To: T10 Technical Committee
From: Rob Elliott, HP (elliott@hp.com)
Date: 19 February 2004
Subject: 03-388r1 SPC-3 SBC-2 Nonvolatile caches

Revision history

Revision 0 (11 December 2003) First revision
Revision 1 (19 February 2004) Incorporated comments from January CAP WG - made the existing behavior be force to medium and the new bit mean force only to NV cache, and added generic VPD bits in SPC-3 to indicate presence of volatile and/or non-volatile caches (which could be supported by non-block device types).

Related documents

sbc2r11 - SCSI Block Commands - 2 revision 11
spc3r17 - SCSI Primary Commands - 3 revision 17

Overview

SBC-2 has several tools that understand the concept of caches:

a) a force unit access (FUA) bit in each read and write command
b) LOCK UNLOCK CACHE, PRE-FETCH, PREVENT ALLOW MEDIUM REMOVAL, and SYNCHRONIZE CACHE commands which control cache usage
c) Caching mode page

However, it is does not comprehend the concept of a non-volatile cache. RAID controllers often employ battery-backed caches that can preserve the data for hours or days.

In many cases, when software wants to flush the cache with SYNCHRONIZE CACHE because it is concerned about temporary power loss, it is not necessary for the RAID controller to do anything; data is secure (at least for for a few days). In other cases, however, it does want the RAID controller flush its non-volatile cache to medium.

For write commands, if force unit access is enabled, sometimes software really means the the medium must be accessed, while other times it is sufficient to guarantee that it is in a non-volatile cache.

Forcing data to the medium is useful when disk drives in a RAID set are going to be moved to another controller. They need to hold a coherent set of data - some of the data cannot be left in a write cache. It also is appropriate when shutting down a server for extended periods of time. It is also useful for removeable media.

Allowing data to remain in non-volatile cache is acceptable if short power loss is expected or when a system is being rebooted. It may also be useful for metadata writes. An unexpected power loss causes more trouble if it loses metadata than if it loses data itself (may disrupt one file vs. many files).

Some operating systems overuse SYNCHRONIZE CACHE and force unit access, which hinders performance as the caches get larger and larger.

Proposal

Define the current behavior as:

a) SYNCHRONIZE CACHE flushes to the medium, not the non-volatile cache
b) FUA bit set to 1 on reads and writes forces access to the medium, not the non-volatile cache

Add a bit to SYNCHRONIZE CACHE to allow synchronization to the non-volatile cache rather than the medium.

Add a FUA_NV bit next to the FUA bit in the reads and writes to allow forcing to the non-volatile cache rather than the medium.

Add a way to report how much battery life remains for a non-volatile cache (if known) and the maximum possible battery life, so software can judge whether forcing to non-volatile cache meets its needs.

Add a bit to the Caching mode page to enable/disable write caching separately for volatile and non-volatile caches.

Suggested changes to SBC-2

3.1.3 cache memory: A temporary (and often volatile) data storage area outside the area accessible by application clients that may contain a subset of the data stored in the non-volatile data storage area.

3.1.16 non-volatile medium: A physical storage medium that retains data written to it for a subsequent read operation through power cycles. An example of this is a disk within a device that stores data as magnetic field changes that do not require device power to exist.

3.1.29 volatile medium: Medium that does not retain data written to it for a subsequent read operation through power cycles. An example of this is a silicon memory device that loses data written to it if device power is lost.

0.0.1 non-volatile cache memory: Cache memory that retains data through power cycles.

0.0.2 volatile cache memory: Cache memory that does not retain data through power cycles.

4.2 Direct-access device type model overview

Direct-access devices store user data for later retrieval in logical blocks. Logical blocks contain user data and may contain protection information. Each block of user data is stored at a unique logical block address (LBA). An application client uses write operations (e.g. WRITE commands) to store user data and read operations (e.g. READ commands) to retrieve user data. Other commands issued by the application client may alsocause write and read operations to occur. A write operation causes one or more logical blocks to be written on the medium. A read operation causes one or more logical blocks to be read from the medium. A verify operation confirms that one or more logical blocks were correctly written and can be read without error from the medium.

Logical blocks are stored by a process that causes localized changes or transitions within the medium. The changes made to the medium to store the logical blocks may be volatile (i.e., not retained through power cycles) or non-volatile (i.e., retained through power cycles). The medium may be divided in parts that are used for user data and protection information, parts that are reserved for defect management, and parts that are reserved for use by the controller for the management of the block device.

4.3 Removable medium

4.3.1 Removable medium overview

The medium may be removable (e.g., used in a floppy disk device) or non-removable (e.g., used in a fixed disk device). The removable medium may be contained within a cartridge (or jacket) to prevent damage to the recording surfaces.

A removable medium has an attribute of being mounted or unmounted on a suitable transport mechanism in a block device. A removable medium is mounted when the block device is capable of performing write or read operations to the medium. A removable medium is unmounted at any other time (e.g., during loading,unloading, or storage).

An application client may check whether a removable medium is mounted by issuing a TEST UNIT READY command. A block device containing a removable medium may need to receive a START STOP UNIT command to become accessible for write or read operations.

The PREVENT ALLOW MEDIUM REMOVAL command allows an application client to restrict the unmounting of the removable medium. This is useful in maintaining system integrity. If the block device implements cache memory (either volatile cache or non-volatile cache), it ensures that all logical blocks of the medium contain the most recent user data and protection information, if any, prior to permitting unmounting of the removable medium. If the application client issues a START STOP UNIT command to eject the removable medium, and the block device is prevented from unmounting by the PREVENT ALLOW MEDIUM REMOVAL command, the START STOP UNIT command is rejected by the device server.

