Universal Serial Bus
Mass Storage Class

Compliance Test Specification

Revision 0.9a
June 30, 2005

Change History

Revision	Issue Date	Comments
0.6	August 15, 2002	Initial draft, pre-release
0.7	September 19, 2002	Initial draft after acceptance by MSC committee
0.7a	December 6, 2002	Minor cleanup
0.7b	May 30, 2003	Added command set assertion, TD infrastructure
0.7c	June 2, 2003	Minor cleanup
0.7d	September 25, 2003	Added CDB table, cleanup
0.7e	November 21, 2003	Restructured command set info, added common procedures, modified CBW checking, other cleanup.
0.7f	December 8, 2003	Edited in changes of RRs accepted at DWG meeting.
0.7g	February 27, 2004	Edited in RRs from February 4, 2004 DWG meeting
0.7h	July 9, 2004	Edited in RR014 from May 26, 2004 DWG meeting
0.7i	August 23, 2004	Edited in RRs from August 4, 2004 DWG meeting
0.8	December 15, 2004	RRs from December 1, 2004 DWG
0.8a	April 6, 2005	RRs from April 6, 2005 DWG; release to DWG
0.9	June 7, 2005	DWG voted to promote to 0.9, added disclaimer in footer, created .pdf according to USB-IF guidelines
0.9a	June 30, 2005	Edited in RR35 approved by electronic ballot June 30, 2005

Contributors

Doug Azzarito, Dell
Jerry Beckmann, Lexar Media
Fred Bhesania, Microsoft
Jim Blackson, Y-E Data
Mark Bohm, SMSC
Robert Chang, SanDisk
Kenny Chu, Hagiwara Sys-Com
Morten Christiansen, Ericsson Mobile
Graham Connolly, Fairchild
Matthew Dharm, Y-E Data
Francois Ennesser, Axalto
Dan Froelich, Intel Corporation
Martin Furuhjelm, Lexar Media
Alan Haffner, Lexar Media
Pat LaVarre, Lexar Media
Trenton Henry, SMSC
Henry Hutton, SanDisk
Masahiro Ito, Fujitsu
Hyo Bae Jeon, Hynix
Priyanka Kamat, SanDisk
Shin-han Kim, Samsung
Steve Kolokowsky, Cypress Semiconductor
Peter T. Larsen, Intel
Antonis Lazaridis, TDK
Frank Lin, SST
David Luke, Cypress Semiconductor
Eric Luttmann, Cypress Semiconductor
Bruce McFarland, Micron
Joseph Meza, SoftConnex
Michael Montgomery, AXALTO
Terry Moore, MCCI
Sivagar Natorajon, AMI
Nathan Obr, Microsoft
Hiromichi Oribe, Hagiwara SysCom
Mike Pearson, Samsung
Matt Pujol, LSI Logic
Dan Repich, SMSC
Eddy Reynolds, Hewlett-Packard
Chris Robinson, Microsoft
Bill Russell, Canon
Jim Sandman, Iomega
Chris Sawran, SMSC
Jack Schwartz, Sun Microsystems
Toyoko Shimizu, Y-E Data
Glen Slick, Microsoft
Ariel Sobelman, M-Systems
Curtis Stevens, Western Digital
Jackie Su, Genesys Logic
Shuba Swaminathan, Micron
Farshid Tabrizi, Lexar Media
Daniel Tuers, SanDisk
David Won, Samsung
Aran Ziv, M-Systems

Table of Contents

Table of Tables

1. Scope

This document specifies assertions and test procedures for use with devices that support one or more Mass Storage Class interfaces. The majority of the test procedures defined herein apply to interfaces that utilize the Bulk-Only Transport protocol. These test procedures provide only limited testing of interfaces that use the Control-Bulk-Interrupt protocol.

This testing is intended to be in addition to standard USB Compliance testing; assertions covered by the USBCV test document, for instance, are not covered here. This testing addresses basics of the descriptors which are specific to Mass Storage Class devices; Class-specific control requests; structure and content of CBW and CSW packets; handling of errors related to MSC protocol errors; basic handling of transport-level errors ("the Thirteen Cases"); and some command set basics.

2. Testing Philosophy and Considerations

2.1 Implementation

The test descriptions specified by this document will be integrated into the USBCV tool available from the USB-IF. It is intended that all devices which report a Mass Storage Class interface will be required to pass this test in order to receive logo certification. The tests described herein shall be run on all interfaces that report themselves as Mass Storage Class.

2.2 Thirteen Cases

A BOT CBW contains some redundant information. Information that describes the direction and size of a data transfer is explicitly stated in the *bmCBWFlags* and *dCBWDataTransferLength* fields. This information is also typically encoded in the command block (CB) found in the *CBWCB* field.

The background philosophy behind the BOT specification was that of a "protocol stack" type layered architecture, wherein the CBW was truly a wrapper for the command block. Therefore, the MSC layer of a protocol stack on the host could wrap the command block with additional information and send it to the device. The MSC layer on the device side could similarly strip off the additional information and pass the CB to the command block layer without needing to interpret it. In this way, the command block layer and the MSC layer could be independent of each other.

However, the BOT specification does not explicitly require this independence. In some device implementations, the protocol logic may be simultaneously aware of both the CB and the additional information found in the CBW.

Given the former interpretation, it is easy to invoke the Thirteen Cases in a test environment. The host can intentionally mismatch the data transfer direction and/or length information in the CB and the additional CBW information, using the independence of the information to provoke the desired device behavior.

This methodology will not work with a device wherein control does not treat these different pieces of information as independent. With such a device, it may be impossible to test all thirteen cases. It should still be possible to test all "normal" cases ($D \leq H$; Cases 1, 4, 5, 6, and 12) by using commands such as Inquiry. Other cases will be tested to ensure that the device response is reasonable, and that the device can recover from these cases.

This is accomplished in the test descriptions in this specification by allowing a device response of either 0x01 (command failed) or 0x02 (phase error) to tests that are attempting to provoke phase error cases. This does not imply a broader interpretation of the Bulk-Only Transport Specification; it is merely a consequence that it may be impossible, with some device implementations, to provoke the desired cases using the intentional mismatch described above.

2.3 Command set

Limited testing of required commands will also be tested. Optional commands will be tested to insure that a device either correctly supports the command or that it fails the command properly.

2.4 Bootability

While performing command set testing, the test application will also keep track of the device's response to the commands required by the Mass Storage Class Bootability Specification. The device under test will be reported as being compliant to that specification if the device supports all required Bootability commands, and the device serial number meets the requirements of the Bootability specification, Section 2 (which means meeting the requirements of the Bulk-Only Transport specification, sections 4.1.1 and 4.1.2).

2.5 High-speed versus full-speed

For devices which support high-speed operation, the test should be run twice – once in full-speed mode, and once in high-speed mode.

2.6 Multi-LUN devices

If a device reports multiple LUNs, the tests described herein shall be run on all LUNs.

3. Command set references

The following normative references define the commands to be tested.

American National Standard Institute

- X3T10/0995D SCSI-3 Primary Commands (SPC)
- X3T10/1236D SCSI Primary Commands-2 (SPC-2)
- X3T10/1416D SCSI Primary Commands-3 (SPC-3)
- X3T10/0996D SCSI-3 Block Commands (SBC)
- X3T10/1417D SCSI Block Commands-2 (SBC-2)
- X3T10/1048D SCSI-3 Multimedia Commands (MMC)
- X3T10/1545D MultiMedia Command Set-4 (MMC-4)
- X3T10/1363D MultiMedia Command Set-3 (MMC-3)

4. Assertions

4.1 Assertions

Note: Assertions are numbered beginning with "5.1.1" to fit into the assertion numbering scheme in use in the USBCV specification.

