Universal Serial Bus
Mass Storage Class

Bulk-Only Transport

Revision 1.0
September 31, 1999

Change History

Revision	Issue Date	Comments
0.7	September 23, 1998	Initial draft, pre-release
0.8	October 6, 1998	Revisions made at the Mass Storage DWG review – Irvine, CA
0.9	October 21, 1998	Revisions made at the Mass Storage DWG review – Plano, TX
0.9a	January 5, 1999	Revisions made at the Mass Storage DWG review – Tigard, OR
0.9b	February 1, 1999	Additions of LUN support - Milpitas, CA
1.0[RC1]	March 5, 1999	RR review - Midway, UT
1.0[RC2]	March 23, 1999	Revisions from reflector review comments
1.0[RC3]	March 29, 1999	Specification line by line review – Milpitas, CA
1.0[RC4]	June 21, 1999	Specification line by line review – RR21 – Milpitas, CA
1.0	September 31, 1999	Final Revision edits for Released Document – SLC, UT

Contributors

Al Rickey, Phoenix Technologies
Alan Haffner, Lexar Media
Bill Stanley, Adaptec
Calaimany Bhoopathi, Shuttle Technology
Curtis E. Stevens, Phoenix Technologies
Darrell Redford , Iomega Corporation
Dave Gilbert, In-System Design
David G. Lawrence, Global Technology Development
David L. Jolley, Iomega Corporation
David Luke, In-System Design
Eric Luttman, In-System Design
Glen Slick, Microsoft Corporation
Hiromichi Oribe, Hagiwara Sys-ComCo
Jan Matejica, PIMC/Philips
Jim Blackson, Y-E Data, Inc
Jim Quigley, Iomega Corporation
Johan Craeybeckk, PIMC/Philips
Jordan Brown, Sun Microsystems
Kenny Chu, Hagiwara Sys-ComCo
Kenichi Hamada, Y-E Data, Inc
Mark Williams, Microsoft Corporation
Masayuki Kitagawa, Mitsumi
Mike Chen, CMD Technology
Mike Glass, Microsoft Corporation
Mike Liebow, eTEK Labs
Mike Nguyen, TEAC
Mike Poulsen, Iomega Corporation
Moto Watanabe, Hagiwara
N.R. Devanathan, Shuttle Technology
Paramita Das, Sun Microsystems
Pat LaVarre, Iomega Corporation
Peter S'Heeren, PIMC/Philips
Ryota Okazaki, NEC Corporation
Sadao Yabuki, TEAC
Shigeyoshi Hashi, NEC Corporation
Shing F. Lin, Adaptec
Steve Bayless, Hewlett-Packard
Steve Kolokowsky, Anchor Chips
Steven Smith, eTEK Labs
Terry Moore, MCCI
Tim Bradshaw, Iomega Corp
Toyoko Shimizu, Y-E Data, Inc
Trenton Henry, SMSC
Troy Davidson, Iomega Corporation
Tsuyoshi Osawa, TEAC
Yuji Oishi, Hagiwara Sys-Com Co Ltd

Table of Contents

List of Tables

List of Figures

1 Specification Overview and Scope

1.1 Scope

A familiarity with the *USB 1.0* and *1.1 Specifications* and the *USB Mass Storage Class Specification Overview* is assumed.

This specification addresses Bulk-Only Transport, or in other words, transport of command, data, and status occurring solely via Bulk endpoints (not via Interrupt or Control endpoints). This specification only uses the default pipe to clear a STALL condition on the Bulk endpoints and to issue class-specific requests as defined below. This specification does not require the use of an Interrupt endpoint.

This specification defines support for logical units that share common device characteristics. Although this feature provides the support necessary to allow like mass storage devices to share a common USB interface descriptor, it is not intended to be used to implement interface bridge devices.

2 Terms and Abbreviations

2.1 Conventions

Numbers without annotation are decimal ---------- 1, 17, 23
Hexadecimal numbers are followed by an 'h' ---------- 1Fh, FCh, 38h
Binary numbers are followed by a 'b' ---------- 011b, 101b, 01110010b
Words in *italics* indicate terms defined by USB or by this specification ---------- *bRequest*, *dCSWTag*

2.2 Definitions

Command Block Wrapper (CBW)
A packet containing a command block and associated information.

Command Status Wrapper (CSW)
A packet containing the status of a command block.

Data-In
Indicates a transfer of data IN from the device to the host.

Data-Out
Indicates a transfer of data OUT from the host to the device.

Device Request
Requests from the host to the device using the default pipe.

Phase Error
An error returned by the device indicating that the results of processing further CBWs will be indeterminate until the device is reset.

Processed
Data received and controlled internally by the device to the point that the host need no longer be concerned about it.

Relevant
The amount of the data sent to the host by the device that is significant.

Reset Recovery
An error recovery procedure by which the host prepares the device for further CBWs.

Thin Diagonal
Cases where the host and device are in complete agreement about the data transfer. See Chapter 6 - Host/Device Data Transfers, for additional information regarding error cases and the "thin diagonal."

