tool features needed for the test case. If a given tool implements a given feature listed below then test cases assigned to the associated criterion are executed. The selection criteria are: base, return value, subset and uninstall.

The base criterion test cases apply to all tools and should always be run. For some tools, minor adjustments to parameter values may be required for test case execution.

If the SWB tool allows configuration of the return value parameter, the cases assigned to the return value criterion are executed.

If the SWB tool allows selection of a subset of covered drives for protection, the test cases assigned to the subset criterion are executed.

If the SWB tool can be uninstalled, the test cases assigned to the uninstall criterion are executed.

If the SWB tool can be active during the boot process, then the test cases assigned to the boot criterion are executed.

9.5 Run Protocol Selection

Most test cases follow the same test procedures. However, some test cases require a special procedure. The programs listed in Table 9-4 are required for testing.

Table 9-4 Software Required for Testing

Program	Description
SWB Tool	The software write block tool to be tested
Monitor	The interrupt 0x13 monitor program. The monitor program blocks all interrupt 0x13 command functions, counts the number of times each function is requested for each drive, and provides an interface for retrieving the count of the number of times each command function was requested for each drive.
Test Harness	The test harness issues (requests) all interrupt 0x13 command functions for a specified command category and then queries the monitor program to determine if the function was blocked or allowed.

Setup a selection of hard drives for use in testing as follows:

1. Select a hard drive.
2. Use the FS-TST **diskwipe** program to initialize the drive.
3. Create and format partitions on the drive (optional step).
4. Compute a reference SHA1 hash of the drive.

The general procedure for executing a test case is as follows:

1. Select a test case to execute.
2. Install the number of hard drives called for by the **installed** parameter.
3. Boot the test computer into DOS.
4. Follow the run protocol for the selected test case.
5. Save test case results to an archive location.

The **run protocol** specifies the actual procedures to follow for the test case. Some test cases require different setup procedures and methods to measure results. The following protocols are defined:

- Typical
  1. Execute the interrupt 0x13 monitor
  2. Execute the SWB tool under test with the specified **return** type.
  3. Execute the test harness for the specified **command category**.
- Uninstall
  1. Execute the interrupt 0x13 monitor
  2. Execute the SWB tool under test with the specified **return** type.
  3. Execute the test harness for the **write** command category.
  4. Uninstall the SWB tool.
  5. Execute the test harness for each **command category**
- Boot
  1. Install the SWB tool. (For DOS systems, include the tool in the **AUTOEXEC.BAT** file.)
  2. Boot the system.
  3. Execute the test harness for the **write** command category.
  4. Shutdown the system.

After all test cases are run, a SHA1 hash of each hard drive used in testing is computed. The hashes taken before testing are compared to the hashes taken after testing. If the hash values are the same then no changes have occurred to the drives during testing.

The **typical** protocol applies to cases 01-36, the **boot** protocol applies to test cases 37 and 38, and the **uninstall** protocol applies to cases 39 and 40.

Appendix A Interrupt 0x13 BIOS Access

A typical set of interrupt 0x13 BIOS drive access commands [Phoenix] implemented in PhoenixBIOS 4.0 Revision 6 can be categorized as in Table A-1. Other BIOS vendors or software installed on a PC may add or change functionality for the interrupt 0x13 interface.

Table A-1 Int 0x13 Commands Implemented in a Typical BIOS

Categorization of Interrupt 0x13 BIOS Commands		
Command	**Code**	**Category**
Reset	00h	Control
Get last status	01h	Information
Read sectors	02h	Read
Write sectors	03h	Write
Verify sectors	04h	Information
Format Cylinder	05h	Configuration
Read Drive Parameters	08h	Information
Initialize Drive Parameters	09h	Configuration
Read Long Sector	0Ah	Read
Write Long Sector	0Bh	Write
Seek Drive	0Ch	Control
Alternate drive reset	0Dh	Control
Test drive ready	10h	Information
Recalibrate drive	11h	Configuration
Controller diagnostic	14h	Configuration
Read drive type	15h	Information
Check extensions present	41h	Information
Extended read	42h	Read
Extended write	43h	Write
Verify sectors	44h	Information
Extended seek	47h	Control
Get drive parameters	48h	Information

The following table is a list of common interrupt 0x13 BIOS drive access commands and widely known extensions derived from http://www.ctyme.com/rbrown.htm (August 23, 2000).The listed commands have either been implemented in some BIOS or in some software module that adds new functionality to an existing BIOS. NIST does not make any claims for the accuracy of the contents but has included this to illustrate commands that should be used for test cases of interrupt 0x13 based software write block tools. Some commands that might modify hard drive contents are highlighted with a gray background.

