Universal Serial Bus
Lockable Storage Devices
Feature Specification

Revision 1.0

August 1, 2007

Scope of this Revision

Distinguish newly standard Lockable Storage interfaces from legacy Mass Storage interfaces.

Revision History

Rev	Date	Filename	Comments
1.0	August 1, 2007	Msc-lock-1.0.doc	Initial Release

USB Lockable Storage Devices FS

Copyright © 2007, USB Implementers Forum, Inc.
All rights reserved.

A LICENSE IS HEREBY GRANTED TO REPRODUCE THIS SPECIFICATION FOR INTERNAL USE ONLY. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, IS GRANTED OR INTENDED HEREBY.

USB-IF AND THE AUTHORS OF THIS SPECIFICATION EXPRESSLY DISCLAIM ALL LIABILITY FOR INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS, RELATING TO IMPLEMENTATION OF INFORMATION IN THIS SPECIFICATION. USB-IF AND THE AUTHORS OF THIS SPECIFICATION ALSO DO NOT WARRANT OR REPRESENT THAT SUCH IMPLEMENTATION(S) WILL NOT INFRINGE THE INTELLECTUAL PROPERTY RIGHTS OF OTHERS.

THIS SPECIFICATION IS PROVIDED "AS IS" AND WITH NO WARRANTIES, EXPRESS OR IMPLIED, STATUTORY OR OTHERWISE. ALL WARRANTIES ARE EXPRESSLY DISCLAIMED. NO WARRANTY OF MERCHANTABILITY, NO WARRANTY OF NON-INFRINGEMENT, NO WARRANTY OF FITNESS FOR ANY PARTICULAR PURPOSE, AND NO WARRANTY ARISING OUT OF ANY PROPOSAL, SPECIFICATION, OR SAMPLE.

IN NO EVENT WILL USB-IF OR USB-IF MEMBERS BE LIABLE TO ANOTHER FOR THE COST OF PROCURING SUBSTITUTE GOODS OR SERVICES, LOST PROFITS, LOSS OF USE, LOSS OF DATA OR ANY INCIDENTAL, CONSEQUENTIAL, INDIRECT, OR SPECIAL DAMAGES, WHETHER UNDER CONTRACT, TORT, WARRANTY, OR OTHERWISE, ARISING IN ANY WAY OUT OF THE USE OF THIS SPECIFICATION, WHETHER OR NOT SUCH PARTY HAD ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

CERTAIN PATENT RIGHTS HAVE BEEN DISCLOSED THAT MAY BE RELEVANT TO THE USE OF THIS SPECIFICATION. THE DISCLOSURE IS AVAILABLE AT http://www.usb.org/developers/devclass_docs#approved. USB-IF HAS NOT EVALUATED THE DISCLOSURE AND TAKES NO POSITION WITH RESPECT TO IT. USERS OF THIS SPECIFICATION ARE RESPONSIBLE FOR MAKING THEIR OWN DECISIONS IN LIGHT OF THE DISCLOSURE.

Please send comments via electronic mail to:

msc-lock@usb.org

Acknowledgement of USB Lockable Storage Technical Contribution

The authors of this specification would like to recognize the following people who participated in the USB Lockable Storage Feature Specification technical working groups. We would also like to thank others in the Promoter companies and throughout the industry who contributed to the development of this specification.

Ed Beeman	2010Tech
Mark Scott-Nash	Cornice
Keith Kleipin	Cypress Semiconductor
Amit Nanda	Cypress Semiconductor
Greg Reynolds	Cypress Semiconductor
Brian Tuttle	Cypress Semiconductor
Jerry Chen	Genesys Logic
Christian Schneckenburger	Infineon Technologies
Martin Furuhjelm	Lexar Media
John Geldman	Lexar Media
Daniel Grassi	Lexar Media
Pat LaVarre	Lexar Media
Steve Whitney	Littlefuse
Paul E. Berg	MCCI
Peter FitzRandolph	MCCI
Hao Cheng	Microsoft
Mark Benedikt	Microsoft
Emily Hill	Microsoft
Thomas Block	Nokia
Jithendra Bethur	Phoenix Technologies
Tony Ahwal	SanDisk
Steve Gross	SanDisk
Mike McMurdie	SanDisk
Avraham Shimor	SanDisk
Dana Zelitzki	SanDisk
Bryan Grimes	SecureWave
Thad Mann	SecureWave
Wesley Cheng 鄭欽維	Silicon Motion
Sam Lin	Silicon Motion
Paul U	Silicon Motion
Arthur Chih	Skymedi
Mark Bohm	SMSC
Mark Loch	SMSC
Leila Movahedian	ST Microelectronics
Curtis E. Stevens	Western Digital Technologies

Table of Contents

1 Introduction 9

1.1 Purpose 9

1.2 Scope 9

1.3 Related Documents 10

1.4 Terms and State Diagram Conventions 11

1.4.1 Terms 11

1.4.2 State Diagram Conventions 13

2 Management Overview 14

2.1 Goals 14

2.2 Ease of Use 14

2.3 Privacy 15

2.4 Compatibility 15

3 Functional Characteristics 17

3.1 Interface States 17

3.2 LU States 19

4 Operational Model 21

4.1 Operations 21

4.1.1 Cycle Power 21

4.1.2 Cycle Presence 21

4.1.3 Personalize 22

4.1.4 Unlock 22

4.1.5 Lock 23

4.1.6 Repersonalize 23

4.1.7 Depersonalize 23

4.1.8 Recover Media 23

4.1.9 Query 24

4.2 Interface State Diagram 25

4.2.1 AnyPersonal 25

4.2.2 CyclingPresence 26

4.2.3 AllImpersonal 26

4.3 LU State Diagram 28

4.3.1 Locked 28

4.3.2 Unlocked 29

4.3.3 Impersonal 30

5 Data Structures 32

5.1 USB Descriptor Template (UDB) 33

5.2 Device Descriptor (DD) 34

5.3 Interface Descriptor (ID) 35
5.4 Lockable Storage Interface Extension Descriptor (LSIED) 36
5.5 Phrase Data (PD) 37
5.6 Hint Data (HD) 38
5.7 Access Data (AD) 39
5.8 Lock Data (LD) 40
6 Requests 43
6.1 Get Requests 44
6.1.1 Get Lock In (GLI) 45
6.2 Put Requests 46
6.2.1 Store Passphrase Out (SPO) 47
6.2.2 Match Passphrase Out (MPO) 48
6.2.3 Change Passphrase Out (CPO) 49
6.2.4 Erase Passphrase Out (EPO) 50
6.2.5 Erase Forgotten Passphrase (EFP) 51
6.2.6 Lock Again (LA) 52
6.2.7 Change Interface Access Out (CIAO) 53
A. Appendix: Assigned Codes 55
A.1 Assigned bmDescriptorType Codes 55
A.2 Assigned bRequest and wValue Codes 56
A.3 Assigned Compatible IDs 57
A.3.1 Negotiable IDs 57

List of Figures

Figure 1 - State & Transition Notation 13

Figure 2 – Functional Interface Behavior 17

Figure 3 - Functional LU Behavior 19

Figure 4 - Lockable Interface States 25

Figure 5 - Lockable LU States 28

List of Tables

Table 1 - Descriptors 32
Table 2 - USB Descriptor 33
Table 3 - Device Descriptor 34
Table 4 - Interface Descriptor 35
Table 5 - Lockable Storage Interface Extension Descriptor (LSIED) 36
Table 6 - Phrase Data (PD) 37
Table 7 - Hint Data (HD) 38
Table 8 - Access Data (AD) 39
Table 9 - Lock Data (LD) 40
Table 10 - Class Specific Requests 43
Table 11 - Get Request Template 44
Table 12 - Get Lock In (GLI) 45
Table 13 – Put Request Template 46
Table 14 - Store Passphrase Out (SPO) 47
Table 15 - Match Passphrase Out (MPO) 48
Table 16 - Change Passphrase Out (CPO) 49
Table 17 - Erase Passphrase Out (EPO) 50
Table 18 - Erase Forgotten Passphrase (EFP) 51
Table 19 - Lock Again (LA) 52
Table 20 – Change Interface Access Out (CIAO) 53

1 Introduction

Today there is no standard infrastructure in USB to control access to USB storage devices.

Loss and theft of these devices concerns both consumers and corporations. Consumers want to avoid accidentally sharing personal information. Corporations want to avoid accidentally sharing trade secrets and other confidential data.

This feature specification extends the USB Mass Storage class to allow hosts and devices to lock and unlock storage, without breaking legacy software behavior that exists today and without resorting to vendor-specific descriptors and requests.

1.1 Purpose

This feature specification lets a device require that the host re-supply a passphrase before trusting the host to access the stored media.

