**From: Gerry Houlder, Seagate Technology** <gerry.houlder@seagate.com>
**Subj: SPC-4 Read debug log proposal**
**Date: Nov. 7, 2006**

---

This document proposes a standard method to control and retrieve debug data related to device errors. Several drive vendors (including Seagate) have proprietary methods for controlling these operations today, but customers have asked that a standard method be defined so their processes can be simpler.

Black text indicates current wording in the SPC-4 rev. 5a draft standard. Blue underlined text is new.

Rev. 1 changes requested at 9/13 meeting:

1. Rename “application log” as “debug log”;
2. Added resumption of debug log updates when resets occur;
3. Add statement after resume that “debug log may include data captured while log update was suspended”;
4. Added CLR_SUP bit in READ BUFFER Retrieve Debug Log data;
5. In table new2, increase Maximum Available Length to 4 bytes;
6. Add statement to Write Buffer defining action when debug log update is suspended;

Rev. 2 has editorial changes in 5.11.3.

## 1. Addition to model section of SPC-4:

**5.11 Setting and retrieving debug logs**

**5.11.1 Debug logs overview**

Debug logs are vendor specific data that has been collected by a SCSI device to aid in troubleshooting device errors. The WRITE BUFFER command (see 6.36.14) provides a method of inserting application client log information into debug logs or clearing the debug logs. The READ BUFFER command (see 6.14.9) provides a method of retrieving debug logs from the SCSI device.

**5.11.2 Application client logging**

Application client logging is a method the application client may use to store application client detected error information in a logical unit’s non-volatile storage (see 6.36.14). The information the application client sends to the logical unit is appended to a debug log. The application client error information may be recovered as part of the debug logs (see 5.11.3) or by means outside the scope of this standard and is not used for any logical unit related error recovery.

**Deleted:** is

**Formatted**

A log that contains a mix of application client error information and logical unit error information may be used to correlate an application client error with any errors internal to the logical unit. This provides a vendor independent way of correlating error logs.

**Deleted:** does not replace the vendor specific methods for collecting and analyzing engineering data, but

Application clients should minimize the amount of error information that is requested to be logged to prevent log overflows.

### 5.11.3 Retrieving debug logs

Device servers may require that debug logs be retrieved using a sequence of READ BUFFER commands. If the required command sequence is not received, a READ BUFFER command that is received in the incorrect sequence shall be terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to COMMAND SEQUENCE ERROR.

All device servers shall support at least retrieving debug logs in the following sequence:

1) Transfer the table of entries that defines the available debug log buffers by issuing a READ BUFFER command with MODE field set to retrieve debug log, BUFFER ID field set to 00h, BUFFER OFFSET field set to zero, and ALLOCATION LENGTH field set large enough to transfer the entire table of entries. As part of processing this READ BUFFER command the device server shall suspend updating the error history data;
2) Retrieve the buffers in any order by issuing the READ BUFFER command with the BUFFER ID value set to any supported value between 01h and FEh. A buffer may be retrieved using a single READ BUFFER command or multiple READ BUFFER commands. Device servers shall support the retrieval of debug logs using multiple READ BUFFER commands in the following sequence:
    1) Issue the initial READ BUFFER command with the BUFFER ID field set to the desired buffer value, the BUFFER OFFSET field set to zero, and the ALLOCATION LENGTH field set to the desired transfer size;
    2) After GOOD status has been returned for the previous READ BUFFER command and the number of bytes transferred equals the allocation length, issue another READ BUFFER command with BUFFER ID field set to the previous buffer value, the BUFFER OFFSET field set to the previous buffer offset plus the previous allocation length value, and the ALLOCATION LENGTH field set to the previous allocation length value;
    3) Repeat step 2) 2) until the number of bytes transferred is less than the allocation length value or a command sequence error occurs.
3) Repeat step 2) until the application client has transferred all of the required buffers.

The device server shall resume updating debug logs when:

a) the application client issues a READ BUFFER command with MODE field set to retrieve debug log, BUFFER ID field set to FFh, BUFFER OFFSET field set to zero, and ALLOCATION LENGTH set to zero;
b) a vendor specific timer expires;
c) a power cycle occurs;
d) a hard reset occurs;
e) an I_T nexus loss occurs; or
f) a logical unit reset occurs.

When updating of debug logs is resumed, the logs may include data captured while debug log updating was suspended.

Debug logs may also be retrieved by vendor specific methods that are outside the scope of this standard.

