

NVDIMM DSM Interface Example

April 2015



Notices

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others

© 2015 Intel Corporation.



Contents

Contents

1	Introduction	5
1.1	Document Scope	5
1.2	Related Documents	5
1.3	Terminology	5
2	NVDIMM Device Specific Method (DSM)	6
3	_DSM Interface for NVDIMM Root Device - Example	7
3.1	Address Range Scrubbing	7
3.1.1	Query ARS Capabilities (Function Index 1)	8
3.1.2	Start ARS (Function Index 2)	8
3.1.3	Query ARS Status (Function Index 3)	9
4	_DSM Interface for NVDIMM Device (non-root) - Example	13
4.1	SMART and Health Info (Function Index 1)	14
4.1.1	Input (Arg3)	14
4.1.2	Output	14
4.2	Get SMART Threshold (Function Index 2)	15
4.2.1	Input (Arg3)	15
4.2.2	Output	16
4.3	Get Block NVDIMM Flags (Function Index 3)	17
4.3.1	Input (Arg3)	17
4.3.2	Output	17
4.4	Get Namespace Label Size (Function Index 4)	18
4.4.1	Input (Arg3)	18
4.4.2	Output	18
4.5	Get Namespace Label Data (Function Index 5)	19
4.5.1	Input (Arg3)	19
4.5.2	Output	19
4.6	Set Namespace Label Data (Function Index 6)	20
4.6.1	Input (Arg3)	20
4.6.2	Output	21
4.7	Get Vendor-Specific Command Effect Log Size (Function Index 7)	21
4.7.1	Input (Arg3)	21
4.7.2	Output	21
4.8	Get Vendor-Specific Command Effect Log (Function Index 8)	22
4.8.1	Input (Arg3)	22
4.8.2	Output	22
4.9	Vendor-Specific Command (Function Index 9)	25
4.9.1	Input (Arg3)	25



4.9.2 Output25

Figures

Figure 3-1 Format of ARS Output buffer 11
Figure 3-2 Format of ARS Error Record 12

Tables

Table 1 – Terminology 5
Table 3-1 Query ARS Capabilities – Input Format 8
Table 3-2 Query ARS Capabilities – Output Format 8
Table 3-3 Start ARS – Input 9
Table 3-4 Start ARS – Output Format 9
Table 3-5 Query ARS Status – Output Format 10
Table 3-6 Fields of ARS Output Buffer Format 11
Table 3-7 ARS Error Record Format 12
Table 4-1 SMART and Health Info – Output Format 14
Table 4-2 SMART and Health Data Format 14
Table 4-3 Get SMART Threshold – Output Format 16
Table 4-4 SMART Threshold Data Format 16
Table 4-5 Get Block NVDIMM Flags - Output Format 17
Table 4-6 Get Namespace Label Size – Output Format 18
Table 4-7 Get Namespace Label Data – Input Format 19
Table 4-8 Get Namespace Label Data – Output Format 19
Table 4-9 Set Namespace Label Data – Input Format 20
Table 4-10 Set Namespace Label Data – Output Format 21
Table 4-11 Get Vendor Specific Command Effect Log Size – Output Format 21
Table 4-12 Get Vendor Specific Command Effect Log Size – Output Format 22
Table 4-13 Command Effect Data - Format 23
Table 4-14 Vendor Specific Command – Input Format 25
Table 4-15 Vendor Specific Command – Output Format 25



1 Introduction

1.1 Document Scope

This document is targeted to writers of BIOS and OS drivers for NVDIMMs whose design adheres to the NFIT Tables in the ACPI V6.0 specification. This document specifically discusses the NVDIMM Device Specific Method (_DSM) example.

1.2 Related Documents

The related documents are ACPI Specification Version 6.0 (<http://www.uefi.org/specifications>) and NVDIMM Namespace Specification (<http://pmem.io/documents>).

1.3 Terminology

Refer to Table 1 for definitions of terms used in this document.