1.2 Scope

This feature specification:

a) Fits within the class-specific extension descriptor mechanism defined in [Core USB].

b) Extends without replacing the existing Mass Storage functionality with new descriptors and requests.

c) Allows devices to divide their storage into multiple LUs, individually locked and unlocked.

d) Requires lockable devices, when plugged in locked, to declare their incompatibility with legacy Mass Storage applications. Requires lockable devices, when unlocked, to be backwards compatible with legacy Mass Storage applications.

e) Does not impose limits on the design of legacy devices that lack this lockable storage feature.

f) Allows lockable and legacy devices to inter-operate with lock-aware and legacy hosts without regard for who made either.

g) Leaves device vendors free to compete over value add features. Also leaves host vendors free to compete over value add features.

h) Requires lockable devices to declare that they support [SCSI] requests, USB class-specific requests, or both for locking and unlocking the media.

1.3 Related Documents

[ATA] AT Attachment with Packet Interface Extension (ATA/ATAPI-7) ANSI INCITS.397-2005 <T13.org>

[BBB] "USB Mass Storage Class Bulk-Only Transport", Revision 1.0, September, 1999 <Usb.org>

[CCS] "Common Class Specification", Revision 1.0, December, 1997; especially "3.11" "Identifying Class and Vendor-Specific Requests and Descriptors" <Usb.org>

[Core USB] "USB Specification", Revision 2.0, April, 2000 (with Errata thru February, 2005); especially "Chapter 9" "USB Device Framework" <Usb.org>

[MSC] "USB Mass Storage Class Overview", Revision 1.2, June, 2003 <Usb.org>

[SAT] SCSI ATA Translation (ATA thru SCSI) T10/1711-D <T10.org>

[SCSI] "Small Computer System Interface", Revision 1, December, 1985 <T10.org>

[Unicode FAQ] "Unicode FAQ", August, 2004 <Unicode.org>

[UTF-8] "UTF-8, a transformation format of ISO 10646", "RFC 3629", November, 2003 <Rfc-editor.org>

1.4 Terms and State Diagram Conventions

1.4.1 Terms

Checked *Checked In* and *Checked Out*

Checked In Device *shall* avoid, host *shall* reject if found, today (tomorrow *may* be different)

Checked Out Host *shall* avoid, device *shall* reject if found, today (tomorrow *may* be different)

Compatible ID The { bInterfaceClass : bInterfaceSubClass : bInterfaceProtocol } tuple

Configuration Bundle The list of descriptors fetched by a Get Descriptor, Configuration request

Device The USB peripheral that stores and fetches and erases bits for the host

Descriptor Bundle The collection of all the standard descriptors of a device, including those fetched by the Device, Configuration, and String variations of the standard Get Descriptor request

Hardware ID The { idVendor : idProduct : bcdDevice : SerialNumber } tuple

Host The client that asks the device to store and fetch and erase bits

Impersonal Media accessible because no passphrase stored

In From storage device to host

Out To storage device from host

Legacy Software Software which does not support the features of this specification

Locked Some or all media is inaccessible

LU Logical Unit. Note, a device *may* contain one or more Logical Units

May A keyword that indicates flexibility of choice with no implied preference

Negotiable IDs The { 08h : 07h : 50h } tuple from [MSC] that is the Compatible ID of a Mass Storage Interface that implements the Lockable [SCSI] subclass with the [BBB] protocol

Power VBus or device power

Reserved *Reserved In* and *Reserved Out*

Reserved In Device *shall* avoid, host *shall* ignore, today (tomorrow *may* be different)

Reserved Out Host *shall* avoid, device *shall* ignore, today (tomorrow *may* be different)

SCSI BBB IDs The { 08h : 06h : 50h } tuple from [MSC] that is the Compatible ID of a Mass Storage Interface that implements the [SCSI] subclass with the [BBB] protocol

Shall A keyword indicating a mandatory requirement

Should A keyword indicating a flexibility of choice, with a strongly preferred alternative, equivalent to the phrase “it is strongly recommended” by the committee that published this specification

Unlocked All media is accessible

USB-IF USB Implementer’s Forum, Inc. <Usb.org>

1.4.2 State Diagram Conventions

Figure 1 - State & Transition Notation

By convention …

Every state gets two labels: first a short State Designator and then also a more meaningful State Name. The same label never appears twice: every state designator or state name refers to exactly one state.

Every state transition gets two labels: first a short Transition Label and then also a more meaningful Transition Condition. The Transition Label is merely a pair of State Designators: first the state *from* which the transition transits, then the state *to* which the transition transits. The Transition Condition explains *why* the transition occurs.

State diagrams, tables, and text ought to agree perfectly. However, if they do disagree, then state diagrams takes precedence over tables, and tables takes precedence over text.

2 Management Overview

This specification defines passphrase-locking protection for USB Mass Storage devices. Locking the data keeps the data private despite loss or theft of the device. Locking does not provide end-to-end security between the device and the host. For example, locking does not encrypt data transfers. Also, unlocking the data for one user unlocks the data for all users of the same host.

This specification requires that locked devices supply new Negotiable IDs on plug-in which enable hosts to differentiate between legacy and locked devices. In particular, the new Negotiable IDs allow legacy hosts to avoid being exposed to a locked mass storage interface which would fail Read requests. As host behavior to failed Read requests is inconsistent in the field, such failures may lead to an unpleasant user experience. This specification also allows unlocked devices to supply legacy IDs for compatibility with legacy software.

Lockable devices can independently lock one or more Logical Units within the device. A device can only present one set of Compatible IDs at a time: SCSI BBB IDs or Negotiable IDs. A device with some Logical Units locked and some unlocked presents Negotiable IDs on plug-in to declare its incompatibility with legacy Mass Storage applications.

2.1 Goals

This specification:

1. Provides automatic privacy - when the device is unpowered it locks.
2. Addresses end-user lost/theft scenarios.
3. Guards privacy without reducing data transfer rates.
4. Delivers consumer level lock and unlock without additional hardware.
5. Clean support for legacy systems:
    - Enables legacy functionality on legacy systems
    - Enables enhanced functionality on supported systems
    - Protects legacy systems from unexpected enhanced functionality
    - Provides mechanism for signaling field update of host software
6. Introduces an extensible architecture for more complex locking.

2.2 Ease of Use

The devices specified here lock the data for the user: the user doesn't have to remember to act. Instead, unplugging the device locks the data automatically, once the user has provided a passphrase for

unlocking the data. Once a locked LU is unlocked, the user can work with protected data same as the user works with undefended data: the user only has to unlock the data after plugging the device in.

This specification lets the user:

1. Store an initial passphrase and hint (4.1.3 Personalize operation).
2. Unlock with the correct passphrase (4.1.4 Unlock operation).
3. Lock until the host resupplies the passphrase (4.1.5 Lock operation).
4. Change the passphrase and hint with the correct passphrase (4.1.6 Repersonalize operation).
5. Erase the passphrase and hint with the correct passphrase (4.1.7 Depersonalize operation).
6. Recover the media (not data) by making the previously written data unretrievable, erasing the hint, and removing the passphrase, without requiring the user to remember the correct passphrase (4.1.8 Recover Media operation).

2.3 Privacy

This specification leaves device and host vendors free to compete over how strong the lock might be. When the device is lost or stolen, the easiest attack on the lock is to attempt a series of passphrases ("a dictionary attack"). A more sophisticated attack is to discover clues about the passphrase by observing time, power, electromagnetic spectrum, or other characteristics of the device as the device rejects incorrect passphrases ("a side-channel attack"). This specification includes some examples for value-added protections against such attacks.

This specification preserves the value of the device even when a user forgets the passphrase. This specification defines an operation to make the previously written data unretrievable, remove the passphrase, and recover the media. In this way, the device defends the privacy of the data against a thief, yet still also allows users who frequently forget passphrases to keep using the device.

2.4 Compatibility

This specification allows a device to present more than one set of Compatible IDs: either SCSI BBB IDs or Negotiable IDs.

An unlocked device can present SCSI BBB IDs to tell the host of its compatibility with legacy drivers and applications. A locked device can present Negotiable IDs to tell legacy software that the device is not a standard mass storage device, and that different software is needed for its additional functionality. Once the new driver unlocks the device, that driver can tell the device to plug back into the host with the SCSI BBB IDs instead, so that legacy Mass Storage drivers and applications again work without modification.

A completely new host driver that runs in parallel with legacy mass storage drivers does not need to use the SCSI BBB IDs. This new driver would replace and enhance the functionality of legacy Mass Storage class drivers for devices that present Negotiable ID's.

Assigning Negotiable IDs provides additional benefits. Legacy boot BIOS, operating systems, and applications rarely respond reasonably when surprised. The Negotiable IDs help protect legacy software from being surprised by the device protecting the data. In addition, the Negotiable IDs can trigger an OS

to update itself to support locking.

3 Functional Characteristics

[Core USB] divides each device into configurations and each configuration into interfaces. This specification divides each Lockable Interface into LUs.

3.1 Interface States

Figure 2 – Functional Interface Behavior

Section 4.1 “Operations” specifies how a host can operate a Lockable Interface of the device by defining terms of the diagram above.

Each Lockable Interface plugs into the host in any of four different ways:

1. Power up with the SCSI BBB IDs when all media of the Interface is accessible (*i.e.*, is Impersonal).

2. Power up with the Negotiable IDs when any media of the Interface is inaccessible (*i.e.*, is Locked).

3. Electrically plug back in with the SCSI BBB IDs on request, provided that all media of the Interface is accessible (*i.e.*, is Unlocked or Impersonal).