4.6 Initialization

Block devices may require initialization prior to write or read operations. This initialization is performed by a FORMAT UNIT command (see 5.3). Parameters related to the geometry and performance characteristics may

be set with the MODE SELECT command prior to the format operation. Some block devices are initialized by means not specified in this standard. The time when the initialization occurs is vendor-specific.

Block devices using a non-volatile medium may save the parameters and only need to be initialized once. However, some mode parameters may need to be initialized after each logical unit reset. A catastrophic failure of the block device may require the FORMAT UNIT command to be reissued.

Block devices that use a volatile medium may need to be initialized after each logical unit reset prior to the processing of read or write operations. Mode parameters may also need initialization after logical unit resets.

NOTE 2 - After changing the block descriptor with MODE SELECT, the new values do not become effective until FORMAT UNIT command completes. Block descriptors read with the MODE SENSE command before a FORMAT UNIT complete return information that may not reflect the true state of the medium.

4.9 Cache memory

Some block devices implement cache memory. A cache memory is usually an area of temporary storage in the block device with a fast access time that is used to enhance performance. It exists separately from the stored logical blocks and is not directly accessible by the application client. Use of cache memory for write or read operations may reduce the access time to a logical block and can increase the overall data throughput.

Cache memory stores user data and protection information, if any.

Cache memory may be volatile or non-volatile. Volatile caches do not retain data through power cycles. Non-volatile caches retain data through power cycles. There may be a limit on the amount of time a non-volatile cache is able to retain data.

During read operations, the block device uses the cache memory to store logical blocks that the application client may request at some future time. The algorithm used to manage the cache memory is not part of this standard. However, parameters are provided to advise the device server about future requests, or to restrict the use of cache memory for a particular request.

During write operations, the block device uses the cache memory to store data that is written to the medium at a later time. This is called write-back caching. The command may complete prior to logical blocks being written to the medium. As a result of using a write-back caching there is a period of time when the data may be lost if power to the device is lost (for volatile caches) or a hardware failure occurs. There is also the possibility of an error occurring during the subsequent write operation. If an error occurred during the write, it may be reported as a deferred error on a later command. The application client may request that write-back caching be disabled with the Caching mode page (see 6.3.2) to prevent detected write errors from being reported by deferred errors. Even with write-back caching disabled undetected write errors may occur. In order to detect these errors, verify commands are provided.

When the cache memory fills up with logical blocks that are being kept for possible future access, new logical blocks that are to be kept replace those currently in cache memory. The disable page out (DPO) bit allows the application client to influence the replacement of logical blocks in the cache. For write operations, setting this bit to one advises the device server to not replace existing logical blocks in the cache memory with the write data. For read operations, setting this bit to one causes logical blocks that are being read to not replace existing ones in the cache memory.

Sometimes the application client may want to have the logical blocks read from the medium instead of from the cache memory. The force unit access (FUA) bit is used to specify that the device server shall access the ~~physical~~ medium. For a write operation, setting the FUA bit to one causes the device server to complete the data write to the ~~physical~~ medium before completing the command, and setting the FUA_NV bit to one allows the device server to complete the write to a non-volatile cache rather than the medium (see 5.26). For a read operation, setting the FUA bit to one causes the logical blocks to be retrieved from the ~~physical~~ medium, and setting the FUA_NV bit to one allows the device server to access the non-volatile cache rather than the medium (see 5.9).

When the DPO and FUA bits are both one, write and read operations, in effect, bypass the cache memory.

When a VERIFY command is processed, a forced unit access is implied, since the logical blocks stored on the medium are being verified. Furthermore, a synchronize cache operation is also implied to write unwritten logical blocks still in the cache memory. These logical blocks are stored on the medium before the verify

operation begins. The DPO bit is provided since the VERIFY command may cause the replacement of logical blocks in the cache. The caching rules also applies to the WRITE AND VERIFY command.

Commands may be implemented by the device server that allow the application client to control other behavior of the cache memory:

a) the LOCK UNLOCK CACHE command (see 5.4) controls whether certain logical blocks shall be held in the data cache for future use. Locking a logical block prevents its replacement by a future access. Unlocking a logical block exposes it to possible replacement by a future access;
b) the PRE-FETCH command (see 5.6) causes a set of logical blocks requested by the application client to be read into the data cache for possible future access. The logical blocks fetched are subject to later replacement unless they are locked;
c) the SYNCHRONIZE CACHE command (see 5.20) forces any pending write data in the requested set of logical blocks to be stored in the ~~physical~~ medium. This command may be used to ensure that the data was written and any detected errors reported;
d) the Caching mode page (see 6.3.2) writeable by the MODE SELECT command allows control of cache behavior and handles certain basic elements of cache replacement algorithms.

5 Commands for direct-access block devices

...

5.3 FORMAT UNIT command

5.3.1 FORMAT UNIT command overview

The FORMAT UNIT command (see table 12) formats the medium into application client addressable logical blocks per the application client defined options. In addition, the medium may be certified and control structures may be created for the management of the medium and defects. The degree that the medium is altered by this command is vendor-specific.

...

5.4 LOCK UNLOCK CACHE (10) command

The LOCK UNLOCK CACHE (10) command (see table 25) requests that the device server disallow or allow logical blocks within the specified range to be removed from the volatile cache memory and/or the non-volatile cache memory by the device server's cache replacement algorithm. Locked logical blocks may be written to the medium when modified, but a copy of the modified logical block shall remain in the cache memory.