### 5.11.4 Interpreting the debug logs

The debug logs are interpreted using a vendor specific parsing application. Any other use of the debug logs is undefined.

### 5.11.5 Clearing the debug logs

The debug logs may be cleared by issuing a WRITE BUFFER command (see 6.36.14) with BUFFER ID field set to download debug log, BUFFER OFFSET field set to zero, parameter list length set to 00001Ah, and parameter data with the CLR bit set and the other fields set to zero.

## 2. Additions to READ BUFFER command

### 6.14.1 READ BUFFER command introduction

The READ BUFFER command (see table 126) is used in conjunction with the WRITE BUFFER command as a diagnostic function for testing memory in the SCSI device and the integrity of the service delivery subsystem. This command shall not alter the medium.

**Table 126 – READ BUFFER command**

| Bit / Byte | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (3Ch) | | | | | | | |
| 1 | RESERVED | | | MODE | | | | |
| 2 | BUFFER ID | | | | | | | |
| 3 | (MSB) | BUFFER OFFSET | | | | | | |
| 5 | | | | | | | | (LSB) |
| 6 | (MSB) | ALLOCATION LENGTH | | | | | | |
| 8 | | | | | | | | (LSB) |
| 9 | CONTROL | | | | | | | |

The function of this command and the meaning of fields within the CDB depend on the contents of the MODE field. The MODE field is defined in table 127.

**Table 127 – READ BUFFER MODE field**

| MODE | Description | Reference |
|---|---|---|
| 00h | Combined header and data [a] | 6.14.2 |
| 01h | Vendor Specific [a] | 6.14.3 |
| 02h | Data | 6.14.4 |
| 03h | Descriptor | 6.14.5 |
| 0Ah | Echo buffer | 6.14.6 |
| 0Bh | Echo buffer descriptor | 6.14.7 |
| 1Ah | Enable expander communications protocol and echo buffer | 6.14.8 |
| 1Ch | Retrieve debug log | 6.14.9 |
| 04h - 09h | Reserved | |
| 0Ch –19h | Reserved | |
| 1Bh | Reserved | |
| 1Dh – 1Fh | Reserved | |
| [a] Modes 00h and 01h are not recommended. | | |

**[Editors Note: clauses 6.14.2 through 6.14.8 are unchanged]**

### 6.14.9 Retrieve debug log mode (1Ch)

#### 6.14.9.1 Retrieve debug log overview

In this mode, the Data-In Buffer contains either a table of entries that describes the supported buffers (see 6.14.9.2) or vendor specific debug data (see 6.14.9.3). The BUFFER ID field specifies which part of the debug data shall be transferred. If the BUFFER ID field is set to an unsupported buffer, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The BUFFER OFFSET field contains the byte offset within the specified buffer from which data shall be transferred. The application client shall conform to the offset boundary requirements returned in the READ BUFFER descriptor (see 6.14.5). If the device server is unable to accept the specified buffer offset, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

#### 6.14.9.2 Table of entries data description

When the BUFFER ID field is set to zero, a table of entries shall be transferred (table new1).

**Table new1 – Table of entries format**

| Bit / Byte | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | MANUFACTURER IDENTIFICATION | | | | | | | |
| | | | | | | | | |
| 7 | | | | | | | | |
| 8 | VERSION | | | | | | | |
| 9 | RESERVED | | | | | | | CLR_SUP |
| 10 | RESERVED | | | | | | | |
| 11 | RESERVED | | | | | | | |
| 12 | RESERVED | | | | | | | |
| 13 | RESERVED | | | | | | | |
| 14 | (MSB) DATA LENGTH (n-15) | | | | | | | |
| 15 | | | | | | | | (LSB) |
| 16 | First table entry | | | | | | | |
| | | | | | | | | |
| 23 | | | | | | | | |
| | : | | | | | | | |
| | : | | | | | | | |
| | : | | | | | | | |
| n-7 | Last table entry | | | | | | | |
| | | | | | | | | |
| n | | | | | | | | |

The MANUFACTURER IDENTIFICATION field contains eight bytes of left-aligned ASCII data (see 4.4.1) identifying the manufacturer of the product. The T10 vendor identification shall be one assigned by INCITS. A list of assigned T10 vendor identifications is in Annex E and on the T10 web site (http://www.T10.org).

The VERSION field indicates a vendor specific version.

The Clear Support (CLR_SUP) bit set to one indicates that the CLR function (see 6.36.14) is supported. The CLR_SUP bit set to zero indicates that the CLR function is not supported.