4. Electrically plug back in with the Negotiable IDs on request.

These variations in the Cycle Power (4.1.1) and Cycle Presence (4.1.2) operations let the Interface make Locked media inaccessible without ever failing a [SCSI] Read request. These variations also let a small driver make Unlocked media accessible to legacy [SCSI] drivers and applications.

To manage the work of electrically or physically unplugging and plugging back in, the Cycle Power and Cycle Presence operations move each Interface among three states:

1. AnyPersonal – media accessible for a while, after host resupplies passphrase, presents Negotiable IDs

2. CyclingPresence – media unplugging

3. AllImpersonal – media access unimpeded at plug in: legacy compatible, presents SCSI BBB IDs

Legacy devices present one Compatible ID per interface. When the device presents its descriptors, the host can copy those descriptors into a cache keyed by the Hardware ID of the device. Caching the descriptors lets the host match each interface of the device with compatible software over time, without re-interrogating the device. Lockable devices can present different Compatible IDs after a Cycle Presence operation. To help the host flush the software that matched the previous Compatible IDs and flush the descriptor cache, the Cycle Presence operation changes not only the Compatible IDs but also the Hardware ID of the device.

Some legacy hosts match devices that present SCSI BBB IDs with software that responds inconsistently to failing Read requests. When some of the media of the Interface is inaccessible, the device presents Negotiable IDs to avoid exposing legacy hosts to inaccessible media. When all of the media of the Interface is accessible, the host can ask the device to present either SCSI BBB IDs or Negotiable IDs. SCSI BBB IDs match with legacy software that might depend on unimpeded media access. Negotiable IDs match with software that negotiates media access.

3.2 LU States

Figure 3 - Functional LU Behavior

Section 4.1 “Operations” specifies how a host can operate the LUs within a Lockable Interface by defining the terms of the diagram above.

Each LU of the Interface adds six operations to the Mass Storage legacy:

1. Store an initial passphrase and hint (4.1.3 Personalize operation).

2. Unlock with the correct passphrase (4.1.4 Unlock operation).

3. Lock until the host resupplies the passphrase (4.1.5 Lock operation).

4. Change the passphrase and hint with the correct passphrase (4.1.6 Repersonalize operation).

5. Erase the passphrase and hint with the correct passphrase (4.1.7 Depersonalize operation).

6. Recover the media (not data) by making the previously written data unretrievable, erasing the hint, and removing the passphrase, without requiring the user to remember the correct passphrase (4.1.8 Recover Media operation).

These operations move each LU among three states:

1. Locked – media inaccessible until host resupplies passphrase

2. Unlocked – media accessible for a while, after host resupplies passphrase

3. Impersonal – media accessible always: legacy compatible, no passphrase stored

4 Operational Model

The operational model of the device consists of Interface operations divided into transitions between Interface states and LU operations divided into transitions between LU states. The transitions between the states occur when the device completes a class-specific request and when power cycles. Operations of an Interface affect every LU of that Interface.

4.1 Operations

To operate a Lockable Interface and its LUs, the host can combine the requests of this specification as follows.

4.1.1 Cycle Power

VBus and Device Power cycle together or independently. And whenever either does cycle, then every Interface and LU starts again. In particular, every LU with a passphrase relocks itself when power cycles, and every Interface that contains an LU with a passphrase presents Negotiable IDs (4.2.1 IF1).

The descriptors of a lockable device differ slightly from the Mass Storage legacy. In the specified place, the descriptor bundle includes a Lockable Storage Interface Extension Descriptor (5.4 LSIED). This carefully placed modification is small enough for legacy software to ignore, while large enough for newly standard software to notice. New software discovers that it can personalize (4.1.3) a compatible Impersonal LU by noticing that an LSIED elaborates the enclosing Interface.

The Cycle Power operation succeeds by:

a) Starting every LU with a passphrase in the Locked state (4.3.1 LU1).

b) Starting every LU without a passphrase in the Impersonal state (4.3.3 LU3).

c) Presenting Negotiable IDs (IF1) if any LU starts in the Locked state (LU1).

4.1.2 Cycle Presence

The host can ask the device to electrically unplug itself and then plug itself back in again, with the Change Interface Access Out request (6.2.7 CIAO).

NOTE: Unplugging the device tells the host to tear down the kernel software driver stack of each Interface and LU. Plugging the device back in again with changed Compatible IDs in the Interface Descriptor (5.3 ID) and corresponding changed Hardware IDs in the Device Descriptor (5.2 DD) tells the host to build a changed kernel software driver stack that matches the changed capabilities of the device.

The CIAO request affects the entire device: all its Configurations, Interfaces, LUs, etc.

The CIAO request copies Access Data (5.7 AD) out to let the host say which Compatible IDs it wants the device to present for the addressed Interface: either the Negotiable IDs (4.2.1 IF1) or SCSI BBB IDs (4.2.3 IF3). A CIAO-P request is a CIAO request that asks for the Negotiable IDs. A CIAO-I request is a CIAO request that asks for SCSI BBB IDs.

The Cycle Presence operation succeeds by completing each of:

a) Leaving the device plugged in long enough for the host to see the interface accept the request before the device unplugs (5.7 AD dwIdleMs).

b) Unplugging the device long enough for the host to see the device unplug before the device plugs back in (5.7 AD dwGoneMs).

c) Plugging the device back in again.

d) Presenting the requested IDs for the addressed interface (IF1 or IF3).

4.1.3 Personalize

The host can tell an Impersonal LU (4.3.3 LU3) to lock when power next cycles and also then to remain locked until a host resupplies a correct passphrase, via the Store Passphrase Out request (6.2.1 SPO). With that same request, the host can also ask the LU to store a hint to help the user remember the passphrase.

The SPO request supplies the passphrase and the hint to store. Thereafter, the host can ask to read the stored hint back (6.1.1 GLI), but the host cannot read the stored passphrase. The host can only ask to compare the stored passphrase with candidate passphrases (6.2.2 MPO).

The Personalize operation succeeds by completing each of:

a) Storing a passphrase and a hint (either or both of which can be empty).

b) Moving the LU to the Unlocked state (4.3.2 LU2).

4.1.4 Unlock

The host can tell a Locked LU (4.3.1 LU1) to make the media accessible without erasing the passphrase, via the Match Passphrase Out request (6.2.2 MPO).

The MPO request supplies a candidate passphrase to compare with the stored passphrase.

The Unlock operation succeeds by completing each of:

a) Supplying a correct passphrase (which can be empty).

b) Moving the LU to the Unlocked state (4.3.2 LU2).

The Negotiable IDs allow the [SCSI] conversation to begin immediately after the LU reaches the Unlocked state. Or the host can allow legacy software to access the media by plugging in SCSI BBB IDs with a Cycle Presence operation (4.1.1).

4.1.5 Lock

The host can tell an Unlocked LU (4.3.2 LU2) to make the media inaccessible without unplugging the device, via the Lock Again request (6.2.6 LA).

The Lock operation succeeds by completing each of:

a) Addressing an LU of an AnyPersonal Interface

b) Moving that LU to the Locked state (4.3.1 LU1).

4.1.6 Repersonalize

The host can tell an Unlocked LU (4.3.2 LU2) to change the stored passphrase and/or hint, via the Change Passphrase Out request (6.2.3 CPO).

The CPO request supplies a candidate passphrase to match with the stored passphrase, and also a new passphrase and new hint.

The Repersonalize operation succeeds by completing each of:

a) Supplying a correct passphrase (which can be empty).

b) Storing a new passphrase and a new hint (either or both of which can be empty).

c) Leaving the LU in the Unlocked state.

4.1.7 Depersonalize

The host can tell an Unlocked LU (4.3.2 LU2) to forget the stored passphrase and hint, via the Erase Passphrase Out request (6.2.4 EPO).

The EPO request supplies a candidate passphrase to match with the stored passphrase.

The Depersonalize operation succeeds by completing each of:

a) Supplying a correct passphrase (which can be empty).

b) Erasing the stored passphrase and hint.

c) Moving the LU to the Impersonal state (4.3.3 LU3).

4.1.8 Recover Media

The host can tell a Locked LU (4.3.1 LU1) to recover the media (not data) by making the previously written data unretrievable and then erasing the forgotten passphrase, via the Erase Forgotten Passphrase request (6.2.5 EFP), without requiring the user to remember the passphrase.

Before sending this request, the host can fetch the guess of how much time this Recover Media operation requires (5.8 LD dwCompletingMs) with a Query operation (4.1.9).

The Recover Media operation succeeds by completing each of:

a) Making the previously written data unretrievable,

b) Erasing the forgotten passphrase and the old hint.

c) Moving the LU to the Impersonal state (4.3.3 LU3).

The Negotiable IDs allow the [SCSI] conversation to begin immediately after the LU reaches the Impersonal state. Or the host can allow legacy software to access the media by plugging in SCSI BBB IDs with a Cycle Presence operation (4.1.2).

4.1.9 Query

The host can examine the Interface and the LU in any observable state of the Interface. The host can examine the Interface via the standard requests of [Core USB]. The host can examine the LU via the Get Lock In request (6.1.1 GLI) of this specification.

The GLI request does not itself cause the LU to move from one state to another, but might fetch different contents if tried and retried. First, the other requests and operations of this specification flip bits of the LU as specified, especially the bits of the LockHint, bLuState, dwSteppingMs, and dwCompletingMs fields. Second, other events such as power cycles have their specified effects.

Unobservable states of the Interface include the device having been physically removed and the device having been unplugged by a Cycle operation (4.1.1 Cycle Power, 4.1.2 Cycle Presence).

Examining the LU answers such questions as:

a) How much time remains in the effort to complete the last Lockable Storage request?

b) Which LUs and Interfaces of the device are Locked or Unlocked or Impersonal?

c) What was the hint stored with the passphrase?

d) How large might the passphrases and hints of the LU be?

The Query operation succeeds by completing each of:

a) Fetching the Lock Data (5.8 LD) that describes the LU.

b) Including the dwSteppingMs guess of when the LD might change again.

c) Including the dwCompletingMs guess of when the LU state might stop changing.

4.2 Interface State Diagram

Figure 4 - Lockable Interface States

4.2.1 AnyPersonal

Cycling power starts the Interface in this state or in the AllImpersonal state (4.2.3 IF3).

In this IF1:AnyPersonal state, the Interface:

a) *shall* present the Negotiable IDs (5.2 DD, 5.3 ID).

b) *shall* let the host access Unlocked and Impersonal LUs.

c) *may* contain Locked LUs (4.3.1 LU1).

d) *may* limit access to Locked LUs.

e) *shall* reject any CIAO-I request received while this Interface contains Locked LUs (LU1).

f) *may* start again in this state or in the AllImpersonal state (IF3) after a power cycle.

In this state, the host can try the Cycle Power (4.1.1) and Cycle Presence (4.1.2) operations of the Interface. However, the Interface rejects CIAO-I operations (4.1.2) while the Interface contains Locked LUs.

Entry Transitions IF1 The Interface *shall* enter this IF1:AnyPersonal state

if the device completes a CIAO-P operation (4.1.2) addressed to this Interface, or

if VBus or device power cycles while this Interface contains Locked or Unlocked LUs (LU1 or 4.3.2 LU2), or

if the device has not assigned a Hardware ID to one or more descriptor bundles that describe the Interface with SCSI BBB IDs.

Transition IF1:IF2 The Interface *shall* enter the IF2:CyclingPresence state after accepting a Change Interface Access Out request (6.2.7 CIAO).

An Interface in the IF1:AnyPersonal state *shall* remain in the IF1:AnyPersonal state even if the host sends a [BBB] [SCSI] Reset (BOMSR), USB Suspend, USB Resume, or USB Reset.

4.2.2 CyclingPresence

Cycling power never starts the Interface in this state.

In this IF2:CyclingPresence state, the Interface:

a) *shall* be unobservable, presenting no IDs.

b) *shall* defer all requests received while preparing to unplug (*i.e.*, Nak and Ack during the dwIdleMs time as specified by 6.2.7 CIAO).

c) *shall* start again in the AnyPersonal state (IF1) or in the AllImpersonal state (IF3) after a power cycle.

Entry Transitions IF2: The Interface *shall* enter this IF2: Cycle Presence state

after the device accepts a CIAO operation (4.1.2) addressed to this Interface.

Transition IF2:IF1 The Interface *shall* go to the IF1:AnyPersonal state

after the device completes a CIAO-P operation (4.1.2) addressed to this Interface.

Transition IF2:IF3 The Interface *shall* go to the IF1:Legacy state

after the device completes a CIAO-I operation (4.1.2) addressed to this Interface.

An Interface in the IF2:CyclingPresence state *may* continue, complete, or rollback the CIAO operation (4.1.2) in progress if the host sends a [BBB] [SCSI] Reset (BOMSR), USB Suspend, USB Resume, or USB Reset.

4.2.3 AllImpersonal

Cycling power starts the Interface in this state or in the AnyPersonal state (4.2.1 IF1).

In this IF3:AllImpersonal state, the Interface:

a) *shall* present the SCSI BBB IDs (5.2 DD, 5.3 ID).

b) *shall* let the host access Unlocked and Impersonal LUs.

c) *may* limit access to Locked LUs.

d) *should not* contain Locked LUs (4.3.1 LU1), since those do not offer media access.

e) *should* intercept and reject any Lock Again (6.2.6 LA) request addressed to an LU of this Interface.

f) *may* start again in this state or in the AnyPersonal state (IF1) after a power cycle.

In this state, the host can try the Cycle Power (4.1.1) and Cycle Presence (4.1.2) operations of the Interface.

Entry Transitions IF3 The Interface *shall* enter this IF3:AllImpersonal state

if the device completes a CIAO-I operation (4.1.2) addressed to this Interface, or

if VBus or device power cycles while this Interface contains only Impersonal LUs (4.3.3 LU3) – provided that the device has assigned a Hardware ID to one or more descriptor bundles that describe the Interface with SCSI BBB IDs.

Transition IF3:IF2 The Interface *shall* enter the IF2:CyclingPresence state

after accepting a Change Interface Access Out request (6.2.7 CIAO).

An Interface in the IF3:AllImpersonal state *shall* remain in the IF3:AllImpersonal state even if the host sends a [BBB] [SCSI] Reset (BOMSR), USB Suspend, USB Resume, or USB Reset.

4.3 LU State Diagram

Figure 5 - Lockable LU States

4.3.1 Locked

Cycling power starts the LU in this state or in the Impersonal state (4.3.3 LU3).

In this LU1:Locked state, the LU:

a) is observable.

b) connects thru an Interface presenting Negotiable IDs (4.2.1 IF1).

c) describes itself with a bLuState code of Locked (5.8 LD).

d) does not offer media access to the host.

e) *shall* start again in this state if power cycles.

In this state, the host can try the Unlock (4.1.4), Recover Media (4.1.8), and Query (4.1.9) operations of the LU.

Before asking the user to provide a passphrase to unlock the LU, the host can try a Query operation to fetch the hint that was stored with the passphrase. The stored hint might have been empty.

Entry Transitions LU1 The LU *shall* enter this LU1:Locked state

after successful completion of a Lock Again request (6.2.6 LA) in the LU2:Unlocked state, or

if VBus or device power cycles while the LU is in this state or the Unlocked state (4.3.2 LU2).

Transition LU1:LU1 The LU *shall* remain in this LU1 state

while completing a Get Lock In request (6.1.1 GLI).

Transition LU1:LU2 The LU *shall* go to the LU2:Unlocked state

after successful completion of a Match Passphrase Out request (6.2.2 MPO) with a correct candidate phrase.

Transition LU1:LU3 The LU *shall* go to the LU3:Impersonal state

after successful completion of an Erase Forgotten Passphrase request (6.2.5 EFP).

An LU in the LU1:Locked state *shall* remain in the LU1:Locked state even if the host sends a [BBB] [SCSI] Reset (BOMSR), USB Suspend, USB Resume, or USB Reset.

4.3.2 Unlocked

Cycling power never starts the LU in this state.

In this LU2:Unlocked state, the LU:

a) is observable

b) connects thru an Interface presenting Negotiable IDs (4.2.1 IF1) or SCSI BBB IDs (4.2.3 IF3).

c) describes itself with a bLuState code of Unlocked (5.8 LD).

d) does offer media access to the host.

e) *shall* start again in the LU1:Locked state if power cycles.

In this state, the host can try the Lock (4.1.5), Repersonalize (4.1.5), Depersonalize (4.1.7), operations, or Query (4.1.9) operations. However, a AllImpersonal Interface *should* reject Lock operations.

Before asking the user to provide a passphrase to Repersonalize or Depersonalize the LU, the host can request a Query operation to fetch the hint that was stored with the passphrase. The stored hint might have been empty.

Entry Transitions LU2 The LU *shall* enter this LU2:Unlocked state

after successful completion of a Match Passphrase Out request (6.2.2 MPO) with a correct candidate phrase in the LU1:Locked state, or

after successful completion of a Store Passphrase Out request (6.2.1 SPO) in the LU3:Impersonal state.

Transition LU2:LU1 The LU *shall* go to the LU1:Locked state

after successful completion of a Lock Again request (6.2.6 LA).

Transition LU2:LU2 The LU *shall* remain in this LU2 state

while completing a Get Lock In request (6.1.1GLI).

Transition LU2:LU2 The LU *shall* remain in this LU2 state

while completing a Change Passphrase Out request (6.2.3 CPO).

Transition LU2:LU3 The LU *shall* go to the LU4:Impersonal state

after successful completion of an Erase Passphrase Out request (6.2.2 EPO) with a correct candidate phrase.

An LU in the LU2:Unlocked state *shall* remain in the LU2:Unlocked state even if the host sends a [BBB] [SCSI] Reset (BOMSR), USB Suspend, USB Resume, or USB Reset.

4.3.3 Impersonal

Cycling power starts the LU in this state or in the Locked state (4.3.1 LU1).

In this LU3:Impersonal state, the LU:

a) is observable

b) connects thru an Interface presenting Negotiable IDs (4.2.1 IF1) or SCSI BBB IDs (4.2.3 IF3).

c) describes itself with a bLuState code of Impersonal (5.8 LD).

d) does offer media access to the host.

e) *shall* start again in this state if power cycles.

In this state, the host can try the Personalize (4.1.1) or Query (4.1.9) operations.

Entry Transitions LU3 The LU *shall* enter this LU3:Impersonal state

after successful completion of an Erase Forgotten Passphrase request (6.2.5 EFP) in the LU1:Locked state, or

after successful completion of an Erase Passphrase Out request (6.2.2 EPO) in the LU3:Unlocked state, or

if VBus or device power cycles while the LU is in this state.

Transition LU3:LU2 The LU *shall* go to the LU2:Unlocked state

after successful completion of a Store Passphrase Out request (6.2.1 **SPO**).

Transition LU3:LU3 The LU *shall* remain in this LU3 state

while completing a Get Lock In request (6.1.1 **GLI**).

An LU in the LU3:Impersonal state *shall* remain in the LU3:Impersonal state even if the host sends a [BBB] [SCSI] Reset (BOMSR), USB Suspend, USB Resume, or USB Reset.

5 Data Structures

This chapter discusses descriptors that do not change while the device remains plugged in and defines some class-specific data structures that change more frequently. Appendix A assigns the codes that distinguish these descriptors and data structures from one another. The following table names all the descriptors and other data structures discussed or defined here.

Table 1 - Descriptors

bmDescriptorType code	Section	Description
n/a	5.1	USB Descriptor Template (UDB)
DEVICE	5.2	Device Descriptor (DD)
INTERFACE	5.3	Interface Descriptor (ID)
LOCKABLE	5.4	Lockable Storage Interface Extension Descriptor (LSIED)
PHRASE	5.5	Phrase Data (PD)
HINT	5.6	Hint Data (HD)
ACCESS	5.7	Access Data (AD)
QUERIED	5.8	Lock Data (LD)

5.1 USB Descriptor Template (UDB)

[CCS] specifies a common structure for all [Core USB] descriptors. The class-specific data structures of this specification share that structure also. That structure is:

Table 2 - USB Descriptor

Offset	Field Name	Byte Size	Value	Description
00h	**bLength**	1	2 <= n	Byte length of this structure
01h	**bmDescriptorType**	1	Bitmap	Characteristics of this descriptor: D7: *Reserved* D6..5: Type 3 = *Checked* 2 = Vendor 1 = USB-IF Class 0 = [Core USB] Standard D4..0: Descriptor Number 0-1Fh (0-31)
02h	**Opaque**	n – 2	Arbitrary	Payload, *i.e.*, opaque content of a variable byte length
n				

A standard Get Descriptor request for a configuration descriptor copies in the Configuration Bundle of descriptors. Each descriptor of that bundle, including the Configuration descriptor, has the structure of a USB Descriptor. The position of each descriptor in the bundle also has a standard meaning. The standard Configuration Descriptor appears first in the bundle, and then zero or more standard Interface and Endpoint Descriptors follow. Each class-specific or vendor-specific descriptor elaborates the last standard descriptor that precedes it within the bundle.

The bmDescriptorType code distinguishes each of the descriptors within the configuration bundle from one another. The USB-IF reserves bmDescriptorType codes 20-3Fh (32-63) for class-specific assignment and gives codes 40-5Fh (64-95) to the device vendor for vendor-specific assignment among all devices that share an idVendor code.

5.2 Device Descriptor (DD)

The *Hardware IDs* of a device are its { idVendor : idProduct : bcdDevice : SerialNumber }.

[Core USB] specifies the form of a Device Descriptor. Lockable Devices *shall* implement Device Descriptors as below:

Table 3 - Device Descriptor

Offset	Field Name	Byte Size	Value	Description
00h	**bLength**	1	12h	Byte length of this structure.
01h	**bmDescriptorType**	1	01h	Code DEVICE.
02h	**bcdUSB**	1	BCD	0200h, 0110h, 0100h, etc.
03h	**bDeviceClass : bDeviceSubClass : bDeviceProtocol**	3	ID:ID:ID	*See [Core USB]*
06h	**bMaxPacketSize0**	1	Number	Max packet size for endpoint zero.
08h	**idVendor**	2	ID	Vendor ID code.
0Ah	**idProduct**	2	ID	Product ID code: the variable portion of the Hardware ID.
0Ch	**bcdDevice**	2	BCD ID	Device release number.
0Eh	**iManufacturer**	1	Index	Zero, else index of the string descriptor that names the manufacturer.
0Fh	**iProduct**	1	Index	Zero, else the index of the string descriptor that names the product.
10h	**iSerialNumber**	1	Index of ID	The nonzero index of the USB SerialNumber ID string.
11h	**bNumConfigurations**	1	Number	Number of configuration bundles.

The device *shall* present one distinct Hardware ID for each distinct descriptor bundle. To distinguish one Hardware ID from another, the device *may* vary the idProduct ID code, the bcdDevice ID code, and/or the SerialNumber ID string. The device *should* vary only the idProduct ID code.

5.3 Interface Descriptor (ID)

The *Compatible IDs* of an Interface are its { bInterfaceClass : bInterfaceSubClass : bInterfaceProtocol }.

[Core USB] specifies the form of an Interface Descriptor. Lockable Devices *shall* implement Interface Descriptors as below.

Table 4 - Interface Descriptor

Offset	Field Name	Byte Size	Value	Description
00h	**bLength**	1	09	Byte length of this structure.
01h	**bmDescriptorType**	1	04h	Code INTERFACE.
02h	**bInterfaceNumber**	1	Number	Number of this interface.
03h	**bAlternateSetting**	1	Number	Value used to select this alt. setting.
04h	**bNumEndpoints**	1	Number	Number of endpoints used by this interface (excluding endpoint zero).
05h	**bInterfaceClass**	1	08h	Mass Storage.
06h	**bInterfaceSubClass**	1	07h or 06h	SubClass ID. 06h if bInterfaceProtocol is 50h, else 07h.
07h	**bInterfaceProtocol**	1	50h	Protocol ID code. 50h if bInterfaceSubClass is 06h.
08h	**iInterface**	1	Index	Zero, else index of the string descriptor that names this interface.

The device *shall* present Negotiable IDs for any Interface in the IF1:AnyPersonal state (4.2.1 IF1).

The device *shall* present SCSI BBB IDs for any Interface in the IF3:AllImpersonal state (4.2.3 IF3).

5.4 Lockable Storage Interface Extension Descriptor (LSIED)

The device distinguishes a Lockable Interface from legacy Mass Storage interfaces by *elaborating* the Mass Storage descriptor of the Lockable Interface with a class-specific Interface Extension Descriptor (IED), specifically a *Lockable Storage Interface Extension Descriptor (LSIED)*.

The structure of an LSIED is:

Table 5 - Lockable Storage Interface Extension Descriptor (LSIED)

Offset	Field Name	Byte Size	Value	Description
00h	**bLength**	1	03h	Byte length of this structure.
01h	**bmDescriptorType**	1	LOCKABLE	Distinguishes this descriptor from other descriptors of the configuration bundle.
02h	**bVariation**	1	Enumeration	00h = C_LOCKABLE, or 01h = B_LOCKABLE. 02h-FFh = *Reserved In*
03h				

For each Mass Storage interface that implements this specification, the device *shall* insert an LSIED after the corresponding Standard Interface descriptor, before the Standard Endpoint descriptor that follows it. The device *shall* include the length of each LSIED in the wTotalLength of the Configuration Descriptor that begins the configuration bundle of descriptors that contains the Interface Descriptor. The device *shall* insert the LSIED after the Interface Descriptor both when presenting SCSI BBB IDs and when presenting Negotiable IDs.

This specification defines only bLength = 03h together with bmDescriptorType equal to LOCKABLE. Other specifications give meaning to the other possible values of those fields.

A device that presents bVariation = C_LOCKABLE *shall* implement all this specification as written.

A device that presents bVariation = B_LOCKABLE *shall* substitute [SAT] requests for the GLI, SPO, MPO, CPO, EPO, and EFP requests defined here, as chapter 6 “Requests” specifies.

The device *may* elaborate the Interface with more than one IED, as [CCS] details. A device that presents C_LOCKABLE in one LSIED and B_LOCKABLE in another *shall* implement all this specification as written as well as process [SAT] requests as specified in chapter 6 “Requests”.

5.5 Phrase Data (PD)

The host personalizes an LU by storing *Phrase Data (PD)* in the LU. Thereafter, the host unlocks the LU by re-supplying that same PD. The structure of PD is:

Table 6 - Phrase Data (PD)

Offset	Field Name	Byte Size	Value	Description
00h	**bLength**	1	3 <= n	Byte length of this structure.
01h	**bmDescriptorType**	1	PHRASE	Distinguishes this structure from others bundled with it, if any.
02h	**Bytes**	n – 3	Arbitrary	The bytes to store now and match later.
n -1	**bEnd**	1	00h	Zero (*Checked*).
n				

For PDs out, the LU *shall* accept bLength values between 3 and 3+bMaxPhrase, inclusive. The LU *shall* reject other bLength values. Similarly, the LU *shall* accept bmDescriptorType = PHRASE and the LU *shall* reject other bmDescriptorType codes.