Table A-2 Int 13 Commands Implemented in Some BIOS

Common Interrupt 0x13 BIOS Commands	
Command	**Description**
AH = 00h	DRIVE - RESET DRIVE SYSTEM
AH = 01h	DRIVE - GET STATUS OF LAST OPERATION
AH = 02h	DRIVE - READ SECTOR(S) INTO MEMORY
AH = 03h	DRIVE - WRITE DRIVE SECTOR(S)
AH = 04h	DRIVE - VERIFY DRIVE SECTOR(S)
AH = 05h	FLOPPY - FORMAT TRACK
AH = 05h	FIXED DRIVE - FORMAT TRACK

Common Interrupt 0x13 BIOS Commands	
Command	**Description**
AH = 05h	Future Domain SCSI BIOS - SEND SCSI MODE SELECT COMMAND
AX = 057Fh	2M - FORMAT TRACK
AH = 06h	FIXED DRIVE - FORMAT TRACK AND SET BAD SECTOR FLAGS (XT,PORT)
AH = 06h	Future Domain SCSI BIOS - FORMAT DRIVE WITH BAD SECTOR MAPPING
AH = 06h	Adaptec AHA-154xA/Bustek BT-542 BIOS - IDENTIFY SCSI DEVICES
AH = 06h	V10DRIVE.SYS - READ DELETED SECTORS
AH = 07h	FIXED DRIVE - FORMAT DRIVE STARTING AT GIVEN TRACK (XT,PORT)
AH = 07h	Future Domain SCSI BIOS - FORMAT DRIVE
AH = 07h	V10DRIVE.SYS - WRITE DELETED SECTORS
AH = 08h	DRIVE - GET DRIVE PARAMETERS (PC,XT286,CONV,PS,ESDI,SCSI)
AH = 08h	V10DRIVE.SYS - SET FORMAT
AX = 08000h	SecureDrive - INSTALLATION CHECK
AH = 09h	HARD DRIVE - INITIALIZE CONTROLLER WITH DRIVE PARAMETERS (AT,PS)
AH = 0Ah	HARD DRIVE - READ LONG SECTOR(S) (AT and later)
AH = 0Bh	HARD DRIVE - WRITE LONG SECTOR(S) (AT and later)
AH = 0Ch	HARD DRIVE - SEEK TO CYLINDER
AH = 0Dh	HARD DRIVE - RESET HARD DRIVES
AH = 0Eh	HARD DRIVE - READ SECTOR BUFFER (XT only)
AH = 0Fh	HARD DRIVE - WRITE SECTOR BUFFER (XT only)
AH = 10h	HARD DRIVE - CHECK IF DRIVE READY
AH = 11h	HARD DRIVE - RECALIBRATE DRIVE
AH = 12h	HARD DRIVE - CONTROLLER RAM DIAGNOSTIC (XT,PS)
AH = 12h	Future Domain SCSI CONTROLLER - STOP SCSI DRIVE
AH = 12h	SyQuest - START/STOP SCSI DRIVE
AH = 13h	HARD DRIVE - DRIVE DIAGNOSTIC (XT,PS)
AH = 13h	SyQuest - READ DRIVE PARAMATERS (for DOS 5+)
AH = 14h	HARD DRIVE - CONTROLLER INTERNAL DIAGNOSTIC
AH = 15h	DRIVE - GET DRIVE TYPE (XT 1/10/86 or later,XT286,AT,PS)
AH = 16h	FLOPPY DRIVE - DETECT DRIVE CHANGE (XT 1/10/86 or later,XT286,AT,PS)
AH = 17h	FLOPPY DRIVE - SET DRIVE TYPE FOR FORMAT (AT,PS)
AX = 1700h	Future Domain SCSI CONTROLLER - GET INQUIRY INFO FROM SCSI DEVICE
AH = 18h	DRIVE - SET MEDIA TYPE FOR FORMAT (AT model 3x9,XT2,XT286,PS)
AH = 18h	Future Domain SCSI BIOS - GET SCSI CONTROLLER INFORMATION
AH = 18h	PU_1700.COM - INSTALLATION CHECK
AH = 18h	XDF.COM - API
AH = 19h	FIXED DRIVE - PARK HEADS ON ESDI DRIVE (XT286,PS)
AH = 19h	Future Domain SCSI CONTROLLER - REINITIALIZE DRIVE
AH = 1Ah	ESDI FIXED DRIVE - FORMAT UNIT (PS)
AH = 1Ah	Future Domain SCSI CONTROLLER - GET SCSI PARTIAL MEDIUM CAPACITY
AH = 1Bh	ESDI FIXED DRIVE - GET MANUFACTURING HEADER
AH = 1Bh	Future Domain SCSI CONTROLLER - GET POINTER TO SCSI DRIVE INFO BLOCK
AH = 1Ch	Future Domain SCSI CONTROLLER - GET POINTER TO FREE CONTROLLER RAM
AH = 1Ch	ESDI FIXED DRIVE - ???
AX = 1C08h	ESDI FIXED DRIVE - GET COMMAND COMPLETION STATUS
AX = 1C09h	ESDI FIXED DRIVE - GET DEVICE STATUS
AX = 1C0Ah	ESDI FIXED DRIVE - GET DEVICE CONFIGURATION
AX = 1C0Bh	ESDI FIXED DRIVE - GET ADAPTER CONFIGURATION
AX = 1C0Ch	ESDI FIXED DRIVE - GET POS INFORMATION
AX = 1C0Dh	ESDI FIXED DRIVE - ???