Descriptor Assertions

Num	Assertion
5.1.1	Device must have a serial number (if the device supports a BOT interface, bInterfaceProtocol = 0x50). **Specification Ref:** MSC BOT Specification, Revision 1.0, Section 4.1.1 **Test Description:** TD.1.2
5.1.2	Serial number must be a string, 12 characters or longer (if the device supports a BOT interface, bInterfaceProtocol = 0x50, and no non-MSC interfaces, bInterfaceClass ≠ 0x08), or exactly 12 characters long (if the device supports a CBI interface, bInterfaceProtocol = 0x00 or 0x01, and has a serial number. This is also a requirement for the interface to be considered as Bootable. **Specification Ref:** MSC BOT Specification, Revision 1.0, Section 4.1.1; MSC CBI Specification, Revision 1.1, Section 3.1; MSC Bootability Specification, Revision 1.1, Section 2 **Test Description:** TD.1.2
5.1.3	Serial number characters must be 0x0030-0x0039 or 0x0041-0x0046 (if the device supports a BOT interface, bInterfaceProtocol = 0x50, and no non-MSC interfaces, bInterfaceClass ≠ 0x08; or if the device supports a CBI interface, bInterfaceProtocol = 0x00 or 0x01). This is also a requirement for the interface to be considered as Bootable. **Specification Ref:** MSC BOT Specification, Revision 1.0, Section 4.1.2; MSC CBI Specification, Revision 1.1, Section 3.1 **Test Description:** TD.1.2

Descriptor Assertions

Num	Assertion
5.1.4	Last 12 characters of the serial number must be unique. This is also a requirement for the interface to be considered as Bootable. **Specification Ref:** MSC BOT Specification, Revision 1.0, Section 4.1.1; MSC CBI Specification, Revision 1.1, Section 3.1 **Test Description:** TD.1.2
5.1.5	Devices must support at least one Data Interface (InterfaceClass = 0x08). **Specification Ref:** MSC BOT Specification, Revision 1.0, Section 4.3, MSC CBI Specification, Revision 1.1, Section 3.3 **Test Description:** TD.1.1
5.1.6	Data Interface SubClass code must be valid (0x01 – 0x06). **Specification Ref:** MSC Overview Specification, Revision 1.1, Section 2 **Test Description:** TD.1.1
5.1.7	Data Interface Protocol code must be 0x50 (BOT interface), or 0x00 or 0x01 (CB/CBI interface). **Specification Ref:** MSC Overview Specification, Revision 1.1, Section 3 **Test Description:** TD.1.1
5.1.8	Data Interface must include at least two endpoints, one Bulk In and one Bulk Out. The first listed Bulk In and Bulk Out endpoints are the ones used for MSC transfers. **Specification Ref:** MSC BOT Specification, Revision 1.0, Section 4.3, 4.4, MSC CBI Specification, Revision 1.1, Section 3.3, 3.4 **Test Description:** TD.1.1
5.1.9	If the Data Interface Protocol code is 0x00 (CBI), then the Data Interface must also include at least one an Interrupt endpoint. The first listed Interrupt endpoint is the one used for MSC transfer status interrupts. **Specification Ref:** MSC CBI Specification, Revision 1.1, Section 3.4.3 **Test Description:** TD.1.1
5.1.10	The CB/CBI device must not be high-speed device. **Specification Ref:** MSC Overview Specification, Revision 1.2, Section 1 **Test Description:** TD.1.1
5.1.11	If the Data Interface Protocol code is 0x00 or 0x01, the Other Speed Configuration Descriptor request must fail. **Specification Ref:** MSC Overview Specification, Revision 1.2, Section 1 **Test Description:** TD.1.1
5.1.12	If the Data Interface Protocol code is 0x00 or 0x01, the Device Qualifier Descriptor request must fail. **Specification Ref:** MSC Overview Specification, Revision 1.2, Section 1 **Test Description:** TD.1.1

CBW Assertions

Num	Assertion
5.2.1	Devices receiving a CBW with an invalid signature should stall further traffic on the Bulk In pipe, and either stall further traffic or accept and discard further traffic on the Bulk Out pipe, until reset recovery. If the device does not react in this manner, it should at least respond correctly to a Reset Recovery. **Specification Ref** MSC BOT Specification, Revision 1.0, Section 3.1, 6.2.1, 6.6.1 **Test Description:** TD.1.4
5.2.2	Devices receiving a CBW with the wrong length should stall further traffic on the Bulk In pipe, and either stall further traffic or accept and discard further traffic on the Bulk Out pipe, until reset recovery. If the device does not react in this manner, it should at least respond correctly to a Reset Recovery. **Specification Ref:** MSC BOT Specification, Revision 1.0, Section 3.1, 6.2.1, 6.6.1 **Test Description:** TD.1.4
5.2.3	Devices must consider the CBW meaningful if no reserved bits are set, the LUN number indicates a LUN supported by the device, bCBWCBLength is in the range of 1 through 16, and the length and content of the CBWCB field are appropriate to the SubClass (report a warning if reaction to such behavior is not graceful). **Specification Ref** MSC BOT Specification, Revision 1.0, Section 6.2.2 **Test Description:** TD.1.5 – TD.1.17
5.2.4	Devices must ignore all data in the CBWCB field beyond that indicated by bCBWLength. **Specification Ref** MSC BOT Specification, Revision 1.0, Section 5.1 **Test Description:** TD.1.19
5.2.5	Devices must ignore the Direction bit if *dCBWDataTransferLength* = 0. **Specification Ref** MSC BOT Specification, Revision 1.0, Section 5.1 **Test Description:** TD.1.5

CSW Assertions

Num	Assertion
5.3.1	The CSW must be 13 bytes long. **Specification Ref** MSC BOT Specification, Revision 1.0, Section 5.2, 6.3.1 **Test Description:** TD.1.5 – TD.1.17
5.3.2	The CSW signature must be 0x53425355. **Specification Ref** MSC BOT Specification, Revision 1.0, Section 5.2, 6.3.1 **Test Description:** TD.1.5 – TD.1.17
5.3.3	The tag field of a CSW must match the tag of the associated CBW. **Specification Ref** MSC BOT Specification, Revision 1.0, Section 5.2, 6.2, 6.3.1 **Test Description:** TD.1.5 – TD.1.17
5.3.4	The CSW status value must be 0x00, 0x01, or 0x02. **Specification Ref** MSC BOT Specification, Revision 1.0, Section 5.2, 6.3.2 **Test Description:** TD.1.5 – TD.1.17

CSW Assertions

Num	Assertion
5.3.5	If the CSW status value is 0x00 or 0x01, the residue must be less than or equal to the transfer length. **Specification Ref** MSC BOT Specification, Revision 1.0, Section 5.2, 6.3.2 **Test Description:** TD.1.5 – TD.1.17

Error Recovery Assertions

Num	Assertion
5.4.1	The interface must respond to invalid CBWs by stalling all traffic until Reset Recovery. **Specification Ref** MSC BOT Specification, Revision 1.0, Section 5.3.4, 6.6.1 **Test Description:** TD.1.4
5.4.2	The interface must respond properly to Get_Status requests at all times. **Specification Ref** USB 2.0, Section 9.4.5 **Test Description:** TD.1.4