Table 1 – Terminology

Term	Description
NFIT	The NVDIMM Firmware Interface Table defines the ACPI-like information created by the BIOS to inform the OS about NVDIMMs in the system.
NVDIMM	Non-volatile memory in a DIMM form factor.
NVDIMM Namespace Label	Labels, stored at a known location on NVDIMMs, which define the DIMM's contribution to NVDIMM Namespaces. This is a software mechanism; the DIMM itself just sees the labels as part of the overall data stored on the DIMM.
NVDIMM Namespace	Similar to an NVMe Namespace or a Logical Unit (LUN) on a SCSI disk, this is a software mechanism for managing ranges of persistence on NVDIMMs.
Persistent Memory	Byte-addressable memory that retains its contents after power loss.
SPA	System Physical Address. A physical address on the host operating system.



2 NVDIMM Device Specific Method (DSM)

ACPI defines an NVDIMM root device under `_SB` scope with a `_HID` of “ACPI0012”. The NVDIMM child devices under the NVDIMM root device are defined with `_ADR` corresponding to the NFIT device handle. The NVDIMM root device and the NVDIMM devices can have device specific methods (`_DSM`) to provide additional functions specific to a particular NVDIMM implementation.

An example name space is shown below for a platform containing one NVDIMM:

```
Scope (\_SB){
  Device (NVDR) // Root device
  {
    Name (_HID, "ACPI0012")
    Method (_STA) {...}
    Method (_FIT) {...}
    Method (_DSM, ...) {...}

    Device (NVD)
    {
      Name(_ADR, h) //where h is NFIT Device Handle for this NVDIMM
      Method (_DSM, ...) {...}
    }
  }
}
```

The chapter 2 in this document describes an example `_DSM` interface for NVDIMM Root Device and the chapter 3 in this document describes an example `_DSM` interface for NVDIMM Device with Region Format Interface Code (RFIC) of 0x0201.



3 ***_DSM Interface for NVDIMM Root Device - Example***

This chapter describes the device specific method (*_DSM*) for NVDIMM root devices is described in this section. Note that the *_DSM* methods defined in this section are required to be implemented only under NVDIMM object with *_HID* of ACPI0012.

Arg0 – *UUID* {set to 2f10e7a4-9e91-11e4-89d3-123b93f75cba}

Arg1 – *Revision ID* (set to 1)

Arg2 – *Function Index*

0 – Query command implemented per ACPI Specification

1 – Query Address Range Scrub (ARS) Capabilities

2 – Start Address Range Scrub (ARS)

3 – Query Address Range Scrub (ARS) Status

Arg3 – A package containing parameters for the function specified by the *UUID*, *Revision ID*, and *Function Index*. The layout of the package for each command along with the corresponding output are illustrated in the respective *Function Index* description sections. The input and output package are a lists of bytes (Buffer).

Implementation Note: This section adopts the following conventions for the *_DSM* function return status codes:

Bytes[1-0]

0 – Success

1 – Not Supported

2 – Invalid Input Parameters

3 – Function-Specific Error Code

4 - FFFFh Reserved

Bytes[3-2] Extended Status Field (Function Specific Status Code on Success)

3.1 Address Range Scrubbing

Address Range Scrubbing (ARS) allows platform to communicate persistent memory errors to system software. This capability allows system software to avoid accessing addresses with uncorrectable errors in persistent memory.

The address range scrubbing command is accessed via *_DSM* interface. The ARS command is system scope and is not specific to a single NVDIMM, i.e., it returns the locations detected to be in error for all the NVDIMMs present in the system.



3.1.1 Query ARS Capabilities (Function Index 1)

This function provides ARS capabilities for a given address range. Its input (Arg3) and the output parameter format are as follows:

3.1.1.1.1 Input (Arg3)

Input is a package containing a single buffer, where the buffer is formatted as shown in Table 3-1.

Table 3-1 Query ARS Capabilities – Input Format

Field	Byte Length	Byte Offset	Description
ARS Start SPA Address	8	0	In bytes
ARS Length	8	8	In bytes

3.1.1.1.2 Output

The return value for this function is a buffer formatted as shown in Table 3-2.