The DATA LENGTH field indicates the number of table entry bytes available to be transferred. This value shall not be altered even if the allocation length is not sufficient to transfer all of the available bytes.

Each table entry shall be 8 bytes as defined in table new2. There shall be an entry for each supported buffer ID. The first entry shall be for buffer ID zero and the entries shall be in order of ascending buffer IDs. The supported buffer IDs are not required to be contiguous.

There shall not be an entry for buffer ID of FFh because there shall be no data associated with this value.

**Table new2 – Table entry format**

| Bit / Byte | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | SUPPORTED BUFFER ID | | | | | | | |
| 1 | RESERVED | | | | | | | |
| 2 | RESERVED | | | | | | | |
| 3 | | | | | | | | |
| 4 | (MSB) | | | | | | | |
| 5 | MAXIMUM AVAILABLE LENGTH | | | | | | | |
| 6 | | | | | | | | |
| 7 | | | | | | | | (LSB) |

The SUPPORTED BUFFER ID field indicates the debug log buffer ID associated with this table entry.

The MAXIMUM AVAILABLE LENGTH field indicates the maximum number of data bytes contained in this debug log buffer. The actual number of bytes available for transfer may be smaller.

**6.14.9.3 Debug log data description**

When the BUFFER ID field is from 01h to FEh, vendor specific debug data is transferred.

When the BUFFER ID field is FFh, no data shall be available to be transferred.

## 3.0 Additions to WRITE BUFFER command

### 6.36.1 WRITE BUFFER command introduction

The WRITE BUFFER command (see table 195) is used in conjunction with the READ BUFFER command as a diagnostic function for testing logical unit memory in the SCSI target device and the integrity of the service delivery subsystem. Additional modes are provided for:

a) Downloading microcode;
b) Downloading and saving microcode;
c) Downloading microcode with deferred activation; and
d) Downloading debug logs (see 5.11).

**Table 195 – WRITE BUFFER command**

| Bit / Byte | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (3Bh) | | | | | | | |
| 1 | RESERVED | | | MODE | | | | |
| 2 | BUFFER ID | | | | | | | |
| 3 | (MSB) | BUFFER OFFSET | | | | | | |
| 5 | | | | | | | | (LSB) |
| 6 | (MSB) | PARAMETER LIST LENGTH | | | | | | |
| 8 | | | | | | | | (LSB) |
| 9 | CONTROL | | | | | | | |

The command shall not alter any medium of the logical unit when the data mode or the combined header and data mode is specified.

The function of this command and the meaning of fields within the CDB depend on the contents of the MODE field. The MODE field is defined in table 196.

**Table 196 – WRITE BUFFER MODE field**

| MODE | Description | Reference |
|---|---|---|
| 00h | Combined header and data [a] | 6.36.2 |
| 01h | Vendor Specific [a] | 6.36.3 |
| 02h | Data | 6.36.4 |
| 04h | Download microcode | 6.36.5 |
| 05h | Download microcode and save | 6.36.6 |
| 06h | Download microcode with offsets [b] | 6.36.7 |
| 07h | Download microcode with offsets and save [b] | 6.36.8 |
| 0Ah | Echo buffer | 6.36.9 |
| 0Eh | Download microcode with offsets and defer activation [b] | 6.36.10 |
| 0Fh | Activate deferred microcode | 6.36.11 |
| 1Ah | Enable expander communications protocol and echo buffer | 6.36.12 |
| 1Bh | Disable expander communications protocol | 6.36.13 |
| 1Ch | Download debug log | 6.36.14 |
| 03h | Reserved | |
| 08h - 09h | Reserved | |
| 0Bh –0Dh | Reserved | |
| 10h – 19h | Reserved | |
| 1Dh – 1Fh | Reserved | |

[a] Modes 00h and 01h are not recommended.
[b] When downloading microcode with buffer offsets, the WRITE BUFFER command mode should be 06h, 07h, or 0Eh.

**[Editors Note: clauses 6.36.2 through 6.36.13 are unchanged]**

### 6.36.14 Download debug log mode (1Ch)

In this mode the device server transfers data from the application client and stores it in a debug log (see 5.11). The format of the debug log data is as specified in table 197. The BUFFER ID field and BUFFER OFFSET field are ignored in this mode.

Upon successful completion of a WRITE BUFFER command the data shall be appended to the debug log. The debug log updates requested in a WRITE BUFFER command are appended to the debug log even if debug log updating is suspended (see 5.11.3).

