1.0 Proposal for additions to SPC

This proposal adds a model for determining Logical Unit behavior. There are additions to the model/behavior section (section 5), new commands (Section 7), and a new set of configuration data (Features) (Section 8).

1.0.0.1 (5.1.5.1)Using the GET CONFIGURATION command

The GET CONFIGURATION command is used to report Features to the initiator.

Features are sets of commands, mode pages, and behaviors or operations specified for a logical unit. Each feature must be implemented entirely to its standard description in order to claim compliance with the feature. Except as explicitly identified, all commands, mode pages, and behaviors within a feature are mandatory. Each feature is associated with a Feature Descriptor.

Features were designed primarily to support multi-function devices that could only function as one device at a time, e.g. CD-R drives act as either a CD-R or CD-ROM depending on the medium. Virtually all removable medium devices are in effect multi-function devices: they can use their medium when present, but cannot perform any media operations when no medium is present.

Mode pages described and required by a feature ***shall*** always be present if that Feature is reported by the Logical Unit, regardless of whether or not the Feature is current. For example, the CD Audio parameters page ***shall*** be available for reading and writing if the CD Audio feature is supported by the device, even if no CD Audio media is present. The current values and changeable masks shall not change, even as features change their “current” state. Default values may change when morphing occurs. Default values shall always reflect a usable set of values for the loaded medium. Changes to the default values shall not generate a Unit Attention condition.

The use of features allows generic host drivers to use logical units that have among their many features some common core functionality. For example, the Random Readable feature may be reported by a very large variety of devices: magnetic disk, CD, DVD, or Magneto-Optical. A common driver to read data would be usable with all of these devices; special code would be needed only to manage extensions unique to each technology.

Features implemented by a logical unit are reported to the host via the GET CONFIGURATION command. This command should be used to identify all possible features, and those features that are current. A feature ***shall*** not be current if any of its mandatory commands or behaviors are not available. For example, a logical unit with writable media loaded and a mechanical write protect active ***shall*** not report any writable features as available. A DVD-ROM logical unit with a non-protected DVD-ROM loaded ***shall*** not report the DVD-CSS feature as being available. A logical unit with no medium present ***shall*** have no read or write or other medium dependent features active. Commands within a feature that is not current may still operate normally, especially when those commands are described in more than one feature.

The introduction of features is not intended to change device behavior. The use of commands that are not current will generate the same errors as legacy devices. Features simply provide a method for avoiding errors and avoids using errors to convey state information. When features are used properly by the host, the host should see only true medium errors and not need to do any capability discovery through error codes.

For a Feature to be considered current, all commands and behaviors described by that Feature should be available to the host. Even if a Feature is not current, its components should function if appropriate for the logical unit’s state. Commands received by a logical unit that are a member of a supported Feature that is not current shall either execute normally or return an appropriate error (i.e. incompatible medium, medium not present, etc.). Logical Units ***shall*** not terminate any Command that is a member of any supported Feature with an INVALID COMMAND OPERATION CODE error. For example, if the Formattable Feature is implemented, the READ FORMAT CAPACITIES command should return valid data regardless of whether or not the Formattable Feature is Current. An attempt to format a medium that cannot be formatted by the logical unit may return CHECK CONDITION status, ILLEGAL REQUEST, INCOMPATIBLE MEDIUM INSTALLED.

Each Feature Descriptor may contain information specific to that feature. The Feature specific information in the Feature Descriptor may not be valid if the Feature is not current.

Commands, pages, and behavior not described by a Feature may exist in the Logical Unit.

1.0.1 (5.1.6) Implementation of Features

1.0.1.1 (5.1.6.1) Introduction to Features

This specification introduces Features. Features were designed to be atomic units of functionality. On the first level, Features are only a description in a document. Traditional drivers work without modification with logical units that implement features. Features were a part of the documentation in SFF 8020, SFF 8090, and MMC; however they were not comprehensive, typically documenting only optional behavior. This specification associates all normal functionality with Features. Detection of a whole group of functions (a “Feature”) was typically accomplished by the host by issuing a command unique to that Feature and examining the completion status of that command.