3 Functional Characteristics

3.1 Bulk-Only Mass Storage Reset *(class-specific request)*

This request is used to reset the mass storage device and its associated interface.

This class-specific request shall ready the device for the next CBW from the host.

The host shall send this request via the default pipe to the device. The device shall preserve the value of its bulk data toggle bits and endpoint STALL conditions despite the Bulk-Only Mass Storage Reset.

The device shall NAK the status stage of the device request until the Bulk-Only Mass Storage Reset is complete.

To issue the Bulk-Only Mass Storage Reset the host shall issue a device request on the default pipe of:

- *bmRequestType*: Class, Interface, host to device
- *bRequest* field set to 255 (FFh)
- *wValue* field set to 0
- *wIndex* field set to the interface number
- *wLength* field set to 0

Table 3.1 – Bulk-Only Mass Storage Reset

bmRequestType	*bRequest*	*wValue*	*wIndex*	*wLength*	Data
00100001b	11111111b	0000h	Interface	0000h	none

3.2 Get Max LUN *(class-specific request)*

The device *may* implement several logical units that share common device characteristics. The host uses *bCBWLUN* (see 5.1 Command Block Wrapper (CBW)) to designate which logical unit of the device is the destination of the CBW. The Get Max LUN device request is used to determine the number of logical units supported by the device. Logical Unit Numbers on the device shall be numbered contiguously starting from LUN 0 to a maximum LUN of 15 (Fh).

To issue a Get Max LUN device request, the host shall issue a device request on the default pipe of:

- *bmRequestType*: Class, Interface, device to host
- *bRequest* field set to 254 (FEh)
- *wValue* field set to 0
- *wIndex* field set to the interface number
- *wLength* field set to 1

Table 3.2 –Get Max LUN

bmRequestType	*bRequest*	*wValue*	*wIndex*	*wLength*	Data
10100001b	11111110b	0000h	Interface	0001h	1 byte

The device shall return one byte of data that contains the maximum LUN supported by the device. For example, if the device supports four LUNs then the LUNs would be numbered from 0 to 3 and the return value would be 3. If no LUN is associated with the device, the value returned shall be 0. The host shall not send a command block wrapper (CBW) to a non-existing LUN.

Devices that do not support multiple LUNs *may* STALL this command.

3.3 Host/Device Packet Transfer Order

The host shall send the CBW before the associated Data-Out, and the device shall send Data-In after the associated CBW and before the associated CSW. The host *may* request Data-In or CSW before sending the associated CBW.

If the *dCBWDataTransferLength* is zero, the device and the host shall transfer no data between the CBW and the associated CSW.

3.4 Command Queuing

The host shall not transfer a CBW to the device until the host has received the CSW for any outstanding CBW. If the host issues two consecutive CBWs without an intervening CSW or reset, the device response to the second CBW is indeterminate.

3.5 Bi-Directional Command Protocol

This specification does not provide for bi-directional data transfer in a single command.

4 Standard Descriptors

The device shall support the following standard USB descriptors:

- **Device**. Each USB device has one device descriptor (per *USB Specification*).
- **Configuration**. Each USB device has one default configuration descriptor, which supports at least one interface.
- **Interface**. The device shall support at least one interface, known herein as the Bulk-Only Data Interface. Some devices *may* support additional interfaces, to provide other capabilities.
- **Endpoint**. The device shall support the following endpoints, in addition to the default pipe that is required of all USB devices:
  (a) Bulk-In endpoint
  (b) Bulk-Out endpoint

  Some devices *may* support additional endpoints, to provide other capabilities. The host shall use the first reported Bulk-In and Bulk-Out endpoints for the selected interface.
- **String**. The device shall supply a unique serial number as detailed in 4.1.1 - Serial Number.

This specification defines no class-specific descriptors.

The rest of this section describes the standard USB device, configuration, interface, endpoint, and string descriptors for the device. For superseding information about these and other standard descriptors, see Chapter 9, "USB Device Framework," of the *USB Specification*.

4.1 Device Descriptor

Each USB device has one device descriptor (per *USB Specification*). The device shall specify the device class and subclass codes in the interface descriptor, and not in the device descriptor.

Table 4.1 - Device Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	Byte	12h	Size of this descriptor in bytes.
1	*bDescriptorType*	Byte	01h	DEVICE descriptor type.
2	*bcdUSB*	Word	????h	*USB Specification* Release Number in Binary-Coded Decimal (i.e. 2.10 = 210h). This field identifies the release of the *USB Specification* with which the device and is descriptors are compliant.
4	*bDeviceClass*	Byte	00h	**Class is specified in the interface descriptor.**
5	*bDeviceSubClass*	Byte	00h	**Subclass is specified in the interface descriptor.**
6	*bDeviceProtocol*	Byte	00h	**Protocol is specified in the interface descriptor.**
7	*bMaxPacketSize0*	Byte	??h	Maximum packet size for endpoint zero. (only 8, 16, 32, or 64 are valid **(08h, 10h, 20h, 40h)**).
8	*idVendor*	Word	????h	Vendor ID (assigned by the USB-**IF**).
10	*idProduct*	Word	????h	Product ID (assigned by the manufacturer).
12	*bcdDevice*	Word	????h	Device release number in binary-coded decimal.
14	*iManufacturer*	Byte	??h	Index of string descriptor describing the manufacturer.
15	*iProduct*	Byte	??h	Index of string descriptor describing this product.
16	*iSerialNumber*	Byte	??h	Index of string descriptor describing the device's serial number. **(Details in 4.1.1 below)**
17	*bNumConfigurations*	Byte	??h	Number of possible configurations.