The PARAMETER LIST LENGTH field specifies the maximum number of bytes that shall be transferred from the Data-Out Buffer to be stored in the debug log. If the PARAMETER LIST LENGTH field specifies a transfer that exceeds the debug log's capacity, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

**Table 197 – Debug log data WRITE BUFFER format**

| Bit / Byte | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | | | | | | | |
| | T10 VENDOR IDENTIFICATION | | | | | | | |
| 7 | | | | | | | | (LSB) |
| 8 | (MSB) | | | | | | | |
| 9 | ERROR TYPE | | | | | | | (LSB) |
| 10 | RESERVED | | | | | | | CLR |
| 11 | RESERVED | | | | | | | |
| 12 | (MSB) | | | | | | | |
| | TIME STAMP | | | | | | | |
| 17 | | | | | | | | (LSB) |
| 18 | RESERVED | | | | | | | |
| 19 | RESERVED | | | | | | | |
| 20 | RESERVED | | | | CODE SET | | | |
| 21 | ERROR LOCATION FORMAT | | | | | | | |
| 22 | (MSB) | | | | | | | |
| 23 | ERROR LOCATION LENGTH (m-25) | | | | | | | (LSB) |
| 24 | (MSB) | | | | | | | |
| 25 | VENDOR SPECIFIC LENGTH (n-m) | | | | | | | (LSB) |
| 26 | (MSB) | | | | | | | |
| | ERROR LOCATION | | | | | | | |
| m | | | | | | | | (LSB) |
| m+1 | VENDOR SPECIFIC | | | | | | | |
| | | | | | | | | |
| n | | | | | | | | |

The T10 VENDOR IDENTIFICATION field contains eight bytes of left-aligned ASCII data (see 4.4.1) identifying the vendor of the product. The T10 vendor identification shall be one assigned by INCITS. A list of assigned T10 vendor identifications is in Annex E and on the T10 web site (http://www.T10.org).

The ERROR TYPE field (see table 198) specifies the error detected by the application client.

**Table 198 – ERROR TYPE field**

| CODE | Description |
|---|---|
| 0000h | No error specified by the application client |
| 0001h | An unknown error was detected by the application client |
| 0002h | The application client detected corrupted data |
| 0003h | The application client detected a permanent error |
| 0004h | The application client detected a service response of SERVICE DELIVERY OR TARGET FAILURE (SAM-3). |
| 0005h – 7FFFh | Reserved |
| 8000h –FFFFh | Vendor specific |

A CLR bit set to one specifies that the error history shall be cleared except for error history contents that are not allowed to be cleared by the application client. The other fields (see table 197) shall be ignored. A CLR bit set to zero specifies that the debug buffer contents shall be preserved.

The TIME STAMP field shall contain:

a) The number of milliseconds that have elapsed since midnight, 1 January 1970 UT (see 3.1.124); or
b) Zero, if the application client is not able to determine the UT of the log entry.

The CODE SET field (see table 199) specifies the code set used for the debug log information and shall only apply to information contained in the VENDOR SPECIFIC field.

NOTE 33 - The CODE SET field is intended to be an aid to software that displays the debug log information.

**Table 199 – CODE SET field**

| Code | Description |
|---|---|
| 0h | Reserved |
| 1h | The debug log information is binary |
| 2h | The debug log information is ASCII printable characters (i.e., code values 20h through 7Eh) |
| 3h | The debug log information is ISO/IEC 10646-1 (UTF-8) codes |
| 4h - Fh | Reserved |

The ERROR LOCATION FORMAT field (see table 200) specifies the format of the ERROR LOCATION field.

**Table 200 – ERROR LOCATION FORMAT field**

| Code | Description |
|---|---|
| 00h | No error specified by the application client |
| 01h | The error location field specifies the logical block (e.g., LBA) associated with the error information contained within the debug log. |
| 02h – 7Fh | Reserved |
| 80h - FFh | Vendor specific |

The ERROR LOCATION LENGTH field specifies the length of the ERROR LOCATION field. The ERROR LOCATION LENGTH field value shall be a multiple of four. An error location length value of zero specifies there is no error location information.

The VENDOR SPECIFIC LENGTH field specifies the length of the VENDOR SPECIFIC field. The VENDOR SPECIFIC LENGTH field value shall be a multiple of four. A vendor specific length value of zero specifies there is no vendor specific information.

The ERROR LOCATION field specifies the location at which the application client detected the error.

The VENDOR SPECIFIC field provides vendor specific information on the error.