Common Interrupt 0x13 BIOS Commands	
Command	**Description**
AX = 1C0Eh	ESDI FIXED DRIVE - TRANSLATE RBA TO ABA
AX = 1C0Fh	ESDI FIXED DRIVE - ???
AX = 1C12h	ESDI FIXED DRIVE - ???
AH = 1Dh	IBMCACHE.SYS - CACHE STATUS
AH = 1Fh	SyQuest - DOOR LATCH/DOOR BUTTON DETECT
AH = 20h	DRIVE - ??? (Western Digital "Super BIOS")
AH = 20h	Compaq, ATAPI Removable Media Device - GET CURRENT MEDIA FORMAT
AH = 20h	QUICKCACHE II v4.20 - DISMOUNT
AH = 21h	HARD DRIVE - PS/1 and newer PS/2 - READ MULTIPLE DRIVE SECTORS
AH = 21h	QUICKCACHE II v4.20 - FLUSH CACHE
AH = 22h	HARD DRIVE - PS/1 and newer PS/2 - WRITE MULTIPLE DRIVE SECTORS
AH = 22h	QUICKCACHE II v4.20 - ENABLE/DISABLE CACHE
AH = 23h	HARD DRIVE - PS/1 and newer PS/2 - SET CONTROLLER FEATURES REGISTER
AH = 23h	QUICKCACHE II v4.20 - GET ??? ADDRESS
AH = 24h	HARD DRIVE - PS/1 and newer PS/2 - SET MULTIPLE MODE
AH = 24h	QUICKCACHE II v4.20 - SET SECTORS
AH = 25h	HARD DRIVE - PS/1 and newer PS/2 - IDENTIFY DRIVE
AH = 25h	QUICKCACHE II v4.20 - SET FLUSH INTERVAL
AH = 26h	QUICKCACHE II v4.20 - UNINSTALL
AH = 27h	QUICKCACHE II v4.20 - INSTALLATION CHECK
AH = 28h	QUICKCACHE II v4.20 - SET AUTOMATIC DISMOUNT
AH = 29h	QUICKCACHE II v4.20 - NOP
AH = 2Ah	QUICKCACHE II v4.20 - SET BUFFER SIZE
AH = 2Bh	QUICKCACHE II v4.20 - DRIVE ACCESS SOUNDS
AH = 2Ch	QUICKCACHE II v4.20 - SET BUFFERED WRITES
AH = 2Dh	QUICKCACHE II v4.20 - SET BUFFERED READ
AH = 2Eh	QUICKCACHE II v4.20 - SET FLUSH COUNT
AH = 2Fh	QUICKCACHE II v4.20 - FORCE IMMEDIATE INCREMENTAL FLUSH
AH = 30h	QUICKCACHE II v4.20 - GET INFO
AH = 31h	QUICKCACHE II v4.20 - RESERVE MEMORY
AH = 32h	QUICKCACHE II v4.20 - ENABLE CACHING FOR SPECIFIC DRIVE
AH = 33h	QUICKCACHE II v4.20 - DISABLE CACHING FOR SPECIFIC DRIVE
AH = 34h	QUICKCACHE II v4.20 - SECTOR LOCKING
AH = 35h	QUICKCACHE II v4.20 - SET LOCK POOL SIZE
AH = 36h	QUICKCACHE II v4.20 - SET TRACE BUFFER SIZE
AH = 37h	QUICKCACHE II v4.20 - SET BUFFERED READS FOR SPECIFIC DRIVE
AH = 38h	QUICKCACHE II v4.20 - SET BUFFERED WRITES FOR SPECIFIC DRIVE
AH = 39h	QUICKCACHE II v4.20 - SET READ BUFFER SIZE FOR SPECIFIC DRIVE
AH = 3Ah	QUICKCACHE II v4.20 - SET WRITE BUFFER SIZE FOR SPECIFIC DRIVE
AH = 3Bh	QUICKCACHE II v4.20 - ENABLE/DISABLE ???
AH = 3Ch	QUICKCACHE II v4.20 - ENABLE/DISABLE ???
AH = 3Dh	QUICKCACHE II v4.20 - ENABLE/DISABLE CYLINDER FLUSH FOR DRIVE
AH = 3Eh	QUICKCACHE II v4.20 - SET SINGLE-SECTOR BONUS
AH = 3Fh	QUICKCACHE II v4.20 - SET BONUS THRESHOLD
AH = 40h	QUICKCACHE II v4.20 - SET "sticky_max"
AH = 41h	IBM/MS INT 13 Extensions - INSTALLATION CHECK
AH = 41h	QUICKCACHE II v4.20 - SAVE/RESTORE ???
AH = 42h	IBM/MS INT 13 Extensions - EXTENDED READ
AX = 4257h ("BW")	Beame&Whiteside BWLPD - INSTALLATION CHECK
AH = 43h	IBM/MS INT 13 Extensions - EXTENDED WRITE

Common Interrupt 0x13 BIOS Commands	
Command	**Description**
AH = 44h	IBM/MS INT 13 Extensions - VERIFY SECTORS
AH = 45h	IBM/MS INT 13 Extensions - LOCK/UNLOCK DRIVE
AH = 46h	IBM/MS INT 13 Extensions - EJECT MEDIA
AH = 47h	IBM/MS INT 13 Extensions - EXTENDED SEEK
AH = 48h	IBM/MS INT 13 Extensions - GET DRIVE PARAMETERS
AH = 49h	IBM/MS INT 13 Extensions - EXTENDED MEDIA CHANGE
AH = 4Ah	Bootable CD-ROM - INITIATE DRIVE EMULATION
AX = 4B00h	Bootable CD-ROM - TERMINATE DRIVE EMULATION
AX = 4B01h	Bootable CD-ROM - GET STATUS
AH = 4Ch	Bootable CD-ROM - INITIATE DRIVE EMULATION AND BOOT
AX = 4D00h	Bootable CD-ROM - RETURN BOOT CATALOG
AH = 4Eh	IBM/MS INT 13 Extensions v2.1 - SET HARDWARE CONFIGURATION
AX = 5001h	Enhanced Disk Drive Spec v3.0 - SEND PACKET COMMAND
AX = 5001h	VIRUS - "Andropinis" - INSTALLATION CHECK
AX = 5342h ("SB")	ScanBoot - INSTALLATION CHECK
AX = 5501h	Seagate ST01/ST02 - Inquiry
AX = 5502h	Seagate ST01/ST02 - RESERVED
AX = 5503h	Seagate ST01/ST01 - Set Device Type Qualifier (DTQ)
AX = 5504h	Seagate - ??? - RETURN IDENTIFICATION
AX = 5504h	Seagate ST01/ST02 - RETURN IDENTIFICATION
AX = 5505h	Seagate - ??? - PARK HEADS
AX = 5505h	Seagate ST01/ST02 - PARK HEADS
AX = 5506h	Seagate ST01/ST02 - SCSI Bus Parity
AX = 5507h to 550Dh	Seagate ST01/ST02 - RESERVED FUNCTIONS
AX = 5514h	Seagate - ???
AX = 5515h	Seagate - PARK HEADS???
AH = 59h	SyQuest - Generic SCSI pass through
AH = 70h	Priam EDVR.SYS DRIVE PARTITIONING SOFTWARE???
AH = 75h	???
AH = 76h	???
AX = 7B00h	NOW! v3.05 - GET INFORMATION
AX = 7B01h	NOW! v3.05 - ???
AX = 7B02h	NOW! v3.05 - SET INFORMATION
AX = 7B03h	NOW! v3.05 - ???
AX = 7B04h	NOW! v3.05 - ???
AX = 7B05h	NOW! v3.05 - GET DRIVE ACCESSES???
AX = 7B06h	NOW! v3.05 - GET ???
AX = 7B07h	NOW! v3.05 - GET ???
AX = 7B08h	NOW! v3.05 - ???
AH = 80h	FAST! v4.02+ - API
AX = 8001h	FAST! v4.02+ - GET CACHE INFORMATION
AX = 8006h	FAST! v4.02+ - INSTALLATION CHECK
AX = 8007h	FAST! v4.02+ - UNHOOK INTERRUPTS
AH = 81h	Super PC-Kwik v3.20+ - ???
AH = 82h	Super PC-Kwik v3.20+ - ???
AH = 83h	Super PC-Kwik v3.20+ - ???
AH = 84h	Super PC-Kwik v3.20+ - ???
AH = 85h	Super PC-Kwik v3.20+ - ???
AH = 86h	Super PC-Kwik v4.00+ - ???