NOTE: Information in this table is from the *USB Specification version 1.1 table 9-7*. **Bold text** has been added for clarifications when using these descriptors with this specification.

4.1.1 Serial Number

The *iSerialNumber* field shall be set to the index of the string descriptor that contains the serial number. The serial number shall contain at least 12 valid digits, represented as a UNICODE string. The last 12 digits of the serial number shall be unique to each USB *idVendor* and *idProduct* pair.

The host *may* generate a globally unique identifier by concatenating the 16 bit *idVendor*, the 16 bit *idProduct* and the value represented by the last 12 characters of the string descriptor indexed by *iSerialNumber*.

The field *iSerialNumber* is an index to a string descriptor and does not contain the string itself. An example format for the String descriptor is shown below.

Table 4.2 - Example Serial Number Format

Offset	Field	Size	Value	Description
0	*bLength*	Byte	??h	Size of this descriptor in bytes - Minimum of 26 (1Ah)
1	*bDescriptorType*	Byte	03h	STRING descriptor type
2	*wString1*	Word	00??h	Serial number character 1
4	*wString2*	Word	00??h	Serial number character 2
6	*wString3*	Word	00??h	Serial number character 3
:	*:*	Word	:	:
:	*:*	Word	:	:
n x 2	*wStringn*	Word	00??h	Serial number character n
Shall be at least 12 characters long				

4.1.2 Valid Serial Number Characters

The following table defines the valid characters that the device shall use for the serial number.

Table 4.3 - Valid Serial Number Characters

Numeric	ASCII
0030h through 0039h	"0" through "9"
0041h through 0046h	"A" through "F"

4.2 Configuration Descriptor

Table 4.4 - Configuration Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	Byte	09h	Size of this descriptor in bytes.
1	*bDescriptorType*	Byte	02h	CONFIGURATION Descriptor Type.
2	*wTotalLength*	Word	????h	Total length of data returned for this configuration. Includes the combined length of all descriptors (configuration, interface, endpoint, and class- or vendor-specific) returned for this configuration.
4	*bNumInterfaces*	Byte	??h	Number of interfaces supported by this configuration. **The device shall support at least the Bulk-Only Data Interface.**
5	*bConfigurationValue*	Byte	??h	Value to use as an argument to the *SetConfiguration()* request to select this configuration.
6	*iConfiguration*	Byte	??h	Index of string descriptor describing this configuration.
7	*bmAttributes*	Byte	?0h	Configuration characteristics: **Bit** **Description** 7 Reserved (set to one) 6 Self-powered 5 Remote Wakeup 4..0 Reserved (reset to zero) **Bit** 7 is reserved and must be set to one for historical reasons. **For a full description of this *bmAttributes* bitmap, see the *USB 1.1 Specification*.**
8	*MaxPower*	Byte	??h	Maximum power consumption of the USB device from the bus in this specific configuration when the device is fully operational. Expressed in 2mA units (i.e. 50 = 100mA)

NOTE: Information in this table is from the *USB Specification version 1.1 table 9-8*. **Bold text** has been added for clarifications when using these descriptors with this specification.

4.3 Interface Descriptors

The device shall support at least one interface, known herein as the Bulk-Only Data Interface. The Bulk-Only Data Interface uses three endpoints.

Composite mass storage devices *may* support additional interfaces, to provide other features such as audio or video capabilities. This specification does not define such interfaces.

The interface *may* have multiple alternate settings. The host shall examine each of the alternate settings to look for the *bInterfaceProtocol* and *bInterfaceSubClass* it supports optimally.

Table 4.5 – Bulk-Only Data Interface Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	Byte	09h	Size of this descriptor in bytes.
1	*bDescriptorType*	Byte	04h	INTERFACE Descriptor Type.
2	*bInterfaceNumber*	Byte	0?h	Number of interface. Zero-based value identifying the index in the array of concurrent interfaces supported by this configuration.
3	*bAlternateSetting*	Byte	??h	Value used to select alternate setting for the interface identified in the prior field.
4	*bNumEndpoints*	Byte	??h	Number of endpoints used by this interface (excluding endpoint zero). **This value shall be at least 2.**
5	*bInterfaceClass*	Byte	08h	**MASS STORAGE Class**.
6	*bInterfaceSubClass*	Byte	0?h	Subclass code (assigned by the USB-**IF**). **Indicates which industry standard command block definition to use. Does not specify a type of storage device such as a floppy disk or CD-ROM drive. (See *USB Mass Storage Overview Specification*)**
7	*bInterfaceProtocol*	Byte	50h	**BULK-ONLY TRANSPORT. (See *USB Mass Storage Overview Specification)***
8	*iInterface*	Byte	??h	Index to string descriptor describing this interface.