Class-Specific Request Assertions

Num	Assertion
5.5.1	If the device supports more than one LUN, it must report so via the Get Max LUN request. **Specification Ref** MSC BOT Specification, Revision 1.0, Section 3.2 **Test Description:** TD.1.3
5.5.2	If the device supports only one LUN, it may report so via the Get Max LUN request or may fail that request. **Specification Ref** MSC BOT Specification, Revision 1.0, Section 3.2 **Test Description:** TD.1.3
5.5.3	If a value is returned by Get Max LUN, the return value must be 0x00 through 0x0F, inclusive. **Specification Ref:** MSC BOT Specification, Revision 1.0, Section 3.2 **Test Description:** TD.1.3
5.5.4	The device must support the Bulk-Only Mass Storage Reset. **Specification Ref** MSC BOT Specification, Revision 1.0, Section 3.1 **Test Description:** TD.1.3
5.5.5	The device must fail with Request Error (STALL) any other Class-Specific requests targeted at the Bulk-Only Data Interface. **Specification Ref** MSC BOT Specification, Revision 1.0, Section 3.1, 3.2 **Test Description:** TD.1.3
5.5.6	The device must fail with Request Error (STALL) the Get Max LUN request with incorrect parameters. **Specification Ref** MSC BOT Specification, Revision 1.0, Section 3.2 **Test Description:** TD.1.3

Class-Specific Request Assertions

Num	Assertion
5.5.7	The device must fail with Request Error (STALL) the Bulk-Only Mass Storage Reset request with incorrect parameters. **Specification Ref** MSC BOT Specification, Revision 5.0, Section 3.1 **Test Description:** TD.1.3

Thirteen Cases Assertions

Num	Assertion
5.6.1	Case 1 – H*n* = D*n* (neither host nor device wants to transfer data) **Specification Ref** MSC BOT Specification, Revision 1.0, Section 6.7.1 **Test Description:** TD.1.5
5.6.2	Case 2 – H*n* < D*i* (host wants to transfer no data, device wants to transfer data in) **Specification Ref** MSC BOT Specification, Revision 1.0, Section 6.7.1 **Test Description:** TD.1.6
5.6.3	Case 3 – H*n* < D*o* (host wants to transfer no data, device wants to transfer data out) **Specification Ref** MSC BOT Specification, Revision 1.0, Section 6.7.1 **Test Description:** TD.1.7
5.6.4	Case 4 – H*i* > D*n* (host wants to transfer data in, device wants to transfer no data) **Specification Ref** MSC BOT Specification, Revision 1.0, Section 6.7.2 **Test Description:** TD.1.8
5.6.5	Case 5 – H*i* > D*i* (host wants to transfer data in, device wants to transfer less data in) **Specification Ref** MSC BOT Specification, Revision 1.0, Section 6.7.2 **Test Description:** TD.1.9
5.6.6	Case 6 – H*i* = D*i* (host and device wants to transfer the same amount of data in) **Specification Ref** MSC BOT Specification, Revision 1.0, Section 6.7.2 **Test Description:** TD.1.10
5.6.7	Case 7 – H*i* < D*i* (host wants to transfer data in, device wants to transfer more data in) **Specification Ref** MSC BOT Specification, Revision 1.0, Section 6.7.2 **Test Description:** TD.1.11
5.6.8	Case 8 – H*i* <> D*o* (host wants to transfer data in, device wants to transfer data out) **Specification Ref** MSC BOT Specification, Revision 1.0, Section 6.7.2 **Test Description:** TD.1.12
5.6.9	Case 9 – H*o* > D*n* (host wants to transfer data out, device wants to transfer no data) **Specification Ref** MSC BOT Specification, Revision 1.0, Section 6.7.3 **Test Description:** TD.1.13
5.6.10	Case 10 – H*o* <> D*i* (host wants to transfer data out, device wants to transfer data in) **Specification Ref** MSC BOT Specification, Revision 1.0, Section 6.7.3 **Test Description:** TD.1.14

Thirteen Cases Assertions

Num	Assertion
5.6.11	Case 11 – H*o* > D*o* (host wants to transfer data out, device wants to transfer less data out) **Specification Ref** MSC BOT Specification, Revision 1.0, Section 6.7.3 **Test Description:** TD.1.15
5.6.12	Case 12 – H*o* = D*o* (host and device want to transfer the same amount of data out) **Specification Ref** MSC BOT Specification, Revision 1.0, Section 6.7.3 **Test Description:** TD.1.16
5.6.13	Case 13 – H*o* < D*o* (host wants to transfer data out, device wants to transfer more data out) **Specification Ref** MSC BOT Specification, Revision 1.0, Section 6.7.3 **Test Description:** TD.1.17

Command Set Assertions - General

Num	Assertion
5.7.1	All devices must support the commands marked as required in 4.2 based on PDT. Devices that support all the commands marked as required for Bootable devices will be reported as Bootable (if they also meet Assertion 5.7.2). **Specification Ref** see Section 3 **Test Description:** TD.2.1
5.7.2	All devices must either support or properly fail the commands marked as optional in 4.2 based on PDT. Devices that support or properly fail all the commands marked as optional for Bootable devices will be reported as Bootable (if they also meet Assertion 5.7.1). **Specification Ref** see Section 3 **Test Description:** TD.2.2

Miscellaneous Assertions

Num	Assertion
5.8.1	The interface shall be capable of responding to commands without failure immediately following configuration. **Specification Ref** MSC BOT Specification, USB 2.0 Specification **Test Description:** TD.1.18

4.2 Command Set Information

The following table indicates which commands are required and optional for various Peripheral Device Types. In addition, the "Bootability" column indicates which commands are required for a device to be considered compliant with the Bootability specification, regardless of PDT.

Command	PDT→	00h/07h/0Eh	05h	Other	Bootability

	Op Code	Req	Opt	Req	Opt	Req	Opt	Req	Opt
Inquiry	12	X		X		X		X	
Request Sense	03	X		X		X		X	
Test Unit Ready	00	X		X		X		X	
Mode Sense(10)	5A		X		X		X	X[1]	
Read(10)	28	X			X	N/A		X	
Read(12)	A8		X	X		N/A		N/A	
Read Capacity	25	X		X		N/A		X	
Write(10)	2A	X			X	N/A		X[1]	
Write(12)	AA		X	X[1]		N/A		N/A	
Mode Select(6)	15		X		X	N/A		N/A	
Mode Select(10)	55		X		X	N/A		N/A	
Mode Sense(6)	1A		X		X	N/A		N/A	
Prevent/Allow Medium Removal	1E		X		X	N/A		N/A	
Read Format Capacity	23		X		X	N/A		N/A	
Start/Stop Unit	1B		X	X		N/A		N/A	
Synchronize Cache	35		X		X	N/A		N/A	
Read TOC/PMA/ATIP	43	N/A		X		N/A		X[2]	
Verify	2F	N/A		N/A		N/A			X

Note 1: These commands are required for read/write devices only.
Note 2: This command is required for devices of PDT = 5 only.

Table 1 Required and Optional Commands

5. Test Descriptions

Test Descriptors 1.1 and 1.2 apply to all Mass Storage Class interfaces. The other Test Descriptors apply only to Mass Storage Class interfaces which support Bulk-Only Transport (bInterfaceProtocol = 0x50).