Table 3-2 Query ARS Capabilities – Output Format

Field	Byte Length	Byte Offset	Description
Status	4	0	Bytes[1-0] 0 – Success 1 – Not Supported (ARS is not supported in this platform) Bytes[3-2] Extended Status Field on Success Bit[0] – If set to 1, indicates Volatile Memory Scrub is supported Bit[1] – If set to 1, Persistent Memory Scrub is supported Bits[15:2] – Reserved Platforms that are not capable of ARS are still required to implement the command and return “Not Supported”.
Max ARS Data Size	4	4	In bytes, Maximum size of the ARS data buffer supported by the platform

3.1.2 Start ARS (Function Index 2)

The Start ARS function triggers an Address Range Scrub for the given range of memory. Address scrubbing can be done for volatile memory, persistent memory, or both. Only one scrub



NVDIMM DSM Interface Example

can be in progress system wide at any given time. You must first issue a Query ARS Status command and ensure no ARS is in progress before issuing a Start ARS function. When an address range scrub operation is started, the previous ARS data is lost. You must call the Query ARS Status function to retrieve any existing ARS data before calling the Start ARS function.

3.1.2.1 Input (Arg3)

Input is a package containing a single buffer, where the buffer is formatted as shown in Table 3-3.

Table 3-3 Start ARS – Input

Field	Byte Length	Byte Offset	Description
ARS Start SPA Address	8	0	In bytes
ARS Length	8	8	In bytes
Type	2	16	Bit[0] – If set to 1, Scrub Volatile Memory Bit[1] – If set to 1, Scrub Persistent Memory Bits[15:2] Reserved
Reserved	6	18	

3.1.2.2 Output

The return value for this function is a buffer formatted as shown in Table 3-4.

Table 3-4 Start ARS – Output Format

Field	Byte Length	Byte Offset	Description
Status	4	0	Bytes[1-0] 0 – Success (ARS Started) 1 – Not Supported (ARS is not supported in this platform) 2 – Invalid Input Parameters (address out of range or no such type in specified range) 3 – ARS already in progress Platforms that are not capable of ARS are still required to implement this function and return “ARS Not Supported”.

3.1.3 Query ARS Status (Function Index 3)

Only one Address Range Scrub is allowed system wide at any given time. The *Query ARS Status* command allows software to get the status of ARS scrub in the system.



3.1.3.1 Input (Arg3)

None

3.1.3.2 Output

The return value for this function is a buffer formatted as shown in Table 3-5.

Table 3-5 Query ARS Status – Output Format

Field	Byte Length	Byte Offset	Description
Status	4	0	Bytes[1-0] 0 – Success 1 – Not Supported (ARS is not supported in this platform) Bytes[3-2] Extended Status Field (valid if Byte[1-0] is Success) 0 – ARS complete 1 – ARS in progress 2 – No ARS performed for current boot.
ARS Data	Varies	4	Format of the output buffer is illustrated in Figure 3-1 and the fields are detailed in Table 3-6.



NVDIMM DSM Interface Example

The output SPA range return indicates the scope of the ARS scrub for the specified type.

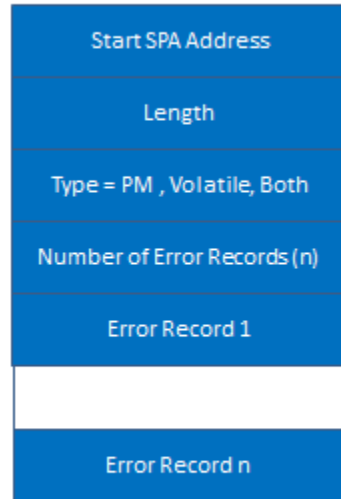


Figure 3-1 Format of ARS Output buffer

Table 3-6 Fields of ARS Output Buffer Format

Field	Byte Length	Byte Offset	Interpretation
Output (Size)	4	0	Size of Output in bytes. If extended status indicates that ARS is still in progress, size returned is 0.
Start SPA	8	4	In bytes
Length	8	12	In bytes ARS performed range is from Start SPA to Start SPA + Length
Type	2	20	Bit[0] – Volatile Memory range if set to 1 Bit[1] – Persistent Memory range if set to 1 If both bit[0] and bit[1] are set, both Persistent Memory and volatile memory are in this range. Bits[15:2] – Reserved
Reserved	2	22	
Number of Error Records	4	24	Number of ARS Error Record structures reported
ARS Error Records	Varies	28	Refer to Table 3-7 for the format of the error record. Format of the output buffer is illustrated in Figure 3-2