NOTE: Information in this table is from the *USB Specification version 1.1 table 9-9*. **Bold text** has been added for clarifications when using these descriptors with this specification.

4.4 Endpoint Descriptors

The device shall support at least three endpoints: Control, Bulk-In and Bulk-Out.

Each USB device defines a Control endpoint (Endpoint 0). This is the default endpoint and does not require a descriptor.

4.4.1 Bulk-In Endpoint

The Bulk-In endpoint is used for transferring data and status from the device to the host.

Table 4.6 - Bulk-In Endpoint Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	Byte	07h	Size of this descriptor in bytes.
1	*bDescriptorType*	Byte	05h	ENDPOINT Descriptor Type.
2	*bEndpointAddress*	Byte	8?h	The address of this endpoint on the USB device. The address is encoded as follows: **Bit** — **Description** 3..0 — The endpoint number 6..4 — Reserved, set to 0 7 — 1 = In
3	*bmAttributes*	Byte	02h	This is a Bulk endpoint.
4	*wMaxPacketSize*	Word	00??h	Maximum packet size. Shall be 8, 16, 32 or 64 bytes (08h, 10h, 20h, 40h).
6	*bInterval*	Byte	00h	Does not apply to Bulk endpoints.

4.4.2 Bulk-Out Endpoint

The Bulk-Out endpoint is used for transferring command and data from the host to the device.

Table 4.7 – Bulk-Out Endpoint Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	Byte	07h	Size of this descriptor in bytes.
1	*bDescriptorType*	Byte	05h	ENDPOINT descriptor type.
2	*bEndpointAddress*	Byte	0?h	The address of this endpoint on the USB device. This address is encoded as follows: **Bit** **Description** 3..0 Endpoint number 6..4 Reserved, set to 0 7 0 = Out
3	*bmAttributes*	Byte	02h	This is a Bulk endpoint.
4	*wMaxPacketSize*	Word	00??h	Maximum packet size. Shall be 8, 16, 32 or 64 bytes (08h, 10h, 20h, or 40h).
6	*bInterval*	Byte	00h	Does not apply to Bulk endpoints.

5 Command/Data/Status Protocol

Figure 1 - Command/Data/Status Flow shows the flow for Command Transport, Data-In, Data-Out and Status Transport. The following sections define Command and Status Transport. Figure 2 - Status Transport Flow shows a detailed diagram of Status Transport. The following sections outline the various conditions for host/device communication, possible errors, and recovery procedures.

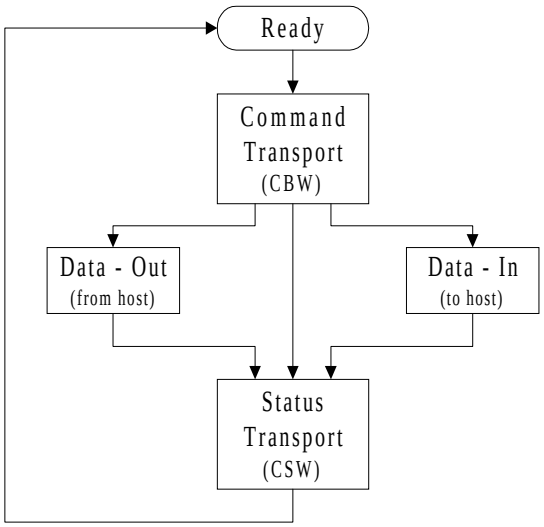

Figure 1 - Command/Data/Status Flow

5.1 Command Block Wrapper (CBW)

The CBW shall start on a packet boundary and shall end as a short packet with exactly 31 (1Fh) bytes transferred. Fields appear aligned to byte offsets equal to a multiple of their byte size. All subsequent data and the CSW shall start at a new packet boundary. All CBW transfers shall be ordered with the LSB (byte 0) first (little endian). Refer to the *USB Specification* Terms and Abbreviations for clarification.

Table 5.1 - Command Block Wrapper

bit / Byte	7	6	5	4	3	2	1	0
0-3	*dCBWSignature*							
4-7	*dCBWTag*							
8-11 (08h-0Bh)	*dCBWDataTransferLength*							
12 (0Ch)	*bmCBWFlags*							
13 (0Dh)	Reserved (0)				*bCBWLUN*			
14 (0Eh)	Reserved (0)			*bCBWCBLength*				
15-30 (0Fh-1Eh)	*CBWCB*							

dCBWSignature:

Signature that helps identify this data packet as a CBW. The signature field shall contain the value 43425355h (little endian), indicating a CBW.

dCBWTag:

A Command Block Tag sent by the host. The device shall echo the contents of this field back to the host in the *dCSWTag* field of the associated CSW. The *dCSWTag* positively associates a CSW with the corresponding CBW.

dCBWDataTransferLength:

The number of bytes of data that the host expects to transfer on the Bulk-In or Bulk-Out endpoint (as indicated by the *Direction* bit) during the execution of this command. If this field is zero, the device and the host shall transfer no data between the CBW and the associated CSW, and the device shall ignore the value of the *Direction* bit in *bmCBWFlags.*

bmCBWFlags:

The bits of this field are defined as follows:

Bit 7 *Direction* - the device shall ignore this bit if the *dCBWDataTransferLength* field is zero, otherwise:
0 = Data-Out from host to the device,
1 = Data-In from the device to the host.
Bit 6 Obsolete. The host shall set this bit to zero.
Bits 5..0 Reserved - the host shall set these bits to zero.

bCBWLUN:

The device Logical Unit Number (LUN) to which the command block is being sent. For devices that support multiple LUNs, the host shall place into this field the LUN to which this command block is addressed. Otherwise, the host shall set this field to zero.

bCBWCBLength:

The valid length of the *CBWCB* in bytes. This defines the valid length of the command block. The only legal values are 1 through 16 (01h through 10h). All other values are reserved.

CBWCB:

The command block to be executed by the device. The device shall interpret the first *bCBWCBLength* bytes in this field as a command block as defined by the command set identified by *bInterfaceSubClass*. If the command set supported by the device uses command blocks of fewer than 16 (10h) bytes in length, the significant bytes shall be transferred first, beginning with the byte at offset 15 (Fh). The device shall ignore the content of the *CBWCB* field past the byte at offset (15 + *bCBWCBLength* - 1).

5.2 Command Status Wrapper (CSW)

The CSW shall start on a packet boundary and shall end as a short packet with exactly 13 (0Dh) bytes transferred. Fields appear aligned to byte offsets equal to a multiple of their byte size. All CSW transfers shall be ordered with the LSB (byte 0) first (little endian). Refer to the *USB Specification* Terms and Abbreviations for clarification.

Table 5.2 - Command Status Wrapper

bit / Byte	7	6	5	4	3	2	1	0
0-3	*dCSWSignature*							
4-7	*dCSWTag*							
8-11 (8-Bh)	*dCSWDataResidue*							
12 (Ch)	*bCSWStatus*							

dCSWSignature:

Signature that helps identify this data packet as a CSW. The signature field shall contain the value 53425355h (little endian), indicating CSW.

dCSWTag:

The device shall set this field to the value received in the *dCBWTag* of the associated CBW.

dCSWDataResidue:

For Data-Out the device shall report in the *dCSWDataResidue* the difference between the amount of data expected as stated in the *dCBWDataTransferLength*, and the actual amount of data processed by the device. For Data-In the device shall report in the *dCSWDataResidue* the difference between the amount of data expected as stated in the *dCBWDataTransferLength* and the actual amount of relevant data sent by the device. The *dCSWDataResidue* shall not exceed the value sent in the *dCBWDataTransferLength*.

bCSWStatus:

bCSWStatus indicates the success or failure of the command. The device shall set this byte to zero if the command completed successfully. A non-zero value shall indicate a failure during command execution according to the following table:

Table 5.3 - Command Block Status Values

Value	Description
00h	Command Passed ("good status")
01h	Command Failed
02h	Phase Error
03h and 04h	Reserved (Obsolete)
05h to FFh	Reserved

5.3 Data Transfer Conditions

This section describes how the host and device remain synchronized.

The host indicates the expected transfer in the CBW using the *Direction* bit and the *dCBWDataTransferLength* field. The device then determines the actual direction and data transfer length. The device responds as defined in 6 - Host/Device Data Transfers by transferring data, STALLing endpoints when specified, and returning the appropriate CSW.

5.3.1 Command Transport

The host shall send each CBW, which contains a command block, to the device via the Bulk-Out endpoint. The CBW shall start on a packet boundary and end as a short packet with exactly 31 (1Fh) bytes transferred.

The device shall indicate a successful transport of a CBW by accepting (ACKing) the CBW. If the CBW is not valid see 6.6.1 - CBW Not Valid. If the host detects a STALL of the Bulk-Out endpoint during command transport, the host shall respond with a Reset Recovery (see 5.3.4 - Reset Recovery).

5.3.2 Data Transport

All data transport shall begin on a packet boundary.

The host shall attempt to transfer the exact number of bytes to or from the device as specified by the *dCBWDataTransferLength* and the *Direction* bit. The device shall respond as specified in 6 - Host/Device Data Transfers.

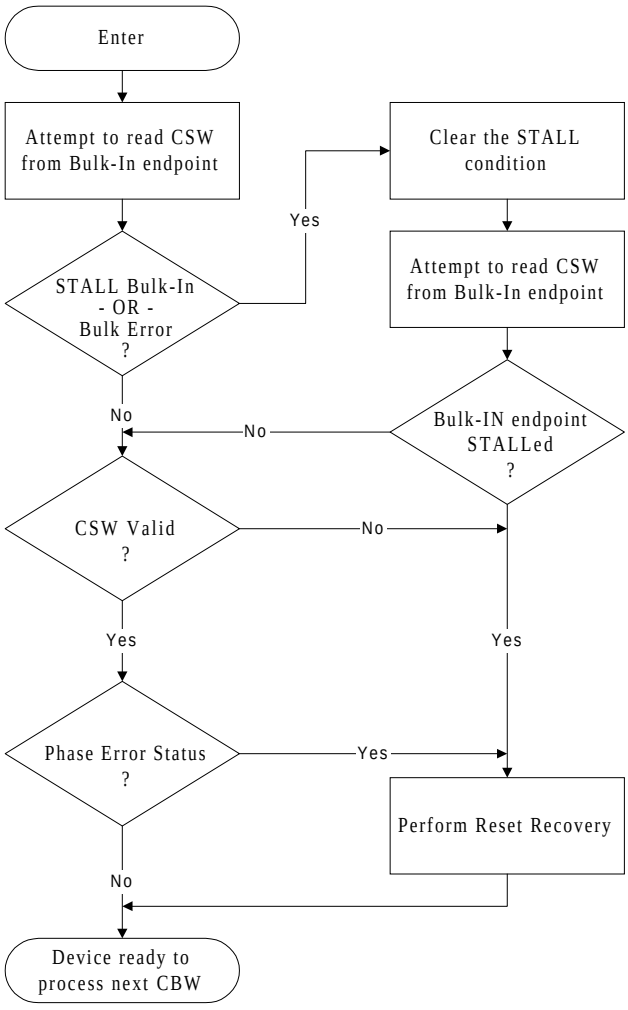

Figure 2 - Status Transport Flow

To report an error before data transport completes and to maximize data integrity, the device *may* terminate the command by STALLing the endpoint in use (the Bulk-In endpoint during data in, the Bulk-Out endpoint during data out).

5.3.3 Status Transport

The device shall send each CSW to the host via the Bulk-In endpoint. The CSW shall start on a packet boundary and end as a short packet with exactly 13 (Dh) bytes transferred. Figure 2 - Status Transport Flow defines the algorithm the host shall use for any CSW transfer.

The CSW indicates to the host the status of the execution of the command block from the corresponding CBW. The *dCSWDataResidue* field indicates how much of the data transferred is to be considered processed or relevant. The host shall ignore any data received beyond that which is relevant.

5.3.3.1 Phase Error

The host shall perform a Reset Recovery when Phase Error status is returned in the CSW.

5.3.4 Reset Recovery

For Reset Recovery the host shall issue in the following order: :

(a) a Bulk-Only Mass Storage Reset
(b) a *Clear Feature HALT* to the Bulk-In endpoint
(c) a *Clear Feature HALT to* the Bulk-Out endpoint

6 Host/Device Data Transfers

6.1 Overview

A Bulk-Only Protocol transaction begins with the host sending a CBW to the device and attempting to make the appropriate data transfer (In, Out or none). The device receives the CBW, checks and interprets it, attempts to satisfy the host's request, and returns status via a CSW. This section describes in more detail this interaction between the host and the device during normal and abnormal Bulk-Only Protocol transactions.

6.2 Valid and Meaningful CBW

The host communicates its intent to the device through the CBW. The device performs two verifications on every CBW received. First, the device verifies that what was received is a valid CBW. Next, the device determines if the data within the CBW is meaningful.

The device shall not use the contents of the *dCBWTag* in any way other than to copy its value to the *dCSWTag* of the corresponding CSW.

6.2.1 Valid CBW

The device shall consider the CBW valid when:

- The CBW was received after the device had sent a CSW or after a reset,
- the CBW is 31 (1Fh) bytes in length,
- and the *dCBWSignature* is equal to 43425355h.

6.2.2 Meaningful CBW

The device shall consider the contents of a valid CBW meaningful when:

- no reserved bits are set,
- the *bCBWLUN* contains a valid LUN supported by the device,
- and both *bCBWCBLength* and the content of the *CBWCB* are in accordance with *bInterfaceSubClass*.

6.3 Valid and Meaningful CSW

The device generally communicates the results of its attempt to satisfy the host’s request through the CSW. The host performs two verifications on every CSW received. First, the host verifies that what was received is a valid CSW Next, the host determines if the data within the CSW is meaningful.

6.3.1 Valid CSW

The host shall consider the CSW valid when:

- the CSW is 13 (Dh) bytes in length,
- and the *dCSWSignature* is equal to 53425355h,
- the *dCSWTag* matches the *dCBWTag* from the corresponding CBW.

6.3.2 Meaningful CSW

The host shall consider the contents of the CSW meaningful when:

either the *bCSWStatus* value is 00h or 01h and the *dCSWDataResidue* is less than or equal to *dCBWDataTransferLength*..

or the *bCSWStatus* value is 02h.

6.4 Device Error Handling

The device may not be able to fully satisfy the host's request. At the point when the device discovers that it cannot fully satisfy the request, there may be a Data-In or Data-Out transfer in progress on the bus, and the host may have other pending requests. The device *may* cause the host to terminate such transfers by STALLing the appropriate pipe.

The response of a device to a CBW that is not meaningful is not specified.

Please note that whether or not a STALL handshake actually appears on the bus depends on whether or not there is a transfer in progress at the point in time when the device is ready to STALL the pipe.

6.5 Host Error Handling

If the host receives a CSW which is not valid, then the host shall perform a Reset Recovery. If the host receives a CSW which is not meaningful, then the host *may* perform a Reset Recovery.

6.6 Error Classes

In every transaction between the host and the device, there are four possible classes of errors. These classes are not always independent of each other and may occur at any time during the transaction.

6.6.1 CBW Not Valid

If the CBW is not valid, the device shall STALL the Bulk-In pipe. Also, the device shall either STALL the Bulk-Out pipe, or the device shall accept and discard any Bulk-Out data. The device shall maintain this state until a Reset Recovery.

6.6.2 Internal Device Error

The device may detect an internal error for which it has no reliable means of recovery other than a reset. The device shall respond to such conditions by:

either STALLing any data transfer in progress and returning a Phase Error status (*bCSWStatus* = 02h).

or STALLing all further requests on the Bulk-In and the Bulk-Out pipes until a Reset Recovery.

6.6.3 Host/Device Disagreements

After recognizing that a CBW is valid and meaningful, and in the absence of internal errors, the device may detect a condition where it cannot meet the host's expectation for data transfer, as indicated by the *Direction* bit of the *bmCBWFlags* field and the *dCBWDataTransferLength* field of the CBW. In some of these cases, the device may require a reset to recover. In these cases, the device shall return Phase Error status (*bCSWStatus* = 02h). Details on which cases result in Phase Error vs. non-Phase Error status are given in 6.7 The Thirteen Cases.

6.6.4 Command Failure

After recognizing that a CBW is valid and meaningful, the device may still fail in its attempt to satisfy the command. The device shall report this condition by returning a Command Failed status (*bCSWStatus* = 01h).

6.7 The Thirteen Cases

This section describes the thirteen possible cases of host expectations and device intent in the absence of overriding error conditions. Table 6.1 – Host/Device Data Transfer Matrix graphically displays these thirteen cases.

Important notes about the thirteen cases.

- Cases **(1)**, **(6)** and **(12)** represent the majority of host and device transactions. They indicate those conditions where the host and device agree as to the direction and amount of data to be transferred. These cases are also referred to as "the thin diagonal."
- Any host or device behavior not specifically outlined in the following sections shall be considered outside this specification and the results are indeterminate.

Table 6.1 - Host/Device Data Transfer Matrix

		HOST		
		H*n*	**H*i***	**H*o***
DEVICE	**D*n***	**(1)** H*n* = D*n*	**(4)** H*i* > D*n*	**(9)** H*o* > D*n*
	D*i*	**(2)** H*n* < D*i*	**(5)** H*i* > D*i*	**(10)** H*o* <> D*i*
			(6) H*i* = D*i*	
			(7) H*i* < D*i*	
	D*o*	**(3)** H*n* < D*o*	**(8)** H*i* <> D*o*	**(11)** H*o* > D*o*
				(12) H*o* = D*o*
				(13) H*o* < D*o*

LEGEND	
Host Expectation	
H*n*	Host expects no data transfers
H*i*	Host expects to receive data from the device
H*o*	Host expects to send data to the device
Device Intent	
D*n*	Device intends to transfer no data
D*i*	Device intends to send data to the host
D*o*	Device intends to receive data from the host

6.7.1 H*n* - Host expects no data transfers

Cases: **(1)** H*n* = D*n*
(2) H*n* < D*i*
(3) H*n* < D*o*

These cases occur when *dCBWDataTransferLength* is zero. This indicates that the host is not expecting to send or receive any data to or from the device.

The **general** requirements of these cases are:

- The value of the *Direction* bit shall not influence the results of these cases.

The specific **host** requirements are:

1. The host shall send a valid and meaningful CBW.
   - The host *may* set or clear the *Direction* bit.
2. The host shall attempt to receive a CSW.
3. On a STALL condition receiving the CSW, then:
   - The host shall clear the Bulk-In pipe.
   - The host shall attempt to receive the CSW again.
4. When the CSW is valid and meaningful, then:
   - [Case **(1)**]
     The *bCSWStatus* = 00h or 01h, and the *dCSWDataResidue* is 0.
   - [Case **(2)** or **(3)**]
     If *bCSWStatus* = 02h, then:
     The host shall Ignore the value of the *dCSWDataResidue*.
     The host shall Perform a Reset Recovery.

The specific **device** requirements are:

1. The device shall receive a CBW.
2. When the CBW is valid and meaningful, then:
   - The device shall attempt the command.

- [Case **(1)**]
  If the device had no data to send or receive, then:
  The device shall set *bCSWStatus* to 00h or 01h.
  The device shall set the *dCSWDataResidue* to 0.
- [Case **(2)** or **(3)**]
  If the device did have data to send or receive, then:
  The device shall set *bCSWStatus* to 02h.

3. The device shall return a valid and meaningful CSW.
   - The device *may* STALL the Bulk-In pipe if *bCSWStatus* is not 00h or 01h.

6.7.2 H*i* - Host expects to receive data from the device

Cases: **(4)** H*i* > D*n*
(5) H*i* > D*i*
(6) H*i* = D*i*
(7) H*i* < D*i*
(8) H*i* <> D*o*

These cases occur when *dCBWDataTransferLength* is non zero and the *Direction* bit is 1 (Data-In). This indicates that the host is expecting to receive data from the device.