...

A LOCK bit set to zero specifies that all logical blocks in the specified range that are currently locked into the cache memory shall be unlocked and may or may not be removed. A LOCK bit set to one specifies that any logical block in the specified range that is currently present in the cache memory shall be locked into cache memory. Only logical blocks that are already present in the cache memory are actually locked.

The NUMBER OF BLOCKS field specifies the total number of contiguous logical blocks within the range. A NUMBER OF BLOCKS field set to 0000h specifies that all remaining logical blocks shall be within the range. Multiple locks may be in effect from more than one initiator port. Locks from different initiator ports may overlap. An unlock of an overlapped area does not release the lock of another initiator port.

Editor's Note 1: I don't think it's necessary to make this command lock/unlock regions of the volatile and non-volatile caches separately. I have yet to hear of any devices implementing this command.

5.5 LOCK UNLOCK CACHE (16) command

The LOCK UNLOCK CACHE (16) command (see table 26) requests that the device server disallow or allow logical blocks within the specified range to be removed from the volatile cache memory and/or the non-volatile cache memory by the device server's cache replacement algorithm.

...

5.6 PRE-FETCH (10) command

The PRE-FETCH (10) command (see table 27) requests that the device server transfer the specified logical blocks from the medium to the volatile cache memory and/or the non-volatile cache memory. No data shall be transferred to the application client.

Editor's Note 2: I don't think it's necessary to make this command prefetch into the volatile and non-volatile caches separately. I have yet to hear of any devices implementing this command.

...

5.7 PRE-FETCH (16) command

The PRE-FETCH (16) command (see table 28) requests that the device server transfer the specified logical blocks from the medium to the volatile cache memory and/or the non-volatile cache memory. No data shall be transferred to the application client.

...

[**5.8 READ (6)** - no FUA bit today, so no need to add FUA_NV]

5.9 READ (10) command

The READ (10) command (see table 31) requests that the device server transfer data to the application client. Data includes user data and protection information, if any. The most recent data value written in the addressed logical block shall be returned.

Table 1 — READ (10) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (28h)							
1	RDPROTECT			DPO	FUA	Reserved	FUA_NV	Obsolete
2	(MSB)							
5	LOGICAL BLOCK ADDRESS							(LSB)
6	Reserved							
7	(MSB)							
8	TRANSFER LENGTH							(LSB)
9	CONTROL							

See the LOCK UNLOCK CACHE (10) command (see 5.2.3) for a definition of the LOGICAL BLOCK ADDRESS field.

...

~~A force unit access (FUA) bit of zero indicates that the device server may satisfy the command by accessing the cache memory. For read operations, any logical blocks that are contained in the cache memory may be transferred to the application client directly from the cache memory. For write operations, logical blocks may be transferred directly to the cache memory. GOOD status may be returned to the application client prior to writing the logical blocks to the medium. Any error that occurs after the GOOD status is returned is a deferred error, and information regarding the error is not reported until a subsequent command.~~

~~A (FUA) bit of one indicates that the device server shall access the media in performing the command prior to returning GOOD status. Read commands shall access the specified logical blocks from the media (i.e., the data is not directly retrieved from the cache). If the cache contains a more recent version of a logical block~~

~~than the media, the logical block shall first be written to the media. Write commands shall not return GOOD status until the logical blocks have actually been written on the media (i.e., the data is not write cached). Read commands that cause data to be written to the media from cache and that encounter an error shall cause a deferred error to be reported. See SPC-3.~~

The force unit access (FUA) and force unit access nonvolatile cache (FUA_NV) bits are defined in table 7.

Table 2 — Force unit access for reads

FUA	FUA_NV	**Description**
0	0	The device server may read the logical blocks from volatile cache, non-volatile cache, and/or the medium. It should read the logical blocks from a cache if available.
0	1	The device server shall read the logical blocks from non-volatile cache or the medium. If a volatile cache contains a more recent version of a logical block, the device server shall first write the logical block to: a) the non-volatile cache, if a non-volatile cache is present; or b) the non-volatile cache, if a non-volatile cache is present, and the medium, before reading it.
1	0 or 1	The device server shall read the logical blocks from the medium. If a cache contains a more recent version of a logical block, the device server shall first write the logical block to the non-volatile cache, if present, and the medium before reading it.

...

[5.10 READ (12) - add FUA_NV bit to byte 1 bit 1; point to READ (10) for definition]

[5.11 READ (16) - add FUA_NV bit to byte 1 bit 1; point to READ (10) for definition]

[5.12 READ CAPACITY (10) - does not perform logical block access]

[5.13 READ CAPACITY (16) - does not perform logical block access]

[5.14 READ DEFECT DATA (10) - does not perform logical block access]

[5.15 READ DEFECT DATA (12) - does not perform logical block access]

[5.16 READ LONG (10) - implicit force unit access to medium]

[5.17 READ LONG (16) - implicit force unit access to medium]

[5.18 REASSIGN BLOCKS - does not perform logical block access]

5.19 START STOP UNIT command

The START STOP UNIT command provides an application client a method to control the power condition of a logical unit (see 4.14.2). This includes specifying that the device server enable or disable the block device for medium access operations by controlling certain power conditions and timers.

Logical units that contain cache memory shall write all cached data to the medium for the logical unit, the same as they would do in response to a SYNCHRONIZE CACHE command (see 5.20 and 5.21) with the SYNC_NV bit set to zero, prior to entering into any power condition that prevents accessing the medium (e.g., before a hard drive stops its spindle motor during transition to the stopped power condition).