TD.1.1 **Interface Descriptor Test**

This test verifies that the device under test responds to valid Get Configuration Descriptor commands and returns a descriptor in compliance with the specification.

Device States For Test

This test is run with the device in the Default, Address, and Configured states.

Overview of Test Steps

The test software performs the following steps.

1. Place the device in the desired starting state.
2. Issue a valid get configuration descriptor command with a requested length of 9 bytes.
3. Issue a valid get configuration descriptor request with a requested length of wTotalLength from the data returned in step 2.
4. Parse the data returned, verify that at least one interface descriptor exists with bInterfaceClass = 0x08. If not, skip to step 11.
5. For each interface descriptor with bInterfaceClass = 0x08, verify that bInterfaceSubClass is in the range 0x01 – 0x06.
6. For each interface descriptor with bInterfaceClass = 0x08, verify that bInterfaceProtocol is 0x50 (BOT) or 0x00 or 0x01 (CBI).
7. For each interface descriptor with bInterfaceClass = 0x08, verify that bNumEndpoints is two or greater (for bInterfaceProtocol = 0x01 or 0x50), or three or greater (for bInterfaceProtocol = 0x00).
8. For each interface descriptor with bInterfaceClass = 0x08, verify that there is at least one Bulk In endpoint descriptor and at least one Bulk Out endpoint descriptor.
9. If bInterfaceProtocol = 0x00, verify that there is at least one Interrupt In endpoint descriptor.
10. If bInterface Protocol = 0x00 or 0x01 (CBI), verify that the device only enumerates into full-speed mode.
11. If the device supports multiple configurations repeat the test for each configuration descriptor.
12. Request the Other Speed Configuration Descriptor. If bInterfaceProtocol = 0x00 or 0x01, verify that the request fails. If the device returns the Other Speed Configuration Descriptor, perform the checks of steps 2 through 9 on that descriptor.
13. If bInterfaceProtocol = 0x00 or 0x01, request the Device Qualifier Descriptor. Verify that the request fails

Results Interpretation

The test transcribes all results to a text based log file.
The test *fails* if:

Device enumeration fails following the method described In this specification.

Valid get descriptor commands fail for any reason.

Valid set address commands fail for any reason.

Valid set configuration commands fail for any reason.

Valid get configuration commands fail for any reason.

Any of the descriptor content checks fail.

No Mass Storage Class interface is found in any configuration.

The device is high-speed capable but bInterfaceProtocol = 0x00 or 0x01.

TD.1.2 **Serial Number Test**

This test verifies that the device under test provides a serial number in compliance with the specification.

Device States For Test

This test is run with the device in the Default, Address, and Configured states.

Overview of Test Steps

The test software performs the following steps.

1. Place the device in the desired starting state.
2. If the device supports a Bulk-Only Transport Data Interface (bInterfaceProtocol = 0x50), verify that iSerialNumber is not zero.
3. Issue a valid get string descriptor command with the index iSerialNumber, with a requested length of 1 byte.
4. Verify that the value of the byte returned is even and greater than or equal to 26 (exactly equal to 26 if the device supports only CBI interfaces, bInterfaceProtocol = 0x00 or 0x01). This is a requirement for the interface to be considered as Bootable.
5. Issue a valid get string descriptor command with the index iSerialNumber, with a requested length of 2 bytes.
6. Verify that the value of the first byte returned is the same as in step 3, and that the value of the second byte returned is 0x03.
7. Issue a valid get string descriptor command with the index iSerialNumber, with a requested length equal to the value previously returned in the first byte.
8. Verify that the number of bytes returned equals the length requested.
9. Verify that the value of the first byte returned is the same as in step 3, and that the value of the second byte returned is 0x03.
10. Verify that the value of each pair of bytes following the first two, the value is in the range 0x0030-0x0039 or 0x0041-0x0046. This is a requirement for the interface to be considered as Bootable.
11. Check that the values of the last 12 pairs of the serial number are not equal to 0x0030 (note: warning only).
12. Issue a valid get string descriptor command with the index iSerialNumber, with a requested length of 0x1000.
13. Verify that the descriptor returned is identical to that returned in step 6.
14. Repeat test with device in the default, address, and configured states.
15. If the test supports multiple configurations repeat test for the configured state for each possible configuration.
16. Re-enumerate the device at a different bus address. Read the serial number and verify that it is the same as what was returned in step 6.

Results Interpretation

The test transcribes all results to a text based log file.
The test *fails* if:

Device enumeration fails following the method described In this specification.

Valid get descriptor requests fail for any reason.

Valid set configuration commands fail for any reason.

Any of the descriptor content checks fail.

(Warning if the last 12 pairs of the serial number are all equal to 0x0030.)

TD.1.3 **Class-Specific Request Test**

This test verifies that the device under correctly implements class-specific requests in compliance with the specification.

Device States For Test

This test is run with the device in the Configured state.

Overview of Test Steps

The test software performs the following steps.

1. Place the device in the desired starting state.
2. Issue a Get Max LUN request. Verify that either:
    a) the request results in Request Error (STALL), or
    b) the device returns one byte with a value of 0x00 through 0x0F, inclusive. This value will be used as the number of supported LUNs.
3. Issue a Get Max LUN request with wIndex set to an unsupported interface number. Verify that the request results in a Request Error (STALL) condition.
4. Issue a Get Max LUN request with wValue set to a value other than zero. Verify that the request results in a Request Error (STALL) condition (report a warning only).
5. Issue a Get Max LUN request with wLength = 0. Verify that the request results in a Request Error (STALL) condition or completes normally but returns no data.
6. Issue a Get Max LUN request with wLength > 1. Verify that the request results in a Request Error (STALL) condition or returns one byte of data. If data is returned, verify that the data is the same as in step 2.
7. Issue a correct Get Max LUN request. Verify that the response is the same as in step 2.
8. Issue a Bulk-Only Mass Storage Reset request. Verify that the request completes normally.
9. Issue a Bulk-Only Mass Storage Reset request with wIndex set to an unsupported interface number. Verify that the request results in a Request Error (STALL) condition.
10. Issue a Bulk-Only Mass Storage Reset request with the following errors. Verify that the request results in a Request Error (STALL) condition (report a warning only).
    a) wValue != 0
    b) wLength != 0
11. Issue a correct Bulk-Only Mass Storage Reset request. Verify that the request completes normally.

Results Interpretation

The test transcribes all results to a text based log file.
The test *fails* if:

Get Max LUN behavior is not as described in step 2.

Requests do not result in a Request Error response at steps 3 and 9.

Different values are returned in steps 6 and 7 than was returned in step 2.

The request does not complete normally in steps 7, 8, and 11.

Requests do not result in a Request Error response at step 9.

Request responses are not as described above in steps 5 and 6.

(Warning if requests do not result in Request Error response in steps 4 and 10.)

TD.1.4 **Error Recovery Test**

This test verifies that the device under test correctly implements the “hard stall” condition and recovery from that condition in compliance with the specification.

Device States For Test

This test is run with the device in the Configured state.

Overview of Test Steps

The test software performs the following steps.