Common Interrupt 0x13 BIOS Commands	
Command	**Description**
AH = 87h	Super PC-Kwik v4.00+ - ???
AH = 88h	Super PC-Kwik v4.00+ - ???
AH = 89h	Super PC-Kwik v5.10+ - ???
AH = 8Ah	Super PC-Kwik v5.10+ - ???
AX = 8EEDh	HyperDrive v4.01+ - ???
AX = 8EEEh	HyperDrive v4.01+ - ???
AX = 8EEFh	HyperDrive v4.01+ - ???
AH = 92h	Super PC-Kwik v5.10+ - ???
AH = 93h	Super PC-Kwik v5.10+ - ???
AH = 94h	Super PC-Kwik v5.10+ - ???
AH = 95h	Super PC-Kwik v5.10+ - ???
AH = 96h	Super PC-Kwik v5.10+ - ???
AH = 97h	Super PC-Kwik v5.10+ - ???
AH = 98h	Super PC-Kwik v5.10+ - ???
AH = 99h	Super PC-Kwik v5.10+ - ???
AH = 9Ah	Super PC-Kwik v5.10+ - ???
AH = 9Bh	Super PC-Kwik v5.10+ - ???
AH = 9Ch	Super PC-Kwik v5.10+ - ???
AH = 9Dh	Super PC-Kwik v5.10+ - ???
AH = A0h	Super PC-Kwik v3.20+ - GET RESIDENT CODE SEGMENT
AH = A1h	Super PC-Kwik v3.20+ - FLUSH CACHE
AH = A2h	Super PC-Kwik v3.20+ - ???
AH = A3h	Super PC-Kwik v5.10+ - DISABLE CACHE
AH = A4h	Super PC-Kwik v5.10+ - ENABLE CACHE
AH = A5h	Super PC-Kwik v5.10+ - PROGRAM TERMINATION NOTIFICATION
AH = A6h	Super PC-Kwik v5.10+ - PROGRAM LOAD NOTIFICATION
AH = A7h	Super PC-Kwik 5.1 - ???
AX = A759h	Novell DOS 7 - SDRes v27.03 - ???
AH = A8h	Super PC-Kwik 5.1 - ???
AH = A9h	Super PC-Kwik 5.1 - EXITCODE RETRIEVAL NOTIFICATION
AH = AAh	Super PC-Kwik v4+ - ???
AH = ABh	Super PC-Kwik v4+ - ???
AH = ACh	Super PC-Kwik v4+ - ???
AH = ADh	Priam HARD DRIVE CONTROLLER???
AH = ADh	Super PC-Kwik v4+ - ???
AH = AEh	Super PC-Kwik v5.10+ - ???
AH = B0h	Super PC-Kwik v3.20+ - ???
AX = E000h	XBIOS - COMMAND
AX = EC00h	VIRUS - "Tiso" - INSTALLATION CHECK
AH = EEh	SWBIOS - SET 1024-CYLINDER FLAG
AH = EFh	Ontrack Drive Rocket - SET CYLINDER OFFSET
AH = F2h	VIRUS - "Neuroquila" - INSTALLATION CHECK
AH = F9h	SWBIOS - INSTALLATION CHECK
AH = FAh	PC Tools v8+ VSAFE, VWATCH - API
AX = FD50h	VIRUS - "Predator" - INSTALLATION CHECK
AH = FEh	SWBIOS - GET EXTENDED CYLINDER COUNT
AH = FFh	EZ-Drive - INSTALLATION CHECK
AH = FFh	IBM SurePath BIOS - Officially "Private" Function
AX = FFFFh	UNIQUE UX Turbo Utility - SET TURBO MODE