...

5.20 SYNCHRONIZE CACHE (10) command

~~The SYNCHRONIZE CACHE (10) command (see table 51) ensures that logical blocks in the cache memory, within the specified range, have their most recent data value recorded on the physical medium. Logical blocks include user data and protection information, if any. If a more recent data value for a logical block within the~~

~~specified range exists in the cache memory than on the physical medium, then the logical block from the cache memory shall be written to the physical medium. Logical blocks may not be removed from the cache memory as a result of the synchronize cache operation. The synchronize cache function is also required implicitly by other SCSI functions as defined in other clauses of this standard.~~

The SYNCHRONIZE CACHE (10) command (see table 51) ensures that logical blocks within the specified range have their most recent data value recorded in non-volatile cache or on the medium, based on the SYNC_NV bit. Logical blocks include user data and protection information, if any. Logical blocks may or may not be removed from the volatile cache and non-volatile cache as a result of the synchronize cache operation. The synchronize cache function is also required implicitly by other SCSI functions as defined in other subclauses of this standard.

Table 3 — SYNCHRONIZE CACHE (10) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (35h)							
1	Reserved					SYNC_NV	IMMED	Obsolete
2	(MSB)							
5	LOGICAL BLOCK ADDRESS							(LSB)
6	Reserved							
7	(MSB)							
8	NUMBER OF BLOCKS							(LSB)
9	CONTROL							

See 4.2.1.9 for reservation requirements for this command.

The SYNC_NV bit specifies whether the device server is required to synchronize volatile and non-volatile caches and is described in table 4.

Table 4 — SYNC_NV bit

SYNC_NV	Device server requirement to synchronize	
	Volatile cache	Non-volatile cache
0	shall synchronize to the medium	shall synchronize to the medium
1	shall synchronize to non-volatile cache or the medium	should not synchronize to the medium

An immediate (IMMED) bit of zero indicates that the status shall not be returned until the operation has been completed. An IMMED bit of one indicates that the device server shall return status as soon as the command descriptor block has been validated. If the IMMED bit is one and the device server does not support the IMMED bit, the command shall terminate with CHECK CONDITION status and the sense key shall be set to ILLEGAL REQUEST with the additional sense code set to INVALID FIELD IN CDB.

See the LOCK UNLOCK CACHE (10) command (see 5.2.3) for a definition of the RELADR bit and the LOGICAL BLOCK ADDRESS field.

The NUMBER OF BLOCKS field specifies the total number of contiguous logical blocks within the range. A number of blocks of zero indicates that all remaining logical blocks on the block device shall be within the range.

A logical block within the specified range that is not in cache memory is not considered an error.

5.21 SYNCHRONIZE CACHE (16) command

~~The SYNCHRONIZE CACHE (16) command (see table 5) ensures that logical blocks in the cache memory, within the specified range, have their most recent data value recorded on the physical medium. If a more recent data value for a logical block within the specified range exists in the cache memory than on the physical medium, then the logical block from the cache memory shall be written to the physical medium. Logical blocks may not be removed from the cache memory as a result of the synchronize cache operation. The synchronize cache function is also required implicitly by other SCSI functions as defined in other clauses of this standard.~~

The SYNCHRONIZE CACHE (16) command (see table 52) ensures that logical blocks within the specified range have their most recent data value recorded in non-volatile cache or on the medium, based on the SYNC_NV bit. Logical blocks include user data and protection information, if any. Logical blocks may or may not be removed from the volatile cache and non-volatile cache as a result of the synchronize cache operation. The synchronize cache function is also required implicitly by other SCSI functions as defined in other subclauses of this standard.

Table 5 — SYNCHRONIZE CACHE (16) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (91h)							
1	Reserved					SYNC_NV	IMMED	Reserved
2	(MSB)							
9	LOGICAL BLOCK ADDRESS							(LSB)
10	(MSB)							
13	NUMBER OF BLOCKS							(LSB)
14	Reserved							
15	CONTROL							

See 4.2.1.9 for reservation requirements for this command. See the SYNCHRONIZE CACHE (10) command (see) for a description of the fields in this command.

[5.22 VERIFY (10) - already has an implicit force access to medium]

[5.23 VERIFY (12) - already has an implicit force access to medium]

[5.24 VERIFY (16) - already has an implicit force access to medium]

[5.25 WRITE (6) - no existing FUA bit, so doesn't make sense to add FUA_NV]

5.26 WRITE (10) command

The WRITE (10) command (see table 6) requests that the device server write the data transferred by the application client to the medium. Data transferred from the application client includes user data and includes protection information as required by the WRPROTECT field and the medium format.

Table 6 — WRITE (10) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (2Ah)							
1	WRPROTECT			DPO	FUA	Reserved	~~Reserved~~ FUA_NV	Obsolete
2	(MSB)	LOGICAL BLOCK ADDRESS						
5								(LSB)
6	Reserved							
7	(MSB)	TRANSFER LENGTH						
8								(LSB)
9	CONTROL							

See the READ (10) command (see) for a definition of the DPO bit ~~and the FUA bit~~. See the LOCK UNLOCK CACHE (10) command (see 5.2.3) for a definition of the LOGICAL BLOCK ADDRESS field.

The force unit access (FUA and FUA_NV) bits are defined in table 7.

Table 7 — Force unit access for writes

FUA	FUA_NV	**Description**
0	0	The device server shall write the logical blocks to volatile cache, non-volatile cache, and/or the medium. If a cache is present and in write-back mode, it should not write the logical blocks to the medium.
0	1	The device server shall write the logical blocks to non-volatile cache and/or the medium. If a non-volatile cache is present and in write-back mode, it should not write the logical blocks to the medium.
1	0 or 1	The device server shall write the logical blocks to the medium, and shall not return GOOD status until the logical blocks have actually been written on the medium.