1. Place the device in the desired starting state.
2. Issue a Case 9 CBW (see Table 2) but with invalid signature of 0xDEADBEEF.
3. Issue several In requests to the Bulk-Only Data Interface Bulk In endpoint. Verify a STALL handshake is returned. If STALL handshake is not returned, skip to step 11.
4. Issue a Get_Status(endpoint) request targeting the Bulk-Only Data Interface bulk In endpoint. Verify that it completes normally, reports endpoint halt status.
5. Issue a Clear_Feature(endpoint halt) request to the stalled Bulk-Only Data Interface bulk In endpoint.
6. Issue several In requests to the Bulk-Only Data Interface bulk In endpoint. Verify that all requests are stalled by the device.
7. Issue several Out requests to the Bulk-Only Data Interface bulk Out endpoint. Verify that requests are either all stalled or all accepted by the device. If a STALL handshake is not returned, skip to step 11.
8. Issue a Get_Status(endpoint) request targeting the Bulk-Only Data Interface bulk Out endpoint. Verify that it completes normally, reports endpoint proper status (halted or not).
9. If the device has stalled the Out endpoint, issue a Clear_Feature(endpoint halt) request to the stalled Bulk-Only Data Interface bulk Out endpoint.
10. Issue several Out requests to the Bulk-Only Data Interface bulk Out endpoint. Verify that requests are either all stalled or all accepted by the device.
11. Perform a Reset Recovery by issuing a Bulk-Only Mass Storage Reset request, followed by Clear_Feature(endpoint halt) requests targeting both bulk endpoints.
12. Issue a Get_Status(endpoint) request to both Bulk-Only Data Interface endpoints. Verify that the requests complete normally, report endpoint not halted status.
13. Perform a Test Unit Ready to verify that the device has recovered from the “hard stall” state.
14. Repeat steps 2-13, replacing the action of step 2 with sending a Case 9 CBW with a valid signature (see Table 2) truncated to 30 bytes, and again replacing the action of step 2 with sending a Case 9 CBW with a valid signature (see Table 2) padded to 32 bytes with 0x00.
15. Issue a normal Case 9 request (see Table 2), verify that it completes correctly.

Results Interpretation

The test transcribes all results to a text based log file.

The test *fails* if:

Any control request (steps 4, 8, 9, 11, or 12) fails.

Any Get_Status returns the wrong status.

The MSC transfers at steps 13 or 15 do not complete normally.

The first In request was stalled at step 3 and:

Any following In requests were not stalled at step 3.

The device reports incorrect endpoint status at step 5.

Any In requests stalled at step 6.

At least one but not all Out requests is stalled at step 7.

Any Out requests stalled at step 10 if any were stalled at step 7.

TD.1.5 **Case 1 Test**

This test verifies that the device under test implements "Case 1" (H*n* = D*n*, neither host nor device want to transfer data) in compliance with the specification.

Device States For Test

This test is run with the device in the Configured state.

Overview of Test Steps

The test software performs the following steps.

1. Execute the Preliminary Procedure (see Section 6.1).
2. Issue correct CBW for no-data transfer (see Table 2) with Direction = 0.
3. Issue In request. Verify that the CSW is 13 bytes long; that it has the correct signature; that the tag matches that of the CSW; that the Residue is 0; and that the Status is 0 or 1.
4. Execute the Recovery Procedure (see Section 6.2).
5. Repeat steps 2 through 4 with Direction = 1.

Results Interpretation

The test transcribes all results to a text based log file.
The test *fails* if:

The CSW is not 13 bytes long.

The CSW signature is not 0x53425355.

The CSW tag doesn't match the CBW tag.

The Residue is not 0.

The Status is not 0 or 1.

TD.1.6 **Case 2 Test**

This test verifies that the device under test implements "Case 2" (H*n* < D*i*, host wants to transfer no data, device wants to transfer data in) in compliance with the specification.

Device States For Test

This test is run with the device in the Configured state.

Overview of Test Steps

The test software performs the following steps.

1. Execute the Preliminary Procedure (see Section 6.1).
2. Issue a CBW to induce Case 2 behavior (see Table 2).
3. Issue In request. If the request is Stalled, check endpoint status, clear the Stall condition, check endpoint status again, and issue another In request. Verify that the CSW is 13 bytes long; that it has the correct signature; that the tag matches that of the CSW; and that the Status is 2. Also check to see if the Residue is equal to the CBW Transfer Length (warning only).
4. Execute the Recovery Procedure (see Section 6.2).

Results Interpretation

The test transcribes all results to a text based log file.
The test *fails* if:

The CSW is not 13 bytes long.

The CSW signature is not 0x53425355.

The CSW tag doesn't match the CBW tag.

The Status is not 2 or 1.

Endpoint status is incorrect before or after clearing an endpoint stall.

(Warning if the Residue is not equal to the CBW Transfer Length.)

TD.1.7 **Case 3 Test**

This test verifies that the device under test implements "Case 3" (H*n* < D*o*, host wants to transfer no data, device wants to transfer data out) in compliance with the specification.

Device States For Test

This test is run with the device in the Configured state.

Overview of Test Steps

The test software performs the following steps.

1. Execute the Preliminary Procedure (see Section 6.1).
2. Issue a CBW to induce Case 3 behavior (see Table 2).
3. Issue In request. If the request is Stalled, check endpoint status, clear the Stall condition, check endpoint status again, and issue another In request. Verify that the CSW is 13 bytes long; that it has the correct signature; that the tag matches that of the CSW; and that the Status is 2. Also check to see if the Residue is equal to the CBW Transfer Length (warning only).
4. Execute the Recovery Procedure (see Section 6.2).

Results Interpretation

The test transcribes all results to a text based log file.
The test *fails* if:

The CSW is not 13 bytes long.

The CSW signature is not 0x53425355.

The CSW tag doesn't match the CBW tag.

The Status is not 2 or 1.

Endpoint status is incorrect before or after clearing an endpoint stall.

(Warning if the Residue is not equal to the CBW Transfer Length.)

TD.1.8 **Case 4 Test**

This test verifies that the device under test implements "Case 4" (H*i* > D*n*, host wants to transfer data in, device wants to transfer no data) in compliance with the specification.

Device States For Test

This test is run with the device in the Configured state.

Overview of Test Steps

The test software performs the following steps.

1. Execute the Preliminary Procedure (see Section 6.1).
2. Issue a CBW to induce Case 4 behavior (see Table 2).
3. Issue In request sufficient to transfer the CBW Transfer Length. If the request completes, issue another In request (for the CSW); if the request is Stalled, check endpoint status, clear the Stall condition, check endpoint status again, and issue another In request (for the CSW). Verify that the CSW is 13 bytes long; that it has the correct signature; that the tag matches that of the CSW; and that the Status is 0 or 1. Also check to see if the Residue is equal to the CBW Transfer Length.
4. Execute the Recovery Procedure (see Section 6.2).

Results Interpretation

The test transcribes all results to a text based log file.
The test *fails* if:

The CSW is not 13 bytes long.

The CSW signature is not 0x53425355.

The CSW tag doesn't match the CBW tag.

The Status is not 0 or 1.

The Residue is not Transfer Length.

Endpoint status is incorrect before or after clearing an endpoint stall.

TD.1.9 **Case 5 Test**

This test verifies that the device under test implements "Case 5" (H*i* > D*i*, host wants to transfer data in, device wants to transfer less data in) in compliance with the specification.

Device States For Test

This test is run with the device in the Configured state.

Overview of Test Steps

The test software performs the following steps.