The specific **host** requirements are:

1. The host shall send a valid and meaningful CBW.
2. The host shall attempt to receive data from the device.
3. On a STALL condition receiving data, then:
   - The host shall accept the data received.
   - The host shall clear the Bulk-In pipe.
4. The host shall attempt to receive a CSW.
5. On a STALL condition receiving the CSW, then:
   - The host shall clear the Bulk-In pipe.
   - The host shall again attempt to receive the CSW.
6. When the CSW is valid and meaningful, then:
   - [Case **(4)**, **(5)**, or **(6)**]
     If *bCSWStatus* = 00h or 01h, then:
     The host shall determine the amount of relevant data received from the difference between *dCBWDataTransferLength* and the value of *dCSWDataResidue*.
   - [Case **(7)** or **(8)**]
     If *bCSWStatus* = 02h, then:
     The host shall ignore the value of the *dCSWDataResidue*.
     The host shall perform a Reset Recovery.

The specific **device** requirements are:

1. The device shall receive a CBW.
2. When the CBW is valid and meaningful, then:
   - The device shall attempt the command.
   - [Case **(6)**]
     If the device intends to send *dCBWDataTransferLength*, then:
     The device shall send *dCBWDataTransferLength* bytes of data.
     The device shall set *bCSWStatus* to 00h or 01h.
     The device shall set *dCSWDataResidue* to zero.
   - [Case **(4)**or **(5)**]
     If the device intends to send less data than the host indicated, then:
     The device shall send the intended data.
     The device *may* send fill data to pad up to a total of *dCBWDataTransferLength*.
     If the device actually transfers less data than the host indicated, then:
     The device *may* end the transfer with a short packet.
     The device shall STALL the Bulk-In pipe.
     The device shall set *bCSWStatus* to 00h or 01h.
     The device shall set *dCSWDataResidue* to the difference between *dCBWDataTransferLength* and the actual amount of relevant data sent.

- [Case **(7)** or **(8)**]
  If the device either intends to send more data than the host indicated or intends to receive data from the host, then:
    The device *may* send data up to a total of *dCBWDataTransferLength.*
    If the device actually transfers less data than the host indicated, then:
      The device *may* end the transfer with a short packet.
      The device shall STALL the Bulk-In pipe.
    If the device actually transfers *dCBWDataTransferLength* then:
      The device *may* STALL the Bulk-In pipe.
    The device shall set *bCSWStatus* to 02h.

3. The device shall return a valid and meaningful CSW:

6.7.3 H*o* - Host expects to send data to the device

Cases: **(9)** H*o* > D*n*
(10) H*o* <> D*i*
(11) H*o* > D*o*
(12) H*o* = D*o*
(13) H*o* < D*o*

These cases occur when *dCBWDataTransferLength* is non zero-and the *Direction* bit is 0 (Data-Out). This indicates that the host is expecting to send data to the device.

The **general** requirement of these cases is:

- The host shall not send zero length packets.

The specific **host** requirements are:

1. The host shall send a valid and meaningful CBW.
2. The host shall send data to the device.
   - The host shall send a short packet only at the end of the data transfer.
3. On a STALL condition sending data, then:
   - The host shall clear the Bulk-Out pipe.
4. The host shall attempt to receive a CSW.
5. On a STALL condition receiving the CSW, then:
   - The host shall clear the Bulk-In pipe.
   - The host shall again attempt to receive the CSW.
6. When the CSW is valid and meaningful, then:
   - [Case **(9)**, **(11)**, or **(12)**]
     If *bCSWStatus* = 00h or 01h, then:
       The host shall determine the amount of data that was processed from the difference of *dCBWDataTransferLength* and the value of *dCSWDataResidue*.
   - [Case **(10)** or **(13)**]
     If *bCSWStatus* = 02h, then:
       The host shall ignore the value of the *dCSWDataResidue*.
       The host shall perform a Reset Recovery.

The specific **device** requirements are:

1. The device shall receive a CBW.
2. When the CBW is valid and meaningful, then:
   - The device shall attempt the command.
   - [Case **(9)**, **(11)**, or **(12)**]
     If the device intends to process less than or equal to the amount of data that the host indicated, then:
       The device shall receive the intended data.
         The device shall either accept a total of *dCBWDataTransferLength*, or end the transfer prematurely by STALLing the Bulk-Out pipe.
       The device shall set *bCSWStatus* to 00h or 01h.
       The device shall set *dCSWDataResidue* to the difference between *dCBWDataTransferLength* and the actual amount of data that was processed by the device.
   - [Case **(10)** or **(13)**]
     If the device either intends to process more data than the host indicated or intends to send data, then:

The device *may* receive data up to a total of *dCBWDataTransferLength.*

The device shall either accept a total of *dCBWDataTransferLength*, or end the transfer prematurely by STALLing the Bulk-Out pipe.

The device shall set *bCSWStatus* to 02h.

3. The device shall return a valid and meaningful CSW.
   - The device *may* STALL the Bulk-In pipe if *bCSWStatus* is not 00h or 01h.