Appendix B Terminology Example

Consider a computer with the following four drives attached:

Example Configuration			
Model	**Drive**	**Physical Interface**	**Access Interfaces**
Fujitsu MPF3153AT	0x80	EIDE	ATA, BIOS
WDC WD200BB-00AUA1	0x81	EIDE	ATA, BIOS
QUANTUM ATLAS10K2	0x82	SCSI	ASPI, BIOS
SEAGATE ST318404LC	0x83	SCSI	ASPI, BIOS

Each drive is attached to one physical interface, in this case, either EIDE or SCSI. Each EIDE drive can be accessed in either of two ways. The EIDE drives (attached as drives 0x80 and 0x81) can be accessed directly by the AT-Attachment (ATA) interface to the drive controller, or the drive can be accessed through the BIOS (by interrupt 0x13). The SCSI drives (attached as drives 0x82 and 0x83) can be accessed through either an ASPI driver or through the BIOS (by interrupt 0x13).

Consider a software write block tool that covers only BIOS drive access through the interrupt 0x13 interface. All of the drives can be accessed through the BIOS, but suppose that the tool is executed with drives 0x81 and 0x82 specified for protection. The **covered interface** is the interrupt 0x13 BIOS access. The tool does not cover the ASPI or ATA interfaces. All the drives are **covered drives** when accessed by the BIOS. None of the drives are **covered drives** when accessed by either the ATA or ASPI interfaces. Drives on drives 0x81 and 0x82 are **protected drives** when accessed by the BIOS. All the drives are **unprotected** when accessed by either ATA or ASPI.

Protection Specified for BIOS interrupt 0x13 access to 0x81 & 0x82		
Drive/access interface	**Covered**	**Protected**
0x80/BIOS	yes	no
0x80/ATA	no	no
0x81/BIOS	yes	yes
0x81/ATA	no	no
0x82/BIOS	yes	yes
0x82/ASPI	no	no
0x83/BIOS	yes	no
0x83/ASPI	no	no

Note that even though drives on drives 0x81 and 0x82 are protected from modification by access through the BIOS interface, these drives could still be modified if accessed through the ATA or ASPI interfaces.