If logical blocks are transferred directly to a cache memory, GOOD status may be returned to the application client prior to writing the logical blocks to the medium. Any error that occurs after the GOOD status is returned is a deferred error, and information regarding the error is not reported until a subsequent command.

...

Editor's Note 3: make sure all the fua references point to the correct read or write sections

[5.27 WRITE (12) - add FUA_NV to byte 1 bit 1, point to WRITE (10)]

[5.28 WRITE (16) - add FUA_NV to byte 1 bit 1, point to WRITE (10)]

[5.29 WRITE AND VERIFY (10) - already has an implicit force unit access to the medium]

[5.30 WRITE AND VERIFY (12) - already has an implicit force unit access to the medium]

[5.31 WRITE AND VERIFY (16) - already has an implicit force unit access to the medium]

[5.32 WRITE LONG (10) command - already has an implicit force unit access to the medium]

[5.33 WRITE LONG (16) command - already has an implicit force unit access to the medium]

[5.34 WRITE SAME (10) command - already has an implicit force unit access to the medium]

[5.35 WRITE SAME (16) command - already has an implicit force unit access to the medium]

[5.36 XDREAD (10) - just reads a temporary buffer, not the medium]

[5.37 XDREAD (32) - just reads a temporary buffer, not the medium]

[5.38 XDWRITE (10) - add FUA_NV to byte 1 bit 1, point to WRITE (10)]

[5.39 XDWRITE (32) - add FUA_NV to byte 10 bit 1, point to WRITE (10)]

[5.40 XDWRITEREAD (10) - add FUA_NV to byte 1 bit 1, point to both WRITE (10) and READ (10)]

[5.41 XDWRITEREAD (32) - add FUA_NV to byte 10 bit 1, point to both WRITE (10) and READ (10)]

[5.42 XPWRITE (10) - add FUA_NV to byte 1 bit 1, point to WRITE (10)]

[5.43 XPWRITE (32) - add FUA_NV to byte 10 bit 1, point to WRITE (10)]

6.2 Log parameters

6.2.1 Log parameters overview

This subclause defines the descriptors and pages for log parameters used with direct-access block devices. See SPC-3 for a detailed description of logging operations. The log page codes for direct-access block devices are defined in table 8.

Table 8 — Log page codes

Log page code	Description	Reference
00h	Supported log pages	SPC-3
01h	Buffer Overrun/Underrun log page	SPC-3
02h	Write Error Counter log page	SPC-3
03h	Read Error Counter log page	SPC-3
04h	Reserved	
05h	Verify Error Counter log page	SPC-3
06h	Non-Medium Error log page	SPC-3
07h	Last N Error Events log page	SPC-3
08h	Format Status log page	
09h	Non-volatile Cache log page	6.1.2.x
~~09h -~~ 0Ah	Reserved	
0Bh	Last N Deferred Error Events log page	SPC-3
0Ch	Reserved	
0Dh	Temperature log page	SPC-3
0Eh	Start-Stop Cycle Counter log page	SPC-3
0Fh	Application Client log page	SPC-3
10h	Self-Test Results log page	SPC-3
11h - 2Eh	Reserved	
2Fh	Information Exceptions log page	SPC-3
30h - 3Eh	Vendor-specific log pages	
3Fh	Reserved	

6.2.n Non-volatile Cache log page [new section]

The Non-volatile Cache log page defined in table 9 indicates the status of battery backup for a non-volatile cache.

Table 9 — Non-volatile Cache log page

Byte\Bit	7	6	5	4	3	2	1	0
0	PAGE CODE (xxh)							
1	Reserved							
2	(MSB)							
3	PAGE LENGTH (n - 3)							(LSB)
	Non-volatile cache log parameters							
4								
	First non-volatile cache log parameters							
	...							
	Last non-volatile cache log parameters							
n								

Table 10 defines the parameter codes.

Table 10 — Non-volatile Cache log parameters

Code	Description
0000h	Remaining Non-volatile Time
0001h	Maximum Non-volatile Time
All others	Reserved

The Remaining Non-volatile Time parameter has the format shown in table 11.

Table 11 — Remaining Non-volatile Time parameter data

Byte\Bit	7	6	5	4	3	2	1	0
0	PARAMETER LENGTH (3h)							
1	(MSB)							
3	REMAINING NON-VOLATILE TIME							(LSB)

The REMAINING NON-VOLATILE TIME field is defined in table 12 .

Table 12 — Remaining non-volatile time

Code	Description
000000h	Non-volatile cache is volatile (either permanently or temporarily, e.g., if batteries need to be recharged).
000001h	Non-volatile cache is expected to remain non-volatile for an unknown amount of time (e.g., if battery status is unknown)
000002h to FFFFFEh	Non-volatile cache is expected to remain non-volatile for the number of minutes indicated (e.g., battery-backed random access memory).
FFFFFFh	Non-volatile cache is indefinitely non-volatile.

The Maximum Non-volatile Time parameter has the format shown in table 13.

Table 13 — Maximum Non-volatile Time parameter data

Byte\Bit	7	6	5	4	3	2	1	0
0	PARAMETER LENGTH (3h)							
1	(MSB)	MAXIMUM NON-VOLATILE TIME						
3								(LSB)

The MAXIMUM NON-VOLATILE TIME field is defined in table 14.

Table 14 — Maximum non-volatile time

Code	Description
000000h	Non-volatile cache is volatile
000001h	Reserved
000002h to FFFFFEh	Non-volatile cache is capable of being non-volatile for the estimated number of minutes indicated. If the time is based on batteries, it shall be based on the last full charge capacity rather than the design capacity of the batteries.
FFFFFFh	Non-volatile cache is indefinitely non-volatile.

Editor's Note 4: The Smart Battery Data specification (www.sbs-forum.org) indicates batteries only estimate their own times in minutes, not seconds. 136 years can be represented in a four byte seconds field, so total expressable time is not an issue (minutes bump that to 8165 years). Minutes chosen for this proposal.

6.3 Mode parameters

6.3.1 Mode parameters overview

This subclause defines the descriptors and pages for mode parameters used with direct-access device types.

The mode parameter list, including the mode parameter header and mode block descriptor are described in SPC-3.

The MEDIUM TYPE field is contained in the mode parameter header (see SPC-3). Table 104 defines this field for direct-access block devices.

...

The DEVICE-SPECIFIC PARAMETER field (see table 15) is contained in the mode parameter header (see SPC-3).

Table 15 — Device-specific parameter

Bit	7	6	5	4	3	2	1	0
	WP	Reserved		DPOFUA	Reserved			

When used with the MODE SELECT command the write protect (WP) bit is not defined.

When used with the MODE SENSE command a WP bit of zero indicates that the medium is write enabled. A WP bit of one indicates that the medium is write protected.

When used with the MODE SELECT command, the DPOFUA bit is not used and the field is reserved.

When used with the MODE SENSE command, a DPOFUA bit of zero indicates that the device server does not support the DPO and FUA bits. When used with the MODE SENSE command, a DPOFUA bit of one indicates that the device server supports the DPO and FUA bits and may support the FUA_NV bit (see 4.2.1.8).

The DENSITY CODE field is contained in the mode parameter block descriptor (see SPC-3). This field is reserved for direct-access block devices.

...

6.3.2 Caching mode page

The Caching mode page (see table 16) defines the parameters that affect the use of the cache(s).

Table 16 — Caching mode page

Byte\Bit	7	6	5	4	3	2	1	0
0	PS	Reserved	PAGE CODE (08h)					
1	PAGE LENGTH (12h)							
2	IC	ABPF	CAP	DISC	SIZE	WCE	MF	RCD
3	DEMAND READ RETENTION PRIORITY				WRITE RETENTION PRIORITY			
4	(MSB)	DISABLE PRE-FETCH TRANSFER LENGTH						
5								(LSB)
6	(MSB)	MINIMUM PRE-FETCH						
7								(LSB)
8	(MSB)	MAXIMUM PRE-FETCH						
9								(LSB)
10	(MSB)	MAXIMUM PRE-FETCH CEILING						
11								(LSB)
12	FSW	LBCSS	DRA	Vendor specific		Reserved		NV_DIS
13	NUMBER OF CACHE SEGMENTS							
14	(MSB)	CACHE SEGMENT SIZE						
15								(LSB)
16	Reserved							
17	(MSB)	NON CACHE SEGMENT SIZE						
19								(LSB)

The parameters savable (PS) bit is only used with the MODE SENSE command. This bit is reserved with the MODE SELECT command. A PS bit of one indicates that the device server is capable of saving the mode page in a non-volatile vendor-specific location. If the PS is one in MODE SENSE data then the mode page shall be savable by issuing a MODE SELECT command with the SP bit of one.

The initiator control (IC) enable bit, when one, requests that the device server use the number of CACHE SEGMENTS or CACHE SEGMENT SIZE fields, dependent upon the Size bit, to control the caching algorithm rather than the device server's own adaptive algorithm.

The abort pre-fetch (ABPF) bit, when one, with the DRA bit equal to zero, requests that the device server abort the pre-fetch upon receipt of a new command. The ABPF bit of one takes precedence over the Minimum Pre-fetch bytes. When the ABPF bit is zero, with the DRA bit equal to zero, the termination of any active pre-fetch is dependent upon Caching mode page bytes 4 through 11 and is operation and/or vendor-specific.

The caching analysis permitted (CAP) bit, when one, requests that the device server perform caching analysis during subsequent operations. When zero, CAP requests that caching analysis be disabled to reduce overhead time or to prevent nonpertinent operations from impacting tuning values.

The discontinuity (DISC) bit, when one, requests that the device server continue the pre-fetch across time discontinuities, such as across cylinders (or tracks in an embedded servo device), up to the limits of the buffer, or segment, space available for the pre-fetch. When zero, the DISC requests that pre-fetches be truncated (or wrapped) at time discontinuities.

The size enable (SIZE) bit, when one, indicates that the CACHE SEGMENT SIZE is to be used to control caching segmentation. When SIZE equals zero, the application client requests that the NUMBER OF CACHE SEGMENTS is to be used to control caching segmentation. Simultaneous use of both the number of segments and the segment size is vendor-specific.

A writeback cache enable (WCE) bit of zero specifies that the device server shall return GOOD status for a WRITE command only after successfully writing all of the data to the medium. A WCE bit of one specifies that the device server may return GOOD status for a WRITE command after successfully receiving the data and prior to having successfully written it to the medium.

A multiplication factor (MF) bit of zero specifies that the device server shall interpret the MINIMUM and MAXIMUM PRE-FETCH fields in terms of the number of logical blocks for each of the respective types of pre-fetch. An MF bit of one specifies that the device server shall interpret the MINIMUM and MAXIMUM PRE-FETCH fields to be specified in terms of a scalar number that, when multiplied by the number of logical blocks to be transferred for the current command, yields the number of logical blocks for each of the respective types of pre-fetch.