1. Execute the Preliminary Procedure (see Section 6.1).
2. Issue a CBW to induce Case 5 (see Table 2).
3. Issue In request sufficient to transfer the CBW Transfer Length. If the request completes, issue another In request (for the CSW); if the request is Stalled, check endpoint status, clear the Stall condition, check endpoint status again, and issue another In request (for the CSW). Verify that the CSW is 13 bytes long; that it has the correct signature; that the tag matches that of the CSW; and that the Status is 0 or 1. Also check to see if the Residue is equal to the CBW Transfer Length minus the amount of data requested in the CBWCB (or in the length field of the Inquiry data).
4. Execute the Recovery Procedure (see Section 6.2).

Results Interpretation

The test transcribes all results to a text based log file.
The test *fails* if:

The CSW is not 13 bytes long.

The CSW signature is not 0x53425355.

The CSW tag doesn't match the CBW tag.

The Status is not 0 or 1.

The Residue is not Transfer Length minus the amount of real data transferred.

Endpoint status is incorrect before or after clearing an endpoint stall.

TD.1.10 **Case 6 Test**

This test verifies that the device under test implements "Case 6" (H*i* = D*i*, host and device wants to transfer the same amount of data in) in compliance with the specification.

Device States For Test

This test is run with the device in the Configured state.

Overview of Test Steps

The test software performs the following steps.

1. Execute the Preliminary Procedure (see Section 6.1).
2. Issue a correct CBW to induce Case 6 behavior (see Table 2).
3. Issue In request sufficient to transfer the CBW Transfer Length. Issue another In request (for the CSW). Verify that the CSW is 13 bytes long; that it has the correct signature; that the tag matches that of the CSW; that the Residue is 0; and that the Status is 0 or 1.
4. Execute the Recovery Procedure (see Section 6.2).

Results Interpretation

The test transcribes all results to a text based log file.
The test *fails* if:

Either In transfer is stalled.

The CSW is not 13 bytes long.

The CSW signature is not 0x53425355.

The CSW tag doesn't match the CBW tag.

The Status is not 0 or 1.

The Residue is not zero.

TD.1.11 **Case 7 Test**

This test verifies that the device under test implements "Case 7" (H*i* < D*i*, host wants to transfer data in, device wants to transfer more data in) in compliance with the specification.

Device States For Test

This test is run with the device in the Configured state.

Overview of Test Steps

The test software performs the following steps.

1. Execute the Preliminary Procedure (see Section 6.1).
2. Issue a CBW to induce Case 7 behavior (see Table 2).
3. Issue In request sufficient to transfer the CBW Transfer Length. If the request completes, issue another In request (for the CSW); if the request is Stalled, check endpoint status, clear the Stall condition, check endpoint status again, and issue another In request (for the CSW). Verify that the CSW is 13 bytes long; that it has the correct signature; that the tag matches that of the CSW; and that the Status 2.
4. Execute the Recovery Procedure (see Section 6.2).

Results Interpretation

The test transcribes all results to a text based log file.
The test *fails* if:

Both In transfers are stalled (one stall is okay, two are bad).

The CSW is not 13 bytes long.

The CSW signature is not 0x53425355.

The CSW tag doesn't match the CBW tag.

The Status is not 2 or 1.

Endpoint status is incorrect before or after clearing an endpoint stall.

TD.1.12 **Case 8 Test**

This test verifies that the device under test implements "Case 8" (H*i* <> D*o*, host wants to transfer data in, device wants to transfer data out) in compliance with the specification.

Device States For Test

This test is run with the device in the Configured state.

Overview of Test Steps

The test software performs the following steps.

1. Execute the Preliminary Procedure (see Section 6.1).
2. Issue a CBW to induce Case 8 behavior ((see Table 2).
3. Issue In request sufficient to transfer the CBW Transfer Length. If the In request times out, perform a Reset Recovery procedure and end the test. If the In request completes, issue another In request (for the CSW); if the request is Stalled, check endpoint status, clear the Stall condition, check endpoint status again, and issue another In request (for the CSW). Verify that the CSW is 13 bytes long; that it has the correct signature; that the tag matches that of the CSW; and that the Status is 2. Also check that the Residue equals the CBW Transfer Length (warning only).
4. Execute the Recovery Procedure (see Section 6.2).

Results Interpretation

The test transcribes all results to a text based log file.
The test *fails* if:

The In request times out and the Reset Recovery procedure fails.

Both In transfers are stalled (one stall is okay, two are bad).

The CSW is not 13 bytes long.

The CSW signature is not 0x53425355.

The CSW tag doesn't match the CBW tag.

The Status is not 2 or 1.

Endpoint status is incorrect before or after clearing an endpoint stall.

(Warning if the CSW Residue is not equal to the CBW Transfer Length.)

TD.1.13 **Case 9 Test**

This test verifies that the device under test implements "Case 9" (H*o* > D*n*, host wants to transfer data out, device wants to transfer no data) in compliance with the specification.

Device States For Test

This test is run with the device in the Configured state.

Overview of Test Steps

The test software performs the following steps.

1. Execute the Preliminary Procedure (see Section 6.1).
2. Issue a CBW to induce Case 9 behavior (see Table 2).
3. Issue Out request for the CBW Transfer Length. If the request completes normally or is stalled, issue an In request (for the CSW). Verify that the CSW is 13 bytes long; that it has the correct signature; that the tag matches that of the CSW; and that the Status is 0 or 1. Also check that the Residue equals the CBW Transfer Length.
4. Execute the Recovery Procedure (see Section 6.2).

Results Interpretation

The test transcribes all results to a text based log file.
The test *fails* if:

The In transfer is stalled.

The CSW is not 13 bytes long.

The CSW signature is not 0x53425355.

The CSW tag doesn't match the CBW tag.

The Status is not 0 or 1.

The CSW Residue is not equal to the CBW Transfer Length.

TD.1.14 **Case 10 Test**

This test verifies that the device under test implements "Case 10" (H*o* <> D*i*, host wants to transfer data out, device wants to transfer data in) in compliance with the specification.

Device States For Test

This test is run with the device in the Configured state.

Overview of Test Steps

The test software performs the following steps.

1. Execute the Preliminary Procedure (see Section 6.1).
2. Issue a CBW to induce Case 10 behavior (see Table 2).
3. Issue Out request for the CBW Transfer Length. If the Out request times out, perform a Reset Recovery procedure and end the test. If the Out request completes normally or is stalled, issue an In request (for the CSW). If the In request is stalled, check endpoint status, clear the endpoint stall, check endpoint status again, and issue another In request (for the CSW). Verify that the CSW is 13 bytes long; that it has the correct signature; that the tag matches that of the CSW; and that the Status is 2. Also check that the Residue equals the CBW Transfer Length (warning only).
4. Execute the Recovery Procedure (see Section 6.2).

Results Interpretation

The test transcribes all results to a text based log file.
The test *fails* if:

The Out request times out and the Reset Recovery procedure fails.

The In transfer is stalled twice.

The CSW is not 13 bytes long.

The CSW signature is not 0x53425355.

The CSW tag doesn't match the CBW tag.

The Status is not 2 or 1.

Endpoint status is incorrect before or after clearing an endpoint stall.

(Warning if the CSW Residue is not equal to the CBW Transfer Length.)

TD.1.15 **Case 11 Test**

This test verifies that the device under test implements "Case 11" (H*o* > D*o*, host wants to transfer data out, device wants to transfer less data out) in compliance with the specification.

Device States For Test

This test is run with the device in the Configured state.

Overview of Test Steps

The test software performs the following steps.