Appendix C Traceability Matrices

This section presents traceability matrices to show the relationships between the requirements and assertions in Table C-1 and the relationships between assertions and test cases in Table C-.

Table C-1 Requirements to Assertions Traceability

	Assertions	Mandatory Requirements								Optional Requirements				
Requirements		SWB-RM-01	SWB-RM-02	SWB-RM-03	SWB-RM-04	SWB-RM-05	SWB-RM-06	SWB-RM-07	SWB-RM-08	SWB-RO-01	SWB-RO-02	SWB-RO-03	SWB-RO-04	SWB-RO-05
Mandatory Assertions	SWB-AM-01	●												
	SWB-AM-02	●												
	SWB-AM-03	●												
	SWB-AM-04		●											
	SWB-AM-05		●											
	SWB-AM-06		●											
	SWB-AM-07			●										
	SWB-AM-08				●									
	SWB-AM-09					●								
	SWB-AM-10						●							
	SWB-AM-11							●						
Optional Assertions	SWB-AO-01	●										●		
	SWB-AO-02	●										●		
	SWB-AO-03	●										●		
	SWB-AO-04		●									●		
	SWB-AO-05		●									●		
	SWB-AO-06		●									●		
	SWB-AO-07								●					●
	SWB-AO-08												●	
	SWB-AO-09										●			
	SWB-AO-10									●				

Table C-2 Assertions to Test Cases Traceability

Test Case	Mandatory Assertions											Optional Assertions									
	SWB-AM-01	SWB-AM-02	SWB-AM-03	SWB-AM-04	SWB-AM-05	SWB-AM-06	SWB-AM-07	SWB-AM-08	SWB-AM-09	SWB-AM-10	SWB-AM-11	SWB-AO-01	SWB-AO-02	SWB-AO-03	SWB-AO-04	SWB-AO-05	SWB-AO-06	SWB-AO-07	SWB-AO-08	SWB-AO-09	SWB-AO-10
SWB-01	●						●	●	●		●										●
SWB-02	●						●	●	●	●											●
SWB-03		●					●	●	●		●										●
SWB-04		●					●	●	●	●											●
SWB-05			●				●	●	●		●										●
SWB-06			●				●	●	●	●											●
SWB-07				●			●	●	●		●										●
SWB-08				●			●	●	●	●											●
SWB-09						●	●	●	●		●										●
SWB-10						●	●	●	●	●											●
SWB-11					●		●	●	●		●										●
SWB-12					●		●	●	●	●											●
SWB-13							●	●	●		●	●						●			●
SWB-14							●	●	●	●		●						●			●
SWB-15							●	●	●		●		●					●			●
SWB-16							●	●	●	●			●					●			●
SWB-17							●	●	●		●			●				●			●
SWB-18							●	●	●	●				●				●			●
SWB-19							●	●	●		●				●			●			●
SWB-20							●	●	●	●					●			●			●
SWB-21							●	●	●		●						●	●			●
SWB-22							●	●	●	●							●	●			●
SWB-23							●	●	●		●					●		●			●
SWB-24							●	●	●	●						●		●			●
SWB-25							●	●	●		●	●						●			●
SWB-26							●	●	●	●		●						●			●
SWB-27							●	●	●		●	●						●			●
SWB-28							●	●	●	●		●						●			●
SWB-29							●	●	●		●	●						●			●
SWB-30							●	●	●	●		●						●			●
SWB-31							●	●	●		●				●			●			●
SWB-32							●	●	●	●					●			●			●
SWB-33							●	●	●		●				●			●			●
SWB-34							●	●	●	●					●			●			●
SWB-35							●	●	●		●				●			●			●
SWB-36							●	●	●	●					●			●			●
SWB-37							●	●	●		●								●		●
SWB-38							●	●	●	●									●		●
SWB-39							●	●	●		●									●	●
SWB-40							●	●	●	●										●	●