The SFF and T10 (MMC) groups have been consciously trying to avoid using errors as a method for status and capability detection. Error handling code is typically one of the more complex parts of implementing drivers; reducing the number of cases that need to be handled helps implementations by reserving error status for only true errors. Status information is reported via explicit status reporting commands such as GET EVENT/STATUS NOTIFICATION and GET CONFIGURATION.

The descriptions of Features in this specification appear complex. However, these descriptions describe almost nothing new; they are simply the descriptions of existing legacy behavior. The only new parts are the descriptors themselves, which are either static identification blocks or groups of information that the Logical Unit must already have to operate, even in a legacy behavior. For example, a Logical Unit must internally identify whether or not a PLAY AUDIO command may succeed; Features are simply a way to let the host in on the secret.

Previously, new devices had to make a choice: to look completely like an old device with added functionality, or as a new device not compatible with old drivers. Using Features and Profiles, a host can first determine if the “right” driver is available by examining the profiles. If “the” right driver isn’t available, the host can identify operable subsets when multiple profiles are reported. Finally, the host can identify basic functions to use the device via the Feature reporting

1.0.1.2 (5.1.6.2) Feature History

The separation of status and error reporting is very important in multitasking environments. Typically, the operating system needs to constantly be aware of the status of the Logical Unit. Various applications, operating through a variety of OS interfaces, may also need to be aware of Logical Unit status. Reporting of status via errors breaks down in this environment; only one process is made aware of state changes via the error, while other processes cannot obtain the same state information because the error (status change) has already been reported to the host (according to the Logical Unit).

Features **do not** replace legacy behavior. Features, in most cases, define a subset of legacy behavior. Several Features, taken together, are generally equivalent to legacy devices of the same type. Error and status reporting in legacy host environments is the same as legacy devices, without any special mode setting.

The Features described in this specification add something new: reporting. Legacy devices, while implementing the content of the Features, did not have any mechanism to report specifically the Logical Unit’s capabilities. The closest mechanism that has existed is a command that reported implemented commands. Implemented mode pages are also reportable via standard mechanisms. However, a command is more than an operation code (opcode). A whole set of commands, mode pages, and behavior needs to be grouped together to be useful. For example, write once MO, hard disk drives, and CD-R all use the WRITE command, but it is impossible to use the same strategies for writing these three media. Typically, different drivers or fragments or drivers are used for each kind of media. The previous mechanism would only identify that the WRITE command was implemented, but could not identify how to use it.

The capabilities of a particular Logical Unit may change at arbitrary times. The most common example of this is seen in a removable medium device. Even a basic removable magnetic medium device changes: from a random read/write device to a virtually functionless device when the medium is removed. Multi-function devices can change their behavior even more radically when they accept a variety of physical and logical formats.

Before features, hosts had to use a trial and error method for determining what would or would not function. Medium codes became outdated even before publication of the relevant standard, and still were not adequate to describe all media. The Profiles, also introduced in this specification, provide an equivalent to the medium type. However, the profile does not indicate exact capabilities for the drive/medium system, only a generic identification of core capabilities.

Feature reporting is not completely new. Operating systems first identify a driver via the device type. The device type implied a core set of functions, e.g. a CD-ROM Logical Unit would support READ, READ TOC, etc. However, even these commands would not work if no medium were loaded. A driver would determine media status by trying a few commands and examining the error codes. After determining that media was present, a driver would have to probe to find out about additional features such as audio or medium changers. Features were identifiable, but each feature had a different mechanism, and many of the mechanisms relied on the success or failure of special “key” commands.

1.0.1.3 (5.1.6.3) Implementation of Features

There is only one requirement to implement Features: the GET CONFIGURATION command. This command is a very basic reporting command that reports some very static information; only a few features have any dynamic fields; most features have only one bit that changes. The command is a form of Inquiry: a technique for the host to identify the device on the bus. The GET CONFIGURATION command simply provides more detail, and the information reported is expected to be dynamic.

Implementation of Feature reporting via the GET CONFIGURATION command is simple: the image of the result data can be copied from device ROM to its buffer, a few fields set with information already known to the Logical Unit (such as the block size), and a few bits set according to already existing flags in the firmware (i.e. DVD vs. CD, audio tracks present, etc.). Devices with non-removable media may have a completely static image that is reported. If a starting point other than the beginning is requested, the Logical Unit walks the table to find the first requested feature, subtracts the offset from the data length, and transfers data starting at the same offset.

As mentioned earlier, Features are not new; their reporting is. This reporting has become very important in modern environments. Multiple drivers are talking to the same device, doing different tasks. For example, a DVD-ROM Logical Unit may use the basic CD-ROM driver when a CD is installed, and another driver when a DVD is installed, and both a basic DVD driver and a separate copy protection process when copy protected media is mounted. All of these processes must interact well to provide seamless and solid support. Feature reporting provides a method for clean interaction.

1.0.1.4 (5.1.6.4) Feature Compatibility

Logical Units implementing Feature reporting are fully compatible with legacy systems.

The GET CONFIGURATION changes no behavior of the Logical Unit; it simply reports existing state information. Repeated GET CONFIGURATION commands will report the same information (unless the user inserts or removes the medium, etc.). GET CONFIGURATION never changes any state information in the Logical Unit, including UNIT ATTENTION conditions.

1.0.1.5 Feature Summary

Features do not radically modify any legacy behavior or functionality. The only new part involves reporting of behavior, and typically reflects state information already required of any firmware implementation.

The benefits include easier coding of highly robust drivers, fewer error conditions, and forward and backward compatibility with operating system drivers.

1.0.2 (5.1.7) Profiles

Profiles define a base set of functions for logical units. Logical units that list a profile as current shall support all Features required by that Profile, but not all Features may be current. Logical units may support Features in addition to those required by the Profile. A single device may implement more than one Profile, and more than one Profile may be active at any given time. All required features may not be current, depending on the medium installed. If a Not Ready response would be given to a TEST UNIT READY command, no Profile shall be current.

For example, a logical unit with unformatted media may not be able to read or write, and the corresponding Features would not be current, but the Profile corresponding to the logical unit/media system may be current. i.e. a DVD-RAM drive with unformatted media loaded may claim compliance to the DVD-RAM profile; A DVD-RAM drive with no media loaded shall claim no Profile as current.

1.0.2.1 Example Profile: Profile 8: CD-ROM

Logical units identifying profile 8 as current shall support the features listed in *"Table 1 - Mandatory features for CD-ROM"* on page 4:

Table 1 - Mandatory features for CD-ROM

Feature Number	Feature Name	Description
0000h	Profile List	A list of all profiles supported by the device
0001h	Core	Mandatory behavior for all devices
0002h	Morphing	Ability to notify host about operational changes and accept host requests to prevent operational changes.
0003h	Removable Medium	The medium may be removed from the device
0010h	Random Readable, PP = 1	Read ability for storage devices with random addressing.
001Eh	CD Read	The ability to read CD specific structures
0100h	Power Management	Host and device directed power management
0105h	Timeout	Ability to respond to all commands within a specific time

1.0.2.2 Profile FFFFh: Logical Units Not Conforming to a Standard Profile

Logical units identifying profile FFFFh as current shall support the features listed in *"Table 2* - Mandatory Features for Logical Units Not Conforming to a Standard Profile":

Table 2 - Mandatory Features for Logical Units Not Conforming to a Standard Profile

Feature Number	Feature Name	Description
0000h	Profile List	A list of all profiles supported by the device
0001h	Core	Mandatory behavior for all devices

Logical Units that list Profile FFFFh shall list no other profiles.

1.1 (7.?) GET CONFIGURATION Command

This command is intended to provide information to the host about the overall capabilities of the device and the current capabilities of the device. Configurations reported by devices, for example, are used by the Host for Driver Identification/loading and other user presentation processes.

The GET CONFIGURATION command requests that the device respond with the configuration of the device and medium. The configuration of the device is described by features. The maximum number of features is 65,536; the maximum number of bytes that a device may return to describe its features in one command is 65,534. Feature lists longer than 65,534 bytes require multiple commands.

Table 3 - GET CONFIGURATION Command

Byte \ Bit	7	6	5	4	3	2	1	0
0	Operation code (46h)							
1	LUN (Obsolete)			Reserved			RT	
2	(MSB) Starting Feature Number							
3								(LSB)
4	Reserved							
5	Reserved							
6	Reserved							
7	MSB Allocation Length							
8								LSB
9	Vendor-Specific		Reserved			NACA	Flag	Link
10	PAD							
11								

The **RT** (Requested Type) field indicates the set of Feature Descriptors desired from the logical unit.

Table 4 - RT field definition

RT field	Description	Starting Feature Number (SFN) Usage
00b	Indicates that the logical unit ***shall*** return the Feature Header and all Feature Descriptors supported by the logical unit whether or not they are currently active.	The first Feature Descriptor returned ***shall*** have a feature number greater than or equal to the SFN.
01b	Indicates that the Feature Header and only those Feature Descriptors that have their Current bit set ***shall*** be returned.	
10b	Indicates that exactly one Feature Header and zero or one Feature Descriptors be returned. If the logical unit does not support the indicated feature, no Feature Descriptor is returned. Note: this may be used to request Feature 0, which is a list of Profiles.	The SFN specifies the Feature Descriptor that ***shall*** be returned.
11b	Reserved	

The **Starting Feature Number** indicates the first feature number to be returned. See *"Table 4 - RT field definition"* on page 5 for a more complete definition.

The **Allocation Length** field specifies the maximum length in bytes of the Get Configuration response data. An Allocation Length field of zero indicates that no data ***shall*** be transferred. This condition ***shall*** not be considered an error.

1.1.1 GET CONFIGURATION response data

The Response Data is a Configuration Data list and ***shall*** contain a header followed by zero or more variable length Feature Descriptors. The format of the Configuration Data is shown in *"Table 5 - GET CONFIGURATION response data format"* on page 6.

Table 5 - GET CONFIGURATION response data format

Byte \ Bit	7	6	5	4	3	2	1	0
0 - 7	Feature Header							
8 - n	Feature Descriptor(s)							

The Feature Header ***shall*** be returned as shown in *"Table 6 - Feature Header"* on page 6.

The Feature Descriptor(s) ***shall*** be returned as shown in *"Table 8 - Feature Descriptor generic format"* on page 8 and in each individual feature description.

Table 6 - Feature Header

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)							
1								
2				Data Length				
3								(LSB)
4	Reserved							
5	Reserved							
6	(MSB)							
7				Current Profile				(LSB)

The **Data Length** field indicates the amount of data available given a sufficient allocation length following this field. This length ***shall*** not be truncated due to an insufficient **Allocation Length**. If the **Data Length** is greater than 65,530 bytes, multiple GET CONFIGURATION commands with different **Starting Feature Numbers** will be required for the host to read all configuration data. This field is adjusted as appropriate for the given **Starting Feature Number**.

The **Current Profile** field ***shall*** indicate the logical unit's current profile. The logical unit ***shall*** choose the most appropriate current profile from the list of profiles with their **CurrentP** bit set. If no profile is current, this field ***shall*** contain zero.

1.1.2 Features

Features are the smallest implementable set of commands, pages, and behavior. *"Table 7 - Feature List"* on page 6 lists defined features.

Table 7 - Feature List

Feature Number	Feature Name	Description
0000h	Profile List	A list of all profiles supported by the device
0001h	Core	Mandatory behavior for all devices

Table 7 - Feature List (Continued)

Feature Number	Feature Name	Description
0002h	Morphing	Ability to notify host about operational changes and accept host requests to prevent operational changes.
0003h	Removable Medium	The medium may be removed from the device
0004h - 000Fh	Reserved	
0010h	Random Readable	Read ability for storage devices with random addressing
0011h - 001Ch	Reserved	
001Dh	Multi-read	The logical unit can read all CD media types; based on OSTA Multi-read
001Eh	CD Read	The ability to read CD specific structures
001Fh	DVD Read	The ability to read DVD specific structures
0020h	Random Writable	Write support for randomly addressed writes
0021h	Incremental Streaming Writable	Write support for sequential recording
0022h	Sector erasable	Write support for erasable media and media that requires an erase pass before overwrite.
0023h	Formattable	Support for formatting of media
0024h	Defect Management	Ability of the drive/media system to provide an apparently defect-free space
0025h	Write Once	Write support for write once media that can be written in random order
0026h	Restricted Overwrite	Write support for media that must be written in multiples of logical blocks
0027h - 002Ch	Reserved	
002Dh	CD Track at Once	Ability to write CD with Track at Once recording
002Eh	CD Mastering	The ability to write CD with Session at Once or Raw write methods.
002Fh	DVD-R Write	The ability to write DVD specific structures
0030h - 00FFh	Reserved	
0100h	Power Management	Host and device directed power management
0101h	S.M.A.R.T.	Self Monitoring Analysis and Reporting Technology (Failure prediction)
0102h	Embedded Changer	Single mechanism multiple disc changer
0103h	CD Audio analog play	Ability to play audio CDs via the drive’s own analog output
0104h	Microcode Upgrade	Ability for the device to accept new microcode via the interface
0105h	Time-out	Ability to respond to all commands within a specific time
0106h	DVD CSS	Ability to perform DVD CSS authentication and RPC
0107h	Real-Time Streaming	Ability to read and write using host requested performance parameters
0108h	Logical Unit serial number	The logical unit has a unique identifier.
0109h - FEFFh	Reserved	
FF00h - FFFFh	Vendor Unique	

Features are related by profiles. An example of some of the relationships is shown in *"Figure 31* - Example Feature Relationships". This diagram shows in a graphic form features that are defined in this specification. Each Feature is represented by a block in the diagram. Each Feature also shows an abbreviated list of the requirements for that Feature. This diagram serves as an example to help the reader understand the Features described in this specification, but should not be used as a reference for Feature implementation. In some cases, Features are dependent on other Features. This hierarchical relationship is shown in the diagram. If a Feature is placed underneath another Feature, then the underlying Feature may require some of the functionality of the overlying Feature. Items in quotes indicate a functionality that is required but not a specific command or page.

Each Feature supported by a logical unit ***shall*** be described by a Feature Descriptor. Each Feature Descriptor has its own parameters. All features ***shall*** be a multiple of four bytes long. The format of a Feature Descriptor is shown in *"Table 8 - Feature Descriptor generic format"* on page 8.

Table 8 - Feature Descriptor generic format

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB) Feature Code							
1								(LSB)
2	Reserved		Version				Persistent	Current
3	Additional Length							
4 - n	Feature Dependent Data							

The **Feature Code** field ***shall*** identify a feature supported by the logical unit.

The **Version** field is reserved and ***shall*** be set to zero. Future versions of a feature will be backward compatible; incompatible changes will be included in a different feature.

The **Persistent** bit, when set to zero, ***shall*** indicate that this feature may change its current status. When set to one, ***shall*** indicate that this feature is always active. The logical unit ***shall*** not set this bit to one if the Current bit is, or may become, zero.

The **Current** bit, when set to zero, indicates that this feature is not currently active and that the Feature Dependent Data may not be valid. When set to one, this feature is currently active and the Feature Dependent Data is valid.

The **Additional Length** field indicates the number of Feature specific bytes that follow this header. This field ***shall*** be an integral multiple of 4.

1.2 (8.5) Configuration Data (Features)

1.2.1 Feature 0000h: Profile List

The Profile List Feature is a list of all profiles supported by a device. This Feature is always current. The only change allowed in the Profile List during morphing is the setting of the CurrentP bits for each profile. Logical units that support removable medium ***shall*** not have any current profiles listed. Profile 0 ***shall*** not be reported in the Profile List, but may be reported in the Current Profile field of the GET CONFIGURATION header to indicate compliance to no profile.

Profiles provide a quick method for identifying the basic functionality of Logical Units. Logical Units may conform to more than one profile at a time. For example, a DVD-RAM drive with DVD-RAM media loaded may report both the Removable Disk and DVD-RAM profiles. This allows generic removable disk drivers to work with DVD-RAM media while also reporting the additional capabilities required by the DVD-RAM profile.

Table 9 - Feature 0000h: Profile List

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)			Feature Code = 0				
1								(LSB)
2	Reserved			Version			Persistent = 1	Current = 1
3	Additional Length							
4 - n	Profile Descriptor(s)							

The **Feature Code** field ***shall*** be set to zero.

The **Version** field is reserved and ***shall*** be set to zero. Future versions of a feature will be backward compatible; incompatible changes will be included in a different feature.

The **Persistent** bit ***shall*** be set to one to indicate that the reporting of the profile list is always supported.

The **Current** bit ***shall*** be set to one.

The **Additional Length** field ***shall*** be set to ((number of Profile Descriptors) * 4).

The Profile Descriptors are shown in *"Table 10 - Profile Descriptor"* on page 9. All profiles supported by the logical unit ***shall*** be always reported. Profile descriptors are returned in the order of preferred operation - most desirable to least desirable. E.g. a DVD-ROM that could also read CD-ROM would list the DVD-ROM profile first and the CD-ROM profile second.

Table 10 - Profile Descriptor

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)			Profile Number				
1								(LSB)
2	Reserved							CurrentP
3	Reserved							

The **Profile Number** identifies a profile to which the logical unit conforms. See *"Table 11 - Profile List"* on page 10.

The **CurrentP** bit, when set to one, ***shall*** indicate that this profile is currently active. If no medium is present, no profile should be active. Multifunction devices ***shall*** select the most appropriate profile(s), if any, to set as current. The most appropriate current profile is also reported in the Feature Header - see *"Table 6 - Feature Header"* on page 6.

Table 11 - Profile List

Profile Number	Profile Name	Description
0000h	Reserved	
0001h	Non-removable disk	Rewritable disk capable with unchanging behavior
0002h	Removable disk	Writable disk capable with removable media
0003h	MO Erasable	Magneto-Optical disk with sector erase capability
0004h	MO Write Once	Magneto-Optical write once
0005h - 0007h	Reserved	
0008h	CD-ROM	Read only Compact Disc capable
0009h	CD-R	Write once Compact Disc capable
000Ah	CD-RW	ReWritable Compact Disc capable
000Bh - 000Fh	Reserved	
0010h	DVD-ROM	Read only DVD
0011h	DVD-R	Write once DVD
0012h	DVD-RAM	Rewritable DVD
0013h - FFFEh	Reserved	
FFFFh	Logical Units Not Conforming to a Standard Profile	The logical unit does not conform to any profile.

Example: A DVD-ROM with CD-ROM read capability would always report two profiles. If no medium were present, the Current Profile field in the Feature Header would contain 0, and the CurrentP bits in both Profile Descriptors would be set to zero. If DVD-ROM media were inserted, the only change would be to set the CurrentP bit of the DVD-ROM profile to one. If CD-ROM media were then inserted, the CurrentP bit of the DVD-ROM profile would be set to zero and the CurrentP bit of the CD-ROM profile would be set to one.

1.2.1.1 Feature 0001h: Core

This feature describes basic logical unit functionality. This Feature ***shall*** always be current. All commands and functions described ***shall*** always function normally.

The INQUIRY command ***shall*** be supported. The INQUIRY command ***shall*** always complete without an error if the Command Packet is valid.

Logical Units ***shall*** be able to report sense to the host. For logical interfaces that report automatic delivery of Logical Unit Sense Information to the host ***shall*** use the transport's mechanism. For other logical interfaces, the REQUEST SENSE command ***shall*** be supported. The REQUEST SENSE command ***shall*** not generate any new sense information unless the Command Packet is invalid.

The MODE SENSE (10) command (see *"10.10 MODE SENSE Command"* on page 177) ***shall*** be supported. Logical Units may not return Block Descriptors. PC field values of 00b, 01b, and 10b ***shall*** be implemented for all supported mode pages. Logical Units ***shall*** be able to report mode pages whether or not appropriate media is loaded.

The MODE SELECT (10) command (see *"10.9 MODE SELECT Command"* on page 175) ***shall*** be supported. The SP bit may not be supported. Logical Units ***shall*** be able to accept mode pages whether or not appropriate media is loaded.

The GET CONFIGURATION command (see *"1.1 (7.?) GET CONFIGURATION Command"* on page 5) ***shall*** be supported. Unit Attention conditions ***shall*** not be reported to the GET CONFIGURATION command.