A read cache disable (RCD) bit of zero specifies that the device server may return data requested by a READ command by accessing either the cache or medium. A RCD bit of one specifies that the device server shall transfer all of the data requested by a READ command from the medium (i.e., data shall not be transferred from the cache).

The DEMAND READ RETENTION PRIORITY field (see table 17) advises the device server the retention priority to assign for data read into the cache that has also been transferred from the logical unit to the application client.

Table 17 — Demand read retention priority and write retention priority

Value	Description
0h	Indicates the device server should not distinguish between retaining the indicated data and data placed into the cache memory by other means (e.g., pre-fetch).
1h	Demand read retention priority: Data put into the cache via a READ command should be replaced sooner (has lower priority) than data placed into the cache by other means (e.g., pre-fetch). Write retention priority: Data put into the cache during a WRITE or WRITE AND VERIFY command should be replaced sooner (has lower priority) than data placed into the cache by other means (e.g., pre-fetch).
2h - Eh	Reserved
Fh	Demand read retention priority: Data put into the cache via a READ command should not be replaced if there is other data in the cache that was placed into the cache by other means (e.g., pre-fetch) and it may be replaced (i.e., it is not locked). Write retention priority: Data put into the cache during a WRITE or WRITE AND VERIFY command should not be replaced if there is other data in the cache that was placed into the cache by other means (e.g., pre-fetch) and it may be replaced (i.e., it is not locked).

The WRITE RETENTION PRIORITY field advises the device server the retention priority to assign for data written into the cache that has also been transferred from the cache memory to the medium.

An anticipatory pre-fetch occurs when data is placed in the cache that has not been requested. This may happen in conjunction with the reading of data that has been requested. All the following parameters give an indication to the device server how it should manage the cache based on the last READ command. An anticipatory pre-fetch may occur based on other information. All the remaining caching parameters are only recommendations to the device server and should not cause a CHECK CONDITION to occur if the device server is not able to satisfy the request.

The DISABLE PRE-FETCH TRANSFER LENGTH field specifies the selective disabling of anticipatory pre-fetch on long transfer lengths. The value in this field is compared to the number of blocks requested by the current READ command. If the number of blocks is greater than the disable pre-fetch transfer length, then an

anticipatory pre-fetch is not done for the command. Otherwise the device server should attempt an anticipatory pre-fetch. If the pre-fetch disable transfer length is zero, then all anticipatory pre-fetching is disabled for any request for data, including those for zero logical blocks.

The MINIMUM PRE-FETCH field indicates either a number of blocks or a scalar multiplier of the TRANSFER LENGTH, depending upon the setting of the MF bit. In either case, the resulting number of blocks is the number to pre-fetch regardless of the delays it might cause in executing subsequent commands.

The pre-fetching operation begins at the logical block immediately after the last logical block of the previous READ command. Pre-fetching shall always halt before exceeding the end of the medium. Errors that occur during the pre-fetching operation shall not be reported to the application client unless the device server is unable to, as a result of the error, execute subsequent commands correctly. In this case the error may be reported either immediately as an error for the current READ command, or as a deferred error, at the discretion of the device server and according to the rules for reporting deferred errors.

If the pre-fetch has read more than the amount of data indicated by the MINIMUM PRE-FETCH then pre-fetching should be terminated whenever another command is ready to execute. This consideration is ignored when the MINIMUM PRE-FETCH is equal to the MAXIMUM PRE-FETCH.

The MAXIMUM PRE-FETCH field indicates either a number of blocks or a scalar multiplier of the TRANSFER LENGTH, depending upon the setting of the MF bit. In either case, the resulting number of blocks is the number to pre-fetch if the pre-fetch does not delay executing subsequent commands.

The MAXIMUM PRE-FETCH field contains the maximum amount of data to pre- fetch into the cache as a result of one READ command. It is used in conjunction with the DISABLE PRE-FETCH TRANSFER LENGTH and MAXIMUM PRE- FETCH CEILING parameters to trade off pre-fetching new data with displacing old data already stored in the cache.

The MAXIMUM PRE-FETCH CEILING field specifies an upper limit on the number of logical blocks computed as the maximum pre-fetch. If this number of blocks is greater than the MAXIMUM PRE-FETCH, then the number of logical blocks to pre-fetch shall be truncated to the value stored in the MAXIMUM PRE-FETCH CEILING field.

> NOTE 1 - If the MF bit is one the MAXIMUM PRE-FETCH CEILING field is useful in limiting the amount of data to be pre-fetched.

The force sequential write (FSW) bit when one, indicates that multiple block writes are to be transferred over the SCSI bus and written to the medium in an ascending, sequential, logical block order. When the FSW bit equals zero, the device server is allowed to reorder the sequence of writing addressed logical blocks in order to achieve a faster command completion.

The logical block cache segment size (LBCSS) bit when one, indicates that the CACHE SEGMENT SIZE field units shall be interpreted as logical blocks. When the LBCSS bit equals zero the CACHE SEGMENT SIZE field units shall be interpreted as bytes. The LBCSS shall not impact the units of other fields.

The disable read-ahead (DRA) bit, when one, requests that the device server not read into the buffer any logical blocks beyond the addressed logical block(s). When the DRA bit equals zero, the device server may continue to read logical blocks into the buffer beyond the addressed logical block(s).

The vendor-specific (VS) bits may optionally be used for vendor-specific purposes.