NOTE: The passphrase resupplied by a second host will exactly match the passphrase stored by the first host when both choose the same algorithm to convert chars to bytes.

NOTE: The Unicode algorithm for converting to bytes from chars which is binary-compatible with the US ASCII legacy in passphrases is “possibly-zero shortest-form [UTF-8] without byte-order mark”, such as an FTP passphrase. For example, this algorithm provides exactly one encoding for such varied characters as the 00h US ASCII NUL, 24h US Dollar, the A3h British Pound Sterling, and the 20ACh Euro (specifically the byte strings h 00, h 24, h C2 A3, and h E2 82 AC respectively). This algorithm provides exactly one encoding for all Unicode chars.

The host *may* encode the chars of a phrase arbitrarily, so the LU *shall* store and match whatever bytes the host provides.

The PD is *empty* when its bLength is 3.

5.6 Hint Data (HD)

The host can store *Hint Data (HD)* in the LU to help people remember the passphrase. The structure of HD is:

Table 7 - Hint Data (HD)

Offset	Field Name	Byte Size	Value	Description
00h	**bLength**	1	3 <= n	Byte length of this structure.
01h	**bmDescriptorType**	1	HINT	Distinguishes this structure from others bundled with it, if any.
02h	**Bytes**	n – 3	Arbitrary	The bytes to store now and display later.
n -1	**bEnd**	1	00h	Zero (*Reserved*).
n				

For HDs out, the LU *shall* accept bLength values between 3 and 3+bMaxHint, inclusive. The LU *shall* reject other bLength values. Similarly, the LU *shall* accept bmDescriptorType = HINT and the LU *shall* reject other bmDescriptorType values. Symmetrically, for HDs in, the LU *shall* provide bLength's between 3 and 3+bMaxHint, inclusive.

NOTE: The hint displayed by a second host will accurately represent the hint stored by the first host when the byte to chars algorithm of the second host adequately reverses the chars to byte algorithm of the first host.

NOTE: The Unicode algorithm for converting between chars and bytes which is binary-compatible with the US ASCII legacy in hints is "possibly-zero shortest-form [UTF-8] without byte-order mark", The host *may* encode the chars of a hint arbitrarily, so the LU *shall* store and fetch whatever bytes the host provides.

The HD is *empty* when its bLength is 3.

5.7 Access Data (AD)

The host tells the device when to electrically unplug itself and when to electrically plug back in by copying out *Access Data (AD)*. The structure of AD is:

Table 8 - Access Data (AD)

Offset	Field Name	Byte Size	Value	Description
00h	**bLength**	1	0Ch	Byte length of this structure. 0Ch, else *Checked Out*.
01h	**bmDescriptorType**	1	ACCESS	Distinguishes this structure from others bundled with it, if any.
02h	**targetSubClass**	1	ID	The requested bInterfaceSubClass code.
03h	**targetProtocol**	1	ID	The requested bInterfaceProtocol code.
04h	**dwIdleMs**	4	Milliseconds	Milliseconds to idle before unplugging
08h	**dwGoneMs**	4	Milliseconds	Milliseconds to remain unplugged before plugging back in.
0Ch				

For ADs out, the device *shall* accept bLength of 0Ch and bmDescriptorType = ACCESS. The device *shall* reject other bLength values and other bmDescriptorType values.

5.8 Lock Data (LD)

The host can examine an LU by fetching *Lock Data (LD)* as needed: at least once per Put request. The structure of LD is:

Table 9 - Lock Data (LD)

Offset	Field Name	Byte Size	Value	Description
00h	**bLength**	1	10h + n	Byte length of this structure. Equal to 10h + n, else *Checked In*.
01h	**bmDescriptorType**	1	QUERIED	Distinguishes this structure from others bundled with it, if any.
02h	**bMaxPhrase**	1	32h (50)	The max byte length of a passphrase.
03h	**bMaxHint**	1	64h (100)	The max byte length of a hint.
04h	**dwSteppingMs**	4	Milliseconds	Zero if LD decided, else a guess of time remaining.
08h	**bLuState**	1	Enumeration	00h = *Reserved*, or 01h = Impersonal, or 02h = Locked, or 03h = Unlocked, or 04h-FFh = *Reserved*.
09h	**bInterfaceNumber**	1	0-255 (0-FFh)	A copy of the bInterfaceNumber in the lower byte of wIndex parameter of the request that fetched this LD.
0Ah	**bLUN**	1	0-255 (0-FFh)	A copy of the bLUN of the Get Lock In (6.1.1 GLI) request that fetched this LD.
0Bh	**bPutAccepted**	1	Enumeration	00h = Last Put request rejected, or 01h = Last Put request accepted.
0Ch	**dwCompletingMs**	4	Milliseconds	A guess of the time needed to decide the LD, else a guess of the time needed to complete a Media Recovery without a passphrase.
10h	**LockHint**	3 <= n	HD (5.6)	Copy of the HD most recently stored in the LU.
10h+n				

The value of bMaxPhrase decides how much data out the device has to parse to decide to accept or reject a request. The LU needs 108 = (2*(3+bMaxPhrase)+2) bytes of the data to decide to accept or reject the Change Passphrase Out request (6.2.3 CPO).

The device *shall revise* the LD after each Cycle Power operation (4.1.1), Cycle Presence operation (4.1.2), or *Put to LU request* (SPO, MPO, CPO, EPO, EFP, and LA).

The meaning of the fields in the LD changes with dwSteppingMs zero or nonzero. Therefore, t\he device *sjhall* provide the host only fully updated LDs,

Each time the device updates the LD to show work in progress, the device *shall*:

a) Set dwSteppingMs to a nonzero guess of when the LD might change again.

b) Set dwCompletingMs to a nonzero guess of when the LD might stop changing.

c) Zero bPutAccepted.

d) Empty the LockHint.

To update the LD to show that work is complete, the device *shall*:

a) Set the bLuState and the bPutAccepted byte to show the completed status of the work.

b) Set the LockHint.to a copy of the HD most recently stored in the LU.

c) Set dwCompletingMs to a nonzero guess of the time needed for a subsequent Erase Forgotten Passphrase (EFP) request to complete from the L1:Locked state.

d) Zero dwSteppingMs.

See Chapter 6 “Requests” for more detail on how the dwSteppingMs mechanism lets Put to LU requests complete slowly.

The device *should* guess dwSteppingMs and dwCompletingMs accurately. The device *may* increase or reduce dwSteppingMs and/or dwCompletingMs, from one sample to the next. The device *should not* report a dwSteppingMs larger than dwCompletingMs.

The device *shall* return an empty hint of three bytes in the LockHint field, rather than the last hint stored if:

a) The bLuState is 01h Impersonal, or

b) The dwSteppingMs guesstimate is nonzero, or

c) The read of the hint fails.

The host *should* begin by sending the Get Max LUN (GML) request of [BBB] to discover how many LUs exist within the interface. Always one LU exists at bLUN address zero. The device *may* respond indeterminately if the host addresses a bLUN beyond bMaxLUN.

While dwSteppingMs is nonzero, the host *should* disregard the bLuState, the bPutAccepted byte, and the LockHint. Also, to avoid wasting bandwidth on polling, the host *should* wait for dwSteppingMs milliseconds to elapse before requesting another copy of the LD.

6 Requests

This specification defines a number of class-specific requests. Appendix A assigns the bRequest and wValue codes that distinguish these requests from other requests and from one another. The following table names all the class-specific requests defined here.

Table 10 - Class Specific Requests

B_LOCKABLE	C_LOCKABLE	Description
[SAT] Identify	6.1.1	Get Lock In (GLI)
[SAT] Set Password	6.2.1	Store Passphrase Out (SPO)
[SAT] Unlock	6.2.2	Match Passphrase Out (MPO)
[SAT] Set Password	6.2.3	Change Passphrase Out (CPO)
[SAT] Disable Password	6.2.4	Erase Passphrase Out (EPO)
[SAT] Security Erase	6.2.5	Erase Forgotten Passphrase (EFP)
6.2.6	6.2.6	Lock Again (LA)
6.2.7	6.2.7	Change Interface Access Out (CIAO)

This chapter defines one class-specific request for the Interface and a number of requests for the LUs within the Interface.

The device *accepts* or *rejects* each request from the host. For Get requests and CIAO requests, the device *shall* Nak to delay answering, Stall to reject, and Ack to accept. But for the *Put to LU requests* (SPO, MPO, CPO, EPO, EFP, and LA):

a) To delay the decision to accept or reject a Put to LU request, the device *may* Nak the control pipe shared among all Interfaces and LUs, but the device *should* instead set dwSteppingMs nonzero and Ack the request.

b) While dwSteppingMs remains nonzero, the host *should not* send and the device *shall* Stall any Put to LU request. The device *shall* still process any other requests as fast as their specifications require.

c) To reject the Put to LU request, the device *may* Stall the request, but the device *should* instead set dwRemaingMs nonzero, Ack the request, and zero bPutAccepted before zeroing dwSteppingMs.

d) To accept the Put to LU request, the device *may* set dwSteppingMs nonzero, Ack the request, and update set bPutAccepted (etc., including bLuState) before zeroing dwSteppingMs; although the device *should* decide quickly, zero dwSteppingMs, and Ack the request.