			and the fields are detailed in Table 3-7.
--	--	--	---

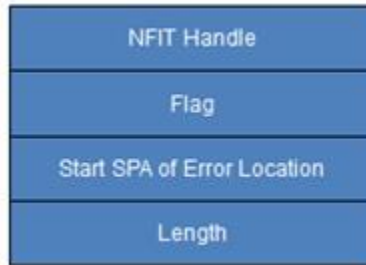


Figure 3-2 Format of ARS Error Record

Table 3-7 ARS Error Record Format

Field	Byte Length	Byte Offset	Interpretation
NFIT Handle	4	0	NFIT Handle indicates the specific NVDIMM that is the source of the error for the address range covered by SPA Start Address and Length.
Flags	4	4	Bit[0] – If set to 1, indicates an overflow condition. The overflow condition is set when either platform or NVDIMM unable to determine all the errors in the input address range. Bits[31:1] Reserved
Start SPA of Error Location	8	8	Start of System Physical Address of the error.
Length	8	16	Length indicates the consecutive bytes from <i>Start SPA of Error Location</i> that are in error. Due to interleaving, the range covered by <i>Start SPA of Error Location</i> and <i>Length</i> may include addresses that are present in other NVDIMMs in an interleave set. In case of overflow, the address range indicated by <i>Start SPA of Error Location</i> and <i>Length</i> will cover the NVDIMM interleave set that is impacted by the error. The range covered by <i>Start SPA of Error Location</i> and <i>Length</i> may exceed the requested scrub range due to platform limitations.



4 ***_DSM Interface for NVDIMM Device (non-root) - Example***

Platforms that have the `_DSM` interface implemented, as outlined in this section, can support a NVDIMM region with Region Format Interface Code (RFIC) of 0x0201.

Note that the `_DSM` methods defined in this section are required to be implemented under NVDIMM devices that are child devices of NVDIMM objects associated with `_HID` of ACPI0012 in ACPI name space hierarchy.

Arg0 – UUID (set to 4309AC30-0D11-11E4-9191-0800200C9A66)

Arg1 – Revision ID (set to 1)

Arg2 – Function Index

0 – Query command implemented per ACPI Specification

1 – SMART and Health Info

2 – Get SMART Threshold

3 – Get Block NVDIMM Flags

4 – Get Namespace Label Size

5 – Get Namespace Label Data

6 – Set Namespace Label Data

7 – Get Vendor-Specific Command Effect Log Size

8 – Get Vendor-Specific Command Effect Log

9 – Vendor-Specific Command

Arg3 – A package containing parameters for the function specified by the `UUID`, `Revision ID`, and `Function Index`. The layout of the package for each command along with the corresponding output are illustrated in the respective `Function Index` description sections. The input and output package are a list of bytes (Buffer).

Implementation Note: This section adopts the following conventions for the `_DSM` function return status codes:

Bytes[1-0]

0 – Success

1 – Not Supported

2 – Non-Existing Memory Device

3 – Invalid Input Parameters

4 – Vendor Specific Error (details in Extended Status Field)

5-FFFFh Reserved

Bytes[3-2] Extended Status Field (Vendor defined)



4.1 SMART and Health Info (Function Index 1)

This function provides information for the SMART and Health function.

4.1.1 Input (Arg3)

None

4.1.2 Output

The return value for this function is a buffer formatted as shown in Table 4-1.

Table 4-1 SMART and Health Info – Output Format

Field	Byte Length	Byte Offset	Description
Status	4	0	Bytes[1-0] 0 – Success 1 – Not Supported 2 – Non-Existing Memory Device 3 - Reserved 4 – Vendor-specific Error(details in Extended Status Field) 5-FