Technical Note:
NVMe Basic Management Command

Revision 1.0
February 24, 2015

NVM Express Workgroup
c/o Virtual, Inc.
401 Edgewater Place, Suite 600
Wakefield, MA 01880
info@nvmexpress.org

The NVMe Management Interface (NVMe-MI) Workgroup is developing a specification using Management Component Transport Protocol (MCTP) messages. One command will not use MCTP and it is being released early by this technical note so that systems may already start polling their NVMe devices for basic health status information over SMBus.

This command does not provide any mechanism to modify or configure the NVMe device. Such features will use the more capable MCTP protocol rather than this command's simpler SMBus Block Read. The host can reuse existing SMBus or serial EEPROM read subroutines for this read and is not required to switch the SMBus between master and slave modes as in MCTP.

The block read protocol is specified by the SMBus specification which is available online at www.smbus.org. First slave address write and command code bytes are transmitted by the host, then a repeated start and finally a slave address read. The host keeps clocking as the drive then responds in slave mode with the selected data. The command code is used as a starting offset into the data block shown in Figure 1, like an address on a serial EEPROM.

The offset value increments on every byte read and is reset to zero on a stop condition. A read command without a repeated start is permissible and would always start transmission from offset zero. Reading more than the block length with an I2C read is also permissible and these reads would continue into the first byte in the next block of data. Note that the calculated PEC includes the host command and will be different depending on how the read starts.

Blocks of data are packed sequentially. The first 2 blocks are defined by the NVMe-MI workgroup. The first block is the dynamic host health data. The second block includes the Vendor ID (VID) and serial number of the drive. Additional blocks of data can be defined by the owner of the VID. Reading past the end of the vendor defined blocks shall return zeros.

The SMBus slave address to read this data structure is allowed to be the same address we use for MCTP, and defaults to 6Ah if ARP is not invoked. Mixed MCTP and block read traffic does not result in any packet corruption.

Here are a few example reads from an NVMe drive at 30°C, no alarms, VID=1234h, serial number is AZ123456. All values are in hexadecimal or ASCII for 'character'. Host transmissions are shown in black, and drive transmissions are shown in grey:

SMBus block read of drive status:

Start D4 00 Start D5 06 BF FF 1E 01 00 00 10 **Stop**

SMBus block read of static data:

Start D4 08 Start D5 16 12 34 'A' 'Z' '1' '2' '3' '4' '5' '6' 20 20 20 20 20 20 20 20 20 20 20 20 DA **Stop**

SMBus send byte to reset Arbitration bit:

Start D4 FF Stop

I2C read of status and vendor content, I2C allows reading across SMBus block boundaries:

Start D4 00 Start D5 06 BF FF 1E 01 00 00 10 16 12 34 'A' 'Z' '1' '2' '3' '4' '5' '6' 20 20 20 20 20 20 20 20 20 20 20 20 B0 **Stop**

The SMBus Arbitration bit may be used for simple arbitration on systems that have multiple drives on the same SMBus segment without ARP or muxes to separate them. To use this mechanism, the host follows this 3 step process to handle collisions for the same slave address:

1. The host does a SMBus byte write to send byte FFh which clears the SMBus Arbitration bit on all listening NVMe Management Endpoints at this slave address.
2. The host does an I2C read starting from offset 0h and continuing at least through the serial number in the second block. The drive transmitting a '0' when other drives sent a '1' eventually wins arbitration and sets the arbitration bit to '1' to give other drives priority on the next read.
3. Repeat step 2 until all drives are read, reading the Arbitration bit as a '1' indicates loop is done.
4. Sort the responses by serial number since the order of drive responses depends partially on health status and temperatures.

Be careful that there are no short reads of similar data between steps 1 and 3. If the read data is the same on multiple drives then all these drives will set the arbitration bit. After that a new send byte FFh is required to restart the process.

The logic levels were intentionally set to normally high in the 2nd and 3rd bytes. This is an additional mechanism to assist systems that do not have ARP or muxes. Since '0' bits win arbitration on SMBus, a drive with an alarm condition will be prioritized over healthy drives in the above arbitration scheme.