6.1 Get Requests

The Get requests let the host ask to copy data in. The following table provides the template for Get requests:

Table 11 - Get Request Template

Request

bmRequestType	bRequest	wValue	wIndex	wLength
1010 0001 b	GET	Varies	*Reserved Out* : bInterfaceNumber	Max Data In

Data In

Offset	Field Name	Byte Size	Value	Description
00h	**GetData**	Variable	Varies	Varies
Variable				

When a Get request addresses the Interface as a whole, the upper byte of wValue is *Reserved Out*. When the request addresses a particular LU of the Interface, then the upper byte of wValue is bLUN.

In any Get request, the upper byte of wIndex is *Reserved Out*.

The device *shall* complete the data in stage of a Get request within 500ms. The device *shall* complete the status stage of a Get request within 50ms. These requirements come from [Core USB].

6.1.1 Get Lock In (GLI)

This request lets the host see examine an LU.

Table 12 - Get Lock In (GLI)

Request

bmRequestType	bRequest	wValue	wIndex	wLength
1010 0001 b	GET	bLUN : GLI	*Reserved Out* : bInterfaceNumber	Max Data In

Data In

Offset	Field Name	Byte Size	Value	Description
00h	**LockData**	Variable	All or part of the LD	The initial bytes of the Lock Data (5.8 LD).
Variable				

The LU *shall* copy in the initial bytes of the Lock Data (5.8 LD), up to the wLength requested. The LU *shall* copy in the same value for the bLength field of the LD, no matter the value of wLength.

NOTE: The host *should* first request the wLength = 12h, to discover the bLength that begins the LD. The host *should* then request wLength = bLength, to complete the LD. The host *may* ask for just zero, one, or more bytes of the LD. The host *may* ask for more bytes than exist, up to FFFFh bytes.

6.2 Put Requests

The Put requests let the host ask for a response without data or to ask to copy data out. The following table provides the template for Put requests:

Table 13 – Put Request Template

Request

bmRequestType	bRequest	wValue	wIndex	wLength
0010 0001 b	PUT	Varies	*Reserved Out* : bInterfaceNumber	Length of Data Out

Data Out

Offset	Field Name	Byte Size	Value	Description
00h	**PutData**	Variable	Varies	Varies
Variable				

When a Put request addresses the Interface as a whole, the upper byte of wValue is *Reserved Out*. When the request addresses a particular LU of the Interface, then the upper byte of wValue is bLUN.

In any Put request, the upper byte of wIndex is *Reserved Out*.

The device *shall* complete the data out stage of a Put request within 500ms. The device *shall* complete the status stage of a Put request within 50ms. This requires the device to respond more quickly than [Core USB] mandates.

6.2.1 Store Passphrase Out (SPO)

The host sends this request to store the initial passphrase and hint.

Table 14 - Store Passphrase Out (SPO)

Request

bmRequestType	bRequest	wValue	wIndex	wLength
0010 0001 b	PUT	bLUN : SPO	*Reserved Out* : bInterfaceNumber	Length of Data Out

Data Out

Offset	Field Name	Byte Size	Value	Description
00h	**NewPhrase**	Variable	PD (5.5)	The new passphrase
(next)	**NewHint**	Variable	HD (5.6)	The new hint
wLength				

The LU *shall* reject this request unless:

a) The LU is Impersonal, and

b) The NewPhrase bLength is between 3 and 3+bMaxPhrase, inclusive, and

c) The NewHint bLength is between 3 and 3+bMaxHint, inclusive, and

d) The wLength equals the NewPhrase bLength plus the NewHint bLength.

To accept this request, the LU *shall* store the passphrase and hint, and *shall* move to Unlocked from Impersonal.

If a Cycle Power operation (4.1.1) interrupts this Put to LU request, the LU *shall* Personalize the LU and store the NewPhrase and NewHint as if this request completed successfully, or else the LU *shall* remain impersonal as if this request had never arrived.

6.2.2 Match Passphrase Out (MPO)

The host sends this request to resupply the passphrase.

Table 15 - Match Passphrase Out (MPO)

Request

bmRequestType	bRequest	wValue	wIndex	wLength
0010 0001 b	PUT	bLUN : MPO	*Reserved Out* : bInterfaceNumber	Length of Data Out

Data Out

Offset	Field Name	Byte Size	Value	Description
00h	**CandidatePhrase**	Variable	PD (5.5)	The candidate Passphrase
wLength				

The LU *shall* reject this request, unless:

a) The LU is Locked, and

b) The LU sees the CandidatePhrase as correct, and

c) The CandidatePhrase bLength is between 3 and 3+bMaxPhrase, inclusive, and

d) The wLength equals the CandidatePhrase bLength.

For this request, as for any “request to match a passphrase”,

a) If the fetch of the stored Passphrase inside the LU fails, the LU *shall* see a correct passphrase as incorrect.

b) To counteract dictionary attacks, the LU *may* see a correct passphrase as incorrect, when the correct passphrase arrives after one or more incorrect passphrases.

c) To counteract side-channel attacks, the LU *should* consume the same amount of time, power, electromagnetic spectrum, etc. before reporting that a candidate passphrase is correct or not, no matter what the stored passphrase was.

To accept this request, the LU *shall* move to Unlocked from Locked.

If a Cycle Power operation (4.1.1) interrupts this Put to LU request, the LU *shall* plug-in as Locked, as if this request had never arrived.

6.2.3 Change Passphrase Out (CPO)

The host sends this request to change the passphrase without erasing it.

Table 16 - Change Passphrase Out (CPO)

Request

bmRequestType	bRequest	wValue	wIndex	wLength
0010 0001 b	PUT	bLUN : CPO	*Reserved Out* : bInterfaceNumber	Length of Data Out

Data Out

Offset	Field Name	Byte Size	Value	Description
00h	**CandidatePhrase**	Variable	PD (5.5)	The candidate passphrase
(next)	**NewPhrase**	Variable	PD (5.5)	The new passphrase
(next)	**NewHint**	Variable	HD (5.6)	The new hint
wLength				

The LU *shall* reject this request, unless:

a) The LU is Unlocked, and

b) The LU sees the CandidatePhrase as correct, and

c) The bLength's of the CandidatePhrase and NewPhrase are between 3 and 3+bMaxPhrase, inclusive, and

d) The bLength of the NewHint is between 3 and 3+bMaxHint, inclusive, and

e) The wLength equals the sum of the bLength's of the CandidatePhrase, NewPhrase, and NewHint.

This request is a "request to match a passphrase" like the Match Passphrase Out (6.2.2 MPO) request is, therefore the LU *shall* see the CandidatePhrase as correct or incorrect as specified there (6.2.2).

To accept this request, the LU *shall* store the NewPhrase and NewHint in place of the old phrase and old hint.

If a Cycle Power operation (4.1.1) interrupts this Put to LU request, the LU *shall* erase the old phrase, erase the old hint, and plug-in Impersonal, or else the LU *shall* retain the old phrase, retain the old hint, and plug in Locked, as if this request had never arrived.

6.2.4 Erase Passphrase Out (EPO)

The host sends this request to erase a known passphrase, thus making the protected data accessible to legacy software and susceptible to loss and theft.

Table 17 - Erase Passphrase Out (EPO)

Request

bmRequestType	bRequest	wValue	wIndex	wLength
0010 0001 b	PUT	bLUN : EPO	*Reserved Out* : bInterfaceNumber	Length of Data Out

Data Out

Offset	Field Name	Byte Size	Value	Description
00h	**CandidatePhrase**	Variable	PD (5.5)	The candidate passphrase
wLength				

The LU *shall* reject this request, unless:

a) The LU is Unlocked, and

b) The LU sees the CandidatePhrase as correct, and

c) The CandidatePhrase bLength is between 3 and 3+bMaxPhrase, inclusive, and

d) The wLength equals the CandidatePhrase bLength.

This request is a “request to match a passphrase” like the Match Passphrase Out (6.2.2 MPO) request is, therefore the LU *shall* see the CandidatePhrase as correct or incorrect as specified there (6.2.2).

To accept this request, the LU *shall* (1) erase the old phrase, (2) erase the old hint, and (3) move to Impersonal from Locked, in that order.

If a Cycle Power operation (4.1.1) interrupts this Put to LU request, the LU *shall* erase the old phrase, erase the old hint, and plug-in Impersonal, or else the LU *shall* retain the old phrase, retain the old hint, and plug in Locked, as if this request had never arrived.

6.2.5 Erase Forgotten Passphrase (EFP)

The host sends this request to erase a forgotten passphrase.

Table 18 - Erase Forgotten Passphrase (EFP)

Request

bmRequestType	bRequest	wValue	wIndex	wLength
0010 0001 b	PUT	bLUN : EFP	*Reserved Out* : bInterfaceNumber	0000h, else *Checked Out*

No Data

The LU *shall* reject this request, unless:

a) The LU is Locked.