The TEST UNIT READY command (see *"10.38 TEST UNIT READY Command"* on page 329) ***shall*** be supported.

Logical Units ***shall*** be able to report Events to the host. For logical interfaces that support Event reporting to the host ***shall*** use the transport's mechanism. For other logical interfaces, the GET EVENT/STATUS NOTIFICATION command

(see *"1.4 SEND EVENT command"* on page 23) ***shall*** be supported. The host should determine supported events by issuing a GET EVENT/STATUS NOTIFICATION command with the Immed bit set. Zero or more event classes may be supported.

Table 12 - Feature 0001h: Core

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)							
1				Feature Code = 0001h				(LSB)
2	Reserved			Version			Persistent	Current
3				Additional Length = 04h				
4	(MSB)							
5								
6				Physical Interface Standard				
7								(LSB)

The **Feature Code** field ***shall*** be set to 0001h.

The **Persistent** bit ***shall*** be set to one.

The **Current** bit ***shall*** be set to one.

The **Additional Length** field ***shall*** be set to 4.

The Physical Interface Standard field ***shall*** be set to the current host to logical unit communication path as shown in *"Table 13 - Physical Interface Standard"* on page 11.

Table 13 - Physical Interface Standard

Physical Interface Standard	Description	Application
00000000h	Unspecified	
01h	SCSI Family	See “SCSI Implementation Notes” on page 347.
02h	ATAPI	See “ATAPI Implementation Notes” on page 337.
03h	IEEE 1394 Family	
04h - FEh	Reserved	
0000FFFFh	Vendor Unique	
00010000h - 0001FFFFh	Defined by NCITS	
00020000h - 0002FFFFh	Defined by SFF	
00030000h - 0003FFFFh	Defined by IEEE	
00040000h - FFFFFFFFh	Reserved	

1.2.1.2 Feature 0002h: Morphing

The Morphing Feature provides a method for identifying changes in logical unit behavior, and to some extent, preventing changes in logical unit behavior without host involvement. The Feature includes a mechanism for notifying the host about events that have occurred and requests for operational changes, a mechanism for identifying the logical unit’s current behavior, and a mechanism for allowing the logical unit to change its behavior. This feature, if implemented, ***shall*** always be current.

Generation of Event Notification class 1 events ***shall*** be supported.

The PREVENT/ALLOW command and the Persistent Prevent bit ***shall*** be supported. When a persistent prevent is in place, the Logical Unit ***shall*** not allow, to the limit of its design, non-host events to change the operational behavior of the device. Devices with a mechanical eject may not be able to prevent ejecting the media. When a persistent prevent is in place, events are reported to the host via the Get Event/Status Notification command instead of causing action within the logical unit. For example, if the user presses the eject button while a persistent prevent is in effect, the only action is to report the button press to the host. The logical unit shall behave as shown in *"Figure 1 - Morphing States"* on page 7.

The SEND EVENT command ***shall*** be supported for any Notification Event class 1 events that the device may generate. This command is used to tell the logical unit to perform an action that was previously requested by the Logical Unit via a Class 1 event notification. The host, after receiving a Class 1 notification, prepares for a possible logical unit change by notifying its drivers and flushing buffers as needed. After the host is prepared for a possible device change, it sends the Class 1 event descriptor back to the logical unit for processing. Event Notification Class 1 ***shall*** be suported.

Table 14 - Feature 0002h: Morphing

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)							
1	Feature Code = 0002h							(LSB)
2	Reserved		Version				Persistent	Current
3	Additional Length = 04h							
4	Reserved							Async
5	Reserved							
6	Reserved							
7	Reserved							

The **Feature Code** field ***shall*** be set to 0002h.

The **Persistent** bit ***shall*** be set to one.

The **Current** bit ***shall*** be set to one.

The **Additional Length** field ***shall*** be set to 4.

The **Async** bit, when set to zero, indicates that the logical unit supports only the polling implementation of GET EVENT/ STATUS NOTIFICATION. When set to one, indicates that the logical unit supports both polling and asynchronous GET EVENT/STATUS NOTIFICATION.