Command Code	Offset	Description
0	00	**Length of Status:** Indicates number of additional bytes to read before encountering PEC. This value should always be 6 in implementations of this version of the spec.
	01	**Status Flags (SFLGS):** This field indicates the status of the NVM subsystem. **SMBus Arbitration** – Bit 7 is set '1' after a SMBus block read is completed all the way to the stop bit without bus contention and cleared to '0' if a SMBus Send Byte FFh is received on this SMBus slave address. **Powered Up** – Bit 6 is set to '1' when the subsystem cannot process NVMe management commands, and the rest of the transmission may be invalid. If cleared to '0', then the NVM subsystem is fully powered and ready to respond to management commands. This logic level intentionally identifies and prioritizes powered up and ready drives over their powered off neighbors on the same SMBus segment. **Drive Functional** – Bit 5 is set to '1' to indicate an NVM subsystem is functional. If cleared to '0', then there is an unrecoverable failure in the NVM subsystem and the rest of the transmission may be invalid. **Reset Not Required** - Bit 4 is set to '1' to indicate the NVM subsystem does not need a reset to resume normal operation. If cleared to '0' then the NVM subsystem has experienced an error that prevents continued normal operation. A controller reset is required to resume normal operation. **PCIe Link Active** - Bit 3 is set to '1' to indicate one or more of the PCIe links is up (i.e., the Data Link Control and Management State Machine is in the DL_Active state). If cleared to '0', then all the PCIe links are down. Bits 2-0 shall be set to '1'.
	02	**SMART Warnings:** This field shall contain the Critical Warning field (byte 0) of the NVMe SMART / Health Information log. Each bit in this field shall be inverted from the NVMe definition (i.e., the management interface shall indicate a '0' value while the corresponding bit is '1' in the log page). Refer to the NVMe specification for bit definitions. If there are multiple controllers in the NVM subsystem, the management endpoint shall combine the Critical Warning field from every controller such that a bit in this field is: • Cleared to '0' if any controller in the subsystem indicates a critical warning for that corresponding bit. • Set to '1' if all controllers in the NVM subsystem do not indicate a critical warning for the corresponding bit.

<table>
<tr><td></td><td>03</td><td>Composite Temperature (CTemp): This field indicates the current temperature in degrees Celsius. If a temperature value is reported, it should be the same temperature as the Composite Temperature from the SMART log of hottest controller in the NVM subsystem. The reported temperature range is vendor specific, and shall not exceed the range -60 to +127°C. The 8 bit format of the data is shown below.

This field should not report a temperature when that is older than 5 seconds. If recent data is not available, the NVMe management endpoint should indicate a value of 80h for this field.

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>00h-7Eh</td><td>Temperature is measured in degrees Celsius (0 to 126C)</td></tr>
<tr><td>7Fh</td><td>127C or higher</td></tr>
<tr><td>80h</td><td>No temperature data or temperature data is more the 5 seconds old.</td></tr>
<tr><td>81h</td><td>Temperature sensor failure</td></tr>
<tr><td>82h-C3h</td><td>Reserved</td></tr>
<tr><td>C4</td><td>Temperature is -60C or lower</td></tr>
<tr><td>C5-FFh</td><td>Temperature measured in degrees Celsius is represented in twos complement (-1 to -59C)</td></tr>
</table></td></tr>
<tr><td></td><td>04</td><td>Percentage Drive Life Used (PDLU): Contains a vendor specific estimate of the percentage of NVM subsystem NVM life used based on the actual usage and the manufacturer’s prediction of NVM life. If an NVM subsystem has multiple controllers the highest value is returned. A value of 100 indicates that the estimated endurance of the NVM in the NVM subsystem has been consumed, but may not indicate an NVM subsystem failure. The value is allowed to exceed 100. Percentages greater than 254 shall be represented as 255. This value should be updated once per power-on hour and equal the Percentage Used value in the NVMe SMART Health Log Page.</td></tr>
<tr><td></td><td>06:05</td><td>Reserved: Shall be set to 0000h</td></tr>
<tr><td></td><td>07</td><td>PEC: An 8 bit CRC calculated over the slave address, command code, second slave address and returned data. Algorithm is in SMBus Specifications.</td></tr>
<tr><td>8</td><td>08</td><td>Length of identification: shall always be 22 in implementations of this version of the spec</td></tr>
<tr><td></td><td>10:09</td><td>Vendor ID: The 2 byte vendor ID, assigned by the PCI SIG. Should match VID in the Identify Controller command response. MSB is transmitted first.</td></tr>
<tr><td></td><td>30:11</td><td>Serial Number: 20 characters that match the serial number in the NVMe Identify Controller command response. First character is transmitted first</td></tr>
<tr><td></td><td>31</td><td>PEC: An 8 bit CRC calculated over the slave address, command code, second slave address and returned data. Algorithm is in SMBus Specifications.</td></tr>
<tr><td>32+</td><td>255:32</td><td>Vendor Specific – This data structure shall not exceed the maximum read length of 255 specified in the SMBus version 3 specification. Preferably length is not greater than 32 for compatibility with SMBus 2.0, additional blocks shall be on 8 byte boundaries.</td></tr>
</table>

Figure 1: Subsystem Management Data Structure