To accept this request, the LU *shall* (1) make the previously written data unretrievable, (2) erase the hint, (3) erase the passphrase, and (4) move to Impersonal from Locked, in that order.

If a Cycle Power operation (4.1.1) interrupts this Put to LU request, the LU *shall* make the previously written data unretrievable, erase the hint, erase the passphrase, and plug-in Impersonal, or else the LU *shall* retain the previously written data, retain the old phrase, retain the old hint, and plug in Locked, as if this request had never arrived.

If a Cycle Power operation interrupts a Recover Media operation after the device completes the Erase Forgotten Passphrase request that launched that operation, the LU *shall* continue the Recover Media operation as if the Cycle Power operation had not occurred. The subsequent response of the LU to a Query operation (4.1.9) then reports whether the Recover Media operation is still in progress or has completed.

6.2.6 Lock Again (LA)

This request lets the host ask an Unlocked LU to become Locked, without plugging the device back in.

Table 19 - Lock Again (LA)

Request

bmRequestType	bRequest	wValue	wIndex	wLength
0010 0001 b	PUT	bLUN : LA	*Reserved Out* : bInterfaceNumber	0000h, else *Checked Out*

No Data

The Interface *should* reject this request, unless:

a) The Interface is AnyPersonal.

The LU *shall* reject this request, unless:

a) The LU is Unlocked.

To accept this request, the Interface *shall* pass the request to the LU, and the LU *shall* move to Locked from Unlocked.

If a Cycle Power operation (4.1.1) interrupts this Put to LU request, the LU *shall* plug-in Locked, as if this request had never arrived.

6.2.7 Change Interface Access Out (CIAO)

This request lets the host ask the device to plug itself back in again, often with different Compatible IDs, without cycling power and thus relocking the Unlocked LUs of the device.

Table 20 – Change Interface Access Out (CIAO)

Request

bmRequestType	bRequest	wValue	wIndex	wLength
0010 0001 b	PUT	*Reserved Out* : CIAO	*Reserved Out* : bInterfaceNumber	000Ch, else *Checked Out*

Data Out

Offset	Field Name	Byte Size	Value	Description
00h	**AccessData**	Variable	AD (5.7)	When to plug back in with what IDs.
wLength				

The *Target Compatible ID* is the bInterfaceClass targetSubClass targetProtocol fields from the Interface Descriptor (5.3 ID) and the AD.

The Interface *shall* reject this request, unless:

a) The wLength equals the bLength of the AD (and that bLength is 0Ch), and

b) The Target Compatible ID correctly describes the Interface, and

c) The device has assigned a Hardware ID to one or more descriptor bundles that describe the Interface with the Target Compatible ID.

To accept this request, the interface *shall* launch a Cycle Presence operation (4.1.2) to plug the device back in again.

If power cycles, then every Unlocked LU *shall* move to Locked from Unlocked, but every Impersonal LU *shall* remain Impersonal. Thus the host *may* cancel this operation by cycling power at any time. But, if the host does not cancel, then:

The device *shall* idle at least for the dwIdleMs time. The host *should not* send new requests in this time. If a USB request arrives in this time, then the device *shall* Ack Control Setup and *may* Ack or Nak Control Data, but the device *shall not* Ack Control Status in. If a [BBB] [SCSI] request arrives in this time, then the interface *may* Ack or Nak the CBW, Data Out, and Data In, but *shall* Nak the CSW.

After the dwIdleMs time, during the dwGoneMs time, the device *shall* unplug (via floating FS D+, *etc.*). After that, then the device *shall* plug back in, as if the host had cycled power, except that the bLuState of the LDs of each LU *shall not* change. If dwGoneMs is zero, the device *shall* not unplug.

The device *shall* then present the target Compatible IDs for this Interface and the corresponding Hardware ID (5.2 DD, 5.3 ID).

The device *shall* idle for at least dwIdleMs before unplugging, and not more than 10 ms beyond dwIdleMs . The device *shall* then wait at least another dwGoneMs before ending the unplugged phase, and not more than10 ms beyond dwGoneMs.

The host *should not* ask for more than 45ms of dwIdleMs, to allow the device to respond promptly to USB requests. The host *should not* ask for more than 90ms of dwGoneMs, to allow the device respond within [Core USB] timing requirements to VBus provided. Some hosts *may* ask for more dwIdleMs and/or more dwGoneMs time to provide predictable operations in their environment.

A. Appendix: Assigned Codes

This specification *should* be self-consistent. If this specification disagrees with itself, then Appendix A *shall* win, else chapters 4 thru 9, else 1.4 Terms and Abbreviations, else chapters 1 thru 3 *shall* win.

A.1 Assigned bmDescriptorType Codes

The device needs assigned bmDescriptorType codes to implement chapter 5 “Data Structures”. These codes distinguish these descriptors from descriptors defined by [Core USB] and distinguish these descriptors from descriptors defined by other class-specific specifications.

Symbolic Value	Numeric Value	Description
LOCKABLE	25h	Identifies a Mass Storage interface elaborated by a Lockable Storage Interface Extension Descriptor (5.4 LSIED) that implements this specification.
PHRASE	LOCKABLE	Associates Phrase Data (5.5 PD) with this specification.
HINT	LOCKABLE	Associates Hint Data (5.6 HD) with this specification.
ACCESS	LOCKABLE	Associates Access Data (5.7 AD) with this specification.
QUERIED	LOCKABLE	Associates Lock Data (5.8 LD) with this specification.

This specification assigns one of the thirty-two descriptor numbers of the 08h Mass Storage Class.

The LSIED needs a distinct assigned LOCKABLE code because the LSIED appears mixed together with other descriptors in the configuration bundle fetched by a [Core USB] Get Descriptor, Configuration (GDC) request.

The PHRASE, HINT, ACCESS, and QUERIED codes do not need to be different from LOCKABLE, because the PD, HD, AD, and LD do not appear mixed with other descriptors in a bundle. Instead, each definition of a class-specific request specifies which data structures to copy in what order.

A.2 Assigned bRequest and wValue Codes

The device needs assigned bRequest codes to implement chapter 6 “Requests”. These codes distinguish these requests from requests defined by [Core USB] and distinguish these requests from requests defined by other class-specific specifications.

Symbolic Value	Numeric Value	Description
PUT	FCh	Identifies a Put request to act without data or a request to copy data out.
GET	FDh	Identifies a Get request to copy data in.

The interface needs assigned codes for the lower byte of wValue to implement chapter 6 “Requests”. These codes distinguish variations of the Put and Get requests from one another.

Symbolic Value	Numeric Value	Description
GLI	00h	Identifies a Get Lock In request.
SPO	01h	Identifies a Store Passphrase Out request.
MPO	02h	Identifies a Match Passphrase Out request.
CPO	03h	Identifies a Change Passphrase Out request.
EPO	04h	Identifies an Erase Passphrase Out request.
EFP	05h	Identifies an Erase Forgotten Passphrase request.
LA	06h	Identifies a Lock Again request.
CIAO	07h	Identifies a Change Interface Access Out request.

A.3 Assigned Compatible IDs

A.3.1 Negotiable IDs

The device needs Negotiable IDs to connect as Mass Storage without offering media access while Locked. These IDs tell newly standard software how to pass [SCSI] thru USB after an Unlock (4.1.4) or Recover Media (4.1.8) operation makes the LU Unlocked or Impersonal, without requiring a Cycle Presence operation (4.1.2).

The Negotiable IDs are { 08h, 07h, 50h } = { MassStorage, LockableSCSI, BBB }.

Symbolic Value	Numeric Value	Description
bInterfaceClass, N_SUBCLASS, N_PROTOCOL	08h, 07h, 50h	The values of bInterfaceClass, bInterfaceSubClass, and bInterfaceProtocol of the Interface after a power-cycle while any LU is Locked. This Compatible ID means the software *should* pass no [SCSI] thru USB while Locked. The software *should* pass [SCSI] thru USB when Unlocked or Impersonal as if the Compatible ID were h 08 06 50 (MassStorage TransparentSCSI BBB). The bVariation field of the LSIED tells the software how to discover the bLuState of the LU and otherwise manage passphrases: via the class-specific control pipe requests of this specification if C_LOCKABLE, else via [SAT] thru [SCSI] thru USB, as specified by 6 “Requests”.

This specification assigns one of the two-hundred fifty-six subclass codes and one of the two-hundred fifty-six protocol codes within that subclass.

To pass [SCSI] thru USB as if the Compatible ID where h 08 06 50, these Negotiable IDs inherit the restriction on Hardware IDs specified by [BBB] for that Compatible ID. Specifically, the SerialNumber ID string of the Hardware ID *shall* be distinct in the last twelve uppercase hex digits across all devices that share the same idVendor:idProduct and present an Interface Descriptor with these Negotiable IDs.

A.3.2 Legacy IDs

The device needs SCSI BBB IDs to connect with legacy Mass Storage software, while all the LUs of an Interface are Unlocked or Impersonal. Legacy specifications completely decide the meaning of SCSI BBB IDs.