1. Execute the Preliminary Procedure (see Section 6.1).
2. Issue a CBW to induce Case 11 behavior (see Table 2).
3. Issue Out request for the CBW Transfer Length. If the request completes normally or is stalled, issue an In request (for the CSW). Verify that the CSW is 13 bytes long; that it has the correct signature; that the tag matches that of the CSW; and that the Status is 0 or 1. Also check that the Residue equals the CBW Transfer Length minus the value in the CBWCB request.
4. Execute the Recovery Procedure (see Section 6.2).

Results Interpretation

The test transcribes all results to a text based log file.
The test *fails* if:

The In transfer is stalled.

The CSW is not 13 bytes long.

The CSW signature is not 0x53425355.

The CSW tag doesn't match the CBW tag.

The Status is not 0 or 1.

The Residue is not Transfer Length minus the value in the CBWCB request.

TD.1.16 **Case 12 Test**

This test verifies that the device under test implements "Case 12" (H*o* = D*o*, host and device want to transfer the same amount of data out) in compliance with the specification.

Device States For Test

This test is run with the device in the Configured state.

Overview of Test Steps

The test software performs the following steps.

1. Execute the Preliminary Procedure (see Section 6.1).
2. Issue a correct CBW to induce Case 12 behavior (see Table 2).
3. Issue Out request for the CBW Transfer Length. If the request completes normally, issue an In request (for the CSW). Verify that the CSW is 13 bytes long; that it has the correct signature; that the tag matches that of the CSW; and that the Status is 0 or 1. Also check that the Residue zero.
4. Execute the Recovery Procedure (see Section 6.2).

Results Interpretation

The test transcribes all results to a text based log file.
The test *fails* if:

The Out or In transfers are stalled.

The CSW is not 13 bytes long.

The CSW signature is not 0x53425355.

The CSW tag doesn't match the CBW tag.

The Status is not 0 or 1.

The Residue is not zero.

TD.1.17 **Case 13 Test**

This test verifies that the device under test implements "Case 13" (H*o* < D*o*, host wants to transfer data out, device wants to transfer more data out) in compliance with the specification.

Device States For Test

This test is run with the device in the Configured state.

Overview of Test Steps

The test software performs the following steps.

1. Execute the Preliminary Procedure (see Section 6.1).
2. Issue a CBW to induce Case 13 behavior (see Table 2).
3. Issue Out request sufficient to transfer the CBW Transfer Length. If the request completes normally or is stalled, issue another In request (for the CSW); if the request is Stalled, check the endpoint status, clear the Stall condition, check the endpoint status again, and issue another In request (for the CSW). Verify that the CSW is 13 bytes long; that it has the correct signature; that the tag matches that of the CSW; and that the Status 2.
4. Execute the Recovery Procedure (see Section 6.2).

Results Interpretation

The test transcribes all results to a text based log file.
The test *fails* if:

The CSW is not 13 bytes long.

The CSW signature is not 0x53425355.

The CSW tag doesn't match the CBW tag.

The Status is not 2 or 1.

Endpoint status is incorrect before or after clearing an endpoint stall.

TD.1.18 **Power-Up Test**

This test verifies that the device under test behaves properly shortly after power-up and enumeration.

Device States For Test

This test is run beginning with the device disconnected and not powered.

Overview of Test Steps

The test software performs the following steps.

1. Prompt the operator to plug the device in and then (if self-powered) to turn power on to the device.
2. Execute the Preliminary Procedure (see Section 6.2).
3.

Results Interpretation

The test transcribes all results to a text based log file.
The test *fails* if:

The device does not return 36 bytes.

The CSW is not 13 bytes long.

The CSW signature is not 0x53425355.

The CSW tag doesn't match the CBW tag.

The Status is not 0 or 1.

TD.1.19 **bCBLength Test**

This test verifies that the device under test ignores all bytes in the CBWCB field beyond what is indicated by bCBWLength.

Device States For Test

This test is run with the device in the Configured state.

Overview of Test Steps

The test software performs the following steps.

1. Place the device in the desired starting state.
2. Send a CBW for a normal transfer (one that should succeed), with bCBWLength set to the exact length of the CB, and the remainder of the CBWCB field padded with 0xFF.
3. Verify that the command completes correctly.
4. Repeat, padding the CBWCB field with 0x55 and 0xAA.

Results Interpretation

The test transcribes all results to a text based log file.
The test *fails* if:

The command does not transfer the data expected.

The CSW is not 13 bytes long.

The CSW signature is not 0x53425355.

The CSW tag doesn't match the CBW tag.

The Status is not 0 or 1.

TD.2.1 Required Commands Test

This test verifies that the device correctly supports required commands. While executing this test, also keep track of whether required Bootability commands are correctly supported. At the end of the test, report whether or not the device under test supports all required Bootability commands.

Device States For Test

This test is run with the device in the Configured state.

Overview of Test Steps

The test software performs the following steps.

1. Execute the Preliminary Procedure (see Section 6.1).
2. Extract the Peripheral Device Type of the device under test from the Inquiry data.
3. Issue a CBW for a command required for this Peripheral Device Type (see Table 1).
4. Verify that the command completed successfully.
5. Repeat steps 3 and 4 for all required commands.

Results Interpretation

The test transcribes all results to a text based log file.
The test *fails* if:

Any command times out.

Any command completes with a bCBWStatus other than 00h.

TD.2.2 Optional Commands Test

This test verifies that the device correctly supports or properly fails optional commands. While executing this test, also keep track of whether optional Bootability commands are correctly supported or fail properly. At the end of the test, report whether or not the device under test properly handles all optional Bootability commands.

Device States For Test

This test is run with the device in the Configured state.

Overview of Test Steps

The test software performs the following steps.

1. Execute the Preliminary Procedure (see Section 6.1).
2. Extract the Peripheral Device Type of the device under test from the Inquiry data.
3. Issue a CBW for a command that is optional for this Peripheral Device Type (see Table 1).
4. Verify that the command completed successfully or fails properly.
5. Repeat steps 3 and 4 for all optional commands.

Results Interpretation

The test transcribes all results to a text based log file.
The test *fails* if:

Any command times out.

Any command completes with a bCBWStatus other than 00h or 01h.

6. Test Details

This section describes the procedures and CDBs used in the test descriptions above.

6.1 Preliminary Procedure

1. Place the device in the desired starting state (configured).
2. Issue an Inquiry command. If the Inquiry fails, abort and fail the test.
3. Issue a Test Unit Ready command. Repeat until the device indicates a ready state.
4. Issue a Read Capacity command. Use the results to determine device block size.

6.2 Recovery Procedure

1. If the device returned Command Passed status, this procedure is done.
2. If the device returned Command Failed status, issue a Request Sense command.
3. If the device returned Phase Error status, perform a Reset Recovery procedure.
4. If the MSC Reset fails, re-enumerate the device.

6.3 CDBs

This section describes the CBWs used in the test descriptions TD.1.5 to TD.1.19 above.