The NUMBER OF CACHE SEGMENTS advises the device server how many segments the host requests that the cache be divided into.

The CACHE SEGMENT SIZE field indicates the requested segment size in bytes. This standard defines that the CACHE SEGMENT SIZE field is valid only when the SIZE bit is one.

An NV_DIS bit set to one specifies that the device server shall disable a non-volatile cache and indicates that a non-volatile cache is supported but disabled. An NV_DIS bit set to zero specifies that the device server may use a non-volatile cache and indicates that a non-volatile cache may be present and enabled.

If the NON CACHE BUFFER SIZE field is greater than zero, this field advises the device server how many bytes the application client requests that the device server allocate for a buffer function when all other cache segments are occupied by data to be retained. If the number is at least one, caching functions in the other segments need not be impacted by cache misses to perform the SCSI buffer function. A NON CACHE SEGMENT

SIZE field set to zero is vendor-specific. If the sum of the NON CACHE SEGMENT SIZE field value plus the CACHE SEGMENT SIZE field value is greater than the buffer size, the result is vendor-specific.

Suggested changes to SPC-3

7.6.5 Extended INQUIRY Data VPD page

The Extended INQUIRY Data VPD page (see table 297) provides the application client with a means to obtain information about the logical unit.

Table 18 — Extended INQUIRY Data VPD page

Byte\Bit	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (86h)							
2	Reserved							
3	PAGE LENGTH (3Ch)							
4	Reserved					GRD_CHK	APTG_CHK	RFTG_CHK
5	Reserved					HEADSUP	ORDSUP	SIMPSUP
6	Reserved						NV_SUP	V_SUP
7	Reserved							
63								

The PERIPHERAL QUALIFIER field and the PERIPHERAL DEVICE TYPE field are as defined in 6.4.2.

The PAGE LENGTH field specifies the length of the following VPD page data and shall be set to 60. If the allocation length is less than the length of the data to be returned, the page length shall not be adjusted to reflect the truncation.

A guard check (GRD_CHK) bit set to zero indicates the device server does not check the LOGICAL BLOCK GUARD field in the protection information (see SBC-2) before transmitting it to an application client. A GRD_CHK bit set to one indicates the device server checks the LOGICAL BLOCK GUARD field in the protection information before transmitting it to an application client. If the application client or device server detects a LOGICAL BLOCK APPLICATION TAG field containing FFFFh, the checking of the LOGICAL BLOCK GUARD field in the protection information shall not be performed for the associated logical block.

An application tag check (APTG_CHK) bit set to zero indicates the device server does not check the LOGICAL BLOCK APPLICATION TAG field in the protection information (see SBC-2) before transmitting it to an application client. An APTG_CHK bit set to one indicates the device server checks the LOGICAL BLOCK APPLICATION TAG field in the protection information before transmitting it to an application client. If the application client or device server detects a LOGICAL BLOCK APPLICATION TAG field containing FFFFh, the checking of the LOGICAL BLOCK APPLICATION TAG field in the protection information shall not be performed for the associated logical block.

A reference tag check (RFTG_CHK) bit set to zero indicates the device server does not check the LOGICAL BLOCK REFERENCE TAG field in the protection information (see SBC-2) before transmitting it to an application client. A RFTG_CHK bit set to one indicates the device server checks the LOGICAL BLOCK REFERENCE TAG field in the protection information before transmitting it to an application client. If the application client or device server detects a LOGICAL BLOCK APPLICATION TAG field containing FFFFh, the checking of the LOGICAL BLOCK REFERENCE TAG field in the protection information shall not be performed for the associated logical block.

A head of queue supported (HEADSUP) bit set to one shall indicate that the HEAD OF QUEUE task attribute (see SAM-3) is supported by the logical unit. A HEADSUP bit set to zero shall indicate that the HEAD OF QUEUE task attribute is not supported. If the HEADSUP bit is set to zero application clients should not specify the HEAD OF QUEUE task attribute as an Execute Command (see 4.2) procedure call argument.

An ordered supported (ORDSUP) bit set to one shall indicate that the ORDERED task attribute (see SAM-3) is supported by the logical unit. An ORDSUP bit set to zero shall indicate that the ORDERED task attribute is not supported. If the ORDSUP bit is set to zero application clients should not specify the ORDERED task attribute as an Execute Command procedure call argument.

A simple supported (SIMPSUP) bit set to one shall indicate that the SIMPLE task attribute (see SAM-3) is supported by the logical unit. Logical units that support the full task management model (see SAM-3) shall set the SIMPSUP bit to one. A SIMPSUP bit set to zero shall indicate that the SIMPLE task attribute is not supported. If the SIMPSUP bit is set to zero application clients should not specify the SIMPLE task attribute as an Execute Command procedure call argument.

SAM-3 defines how unsupported task attributes are processed.

A V_SUP bit set to one indicates that the device server supports a volatile cache. An V_SUP bit set to zero indicates that the device server may or may not support a volatile cache with command set-specific features to control usage.

An NV_SUP bit set to one indicates that the device server supports a non-volatile cache. An NV_SUP bit set to zero indicates that the device server may or may not support a non-volatile cache.

Editor's Note 5: So far only SBC-2 specifically mentions caches and has a Caching mode page to control them. Other devices types might follow suit.

Editor's Note 6: There was a request to add a bit indicating if the device supports redundant caches. Software might have more confidence in them if set to one. However, that might require dual log page sets for NV capacity with cache identifiers. And, it would be hard to standardize what "redundant" means - could be 2, 3, or n copies.