TD	Case	Command Name	Op code	No. of blocks	CBWDataTransferLength	Direction
Preliminary Procedure (6.1)	--	Inquiry Test Unit Ready Read Capacity	0x12 0x00 0x25	0x24 bytes -- 0x0A bytes	0x24 0x0000 0x0A	1 N/A 1
TD.1.5	1	Test Unit Ready	0x00	--	0x0000	N/A
TD.1.6	2	Read(10)	0x28	0x01	0x0000	N/A
TD.1.7	3	Write(10)	0x2A	0x01	0x0000	N/A
TD.1.8	4	Read(10)	0x28	0x00	block size	1
TD.1.9	5	Read(10)	0x28	0x01	block size * 2	1
TD.1.10	6	Read(10)	0x28	0x01	block size	1
TD.1.11	7	Read(10)	0x28	0x02	block size	1
TD.1.12	8	Write(10)	0x2A	0x01	block size	1
TD.1.13	9	Test Unit Ready	0x00	--	block size	0
TD.1.14	10	Read(10)	0x28	0x01	block size	0
TD.1.15	11	Write(10)	0x2A	0x01	block size * 2	0
TD.1.16	12	Write(10)	0x2A	0x02	block size * 2	0
TD.1.17	13	Write(10)	0x2A	0x02	block size	0
TD.1.19	--	Test Unit Ready	0x00	--	0x0000	N/A

Table 2 Command Block Wrappers

6.4 Command Set Test

This section describes the Command Sets used in the test descriptions TD.2.1 to TD.2.2 above.

6.4.1 Inquiry

Bit Byte	7	6	5	4	3	2	1	0
0	Operation Code(12h)							
1	Reserved(0)						Obs(0)	EVPD(0)
2	PAGE CODE(0)							
3-4	[MSB] Allocation Length(see below) [LSB]							
5	Control(0)							

Test parameters:

Command is executed with Allocation Length = 00h, 01h, 02h, 04h, 05h, 24h, and FFh

6.4.2 Request Sense

Bit Byte	7	6	5	4	3	2	1	0
0	Operation Code(03h)							
1	Reserved(0)							DESC(0)
2	Reserved(0)							
3	Reserved(0)							
4	Allocation Length(see below)							
5	Control(0)							

Test parameters:

Command is executed with Allocation Length = 00h, 01h, 02h, 12h, and FFh

6.4.3 Test Unit Ready

Bit Byte	7	6	5	4	3	2	1	0
0	Operation Code(00h)							
1-4	Reserved(0)							
5	Control(0)							

Test parameters:

Command is executed once

6.4.4 Read(10)

Bit Byte	7	6	5	4	3	2	1	0
0	Operation Code(28h)							
1	RDPROTECT(0)			DPO(0)	FUA(0)	Reserved(0)		Obs(0)
2-5	[MSB] Logical Block Address(0) [LSB]							
6	Reserved(0)							
7-8	[MSB] Transfer Length(see below) [LSB]							
9	Control(0)							

Test parameters:

Command is executed with Transfer Length = 4 kB, 8 kB, 16 kB, 32 kB, and 64 kB

6.4.5 Read(12)

Bit Byte	7	6	5	4	3	2	1	0
0	Operation Code(A8h)							
1	RDPROTECT(0)			DPO(0)	FUA(0)	Reserved(0)		Obs(0)
2-5	[MSB] Logical Block Address(0) [LSB]							
6-9	[MSB] Transfer Length(see below) [LSB]							
10	Reserved(0)							
11	Control(0)							

Test parameters:

Command is executed with Transfer Length = 4 kB, 8 kB, 16 kB, 32 kB, and 64 kB

6.4.6 Read Capacity

Bit Byte	7	6	5	4	3	2	1	0
0	Operation Code(25h)							
1	Reserved(0)							Obs(0)
2-5	[MSB] Logical Block Address(0) [LSB]							
6-7	Reserved(0)							
8	Reserved(0)							PMI(0)
9	Control(0)							

Test parameters:

Command is executed once

6.4.7 Write(10)

Bit Byte	7	6	5	4	3	2	1	0
0	Operation Code(2Ah)							
1	WRPROTECT(0)			DPO(0)	FUA(0)	Reserved(0)		Obs(0)
2-5	Logical Block Address(0)							
6	Reserved(0)							
7-8	[MSB] Transfer Length(see below) [LSB]							
9	Control(0)							

Test parameters:

Command is executed with Transfer Length = 4 kB, 8 kB, 16 kB, 32 kB, and 64 kB

6.4.8 Write(12)

Bit Byte	7	6	5	4	3	2	1	0
0	Operation Code(AAh)							
1	WRPROTECT(0)			DPO(0)	FUA(0)	Reserved(0)		Obs(0)
2-5	Logical Block Address(0)							
6-9	[MSB] Transfer Length(see below) [LSB]							
10	Reserved(0)							
11	Control(0)							

Test parameters:

Command is executed with Transfer Length = 4 kB, 8 kB, 16 kB, 32 kB, and 64 kB

6.4.9 Mode Select(6)

Bit Byte	7	6	5	4	3	2	1	0
0	Operation Code(15h)							
1	Reserved(0)			PF(0)	Reserved(0)			SP(0)
2-3	Reserved(0)							
4	Parameter List Length(0)							
5	Control(0)							

Test parameters:

Command is executed once

6.4.10 Mode Select(10)

Bit Byte	7	6	5	4	3	2	1	0
0	Operation Code(55h)							
1	Reserved(0)			PF(0)	Reserved(0)			SP(0)
2-6	Reserved(0)							
7-8	[MSB] Parameter List Length(0) [LSB]							
5	Control(0)							

Test parameters:

Command is executed once

6.4.11 Mode Sense(6)

Bit Byte	7	6	5	4	3	2	1	0
0	Operation Code(1Ah)							
1	Reserved(0)				DBD(0)	Reserved(0)		
2	PC(0)		PAGE CODE(3Fh)					
3	Subpage Code(0)							
4	Allocation Length(8h)							
5	Control(0)							

Test parameters:

Command is executed once

6.4.12 Mode Sense(10)

Bit Byte	7	6	5	4	3	2	1	0
0	Operation Code(5Ah)							
1	Reserved(0)			LLBAA(0)	DBD(0)	Reserved(0)		
2	PC(0)		Page Code(3Fh)					
3	Subpage Code(0)							
4-6	Reserved(0)							
7-8	[MSB] Parameter List Length(8h) [LSB]							
9	Control(0)							

Test parameters:

Command is executed once

6.4.13 Prevent/Allow Medium Removal

Bit Byte	7	6	5	4	3	2	1	0
0	Operation Code(1Eh)							
1-3	Reserved(0)							Immed(0)
4	Power Conditions(0)				Reserved(0)		LOEJ(0)	Start(1)
5	Control(0)							

Test parameters:

Command is executed once

6.4.14 Start/Stop Unit

Bit Byte	7	6	5	4	3	2	1	0
0	Operation Code(1Bh)							
1	Reserved(0)							Immed(0)
2-3	Reserved(0)							
4	Reserved(0)							Prevent(0)
5	Control(0)							

Test parameters:

Command is executed once

6.4.15 Read TOC/PMA/ATIP

Bit Byte	7	6	5	4	3	2	1	0
0	Operation Code(43h)							
1	Reserved(0)						MSF(0)	Reserved (0)
2	Reserved(0)				Format(0)			
3-5	Reserved(0)							
6	Track/Session Number(0)							
7-8	[MSB] Allocation Length(4h) [LSB]							
9	Control(0)							

Test parameters:

Command is executed once

6.4.16 Verify(10)

Bit Byte	7	6	5	4	3	2	1	0
0	Operation Code(2Fh)							
1	Reserved(0)			DPO(0)	Reserved (0)	BLKVFY (0)	BYTCHK (0)	RELADR (0)
2-5	Logical Block Address(0)							
6	Reserved(0)							
7-8	[MSB] Verification Length(see below) [LSB]							
9	Control(0)							

Test parameters:

Command is executed with Verification Length = 4 kB, 8 kB, 16 kB, 32 kB, and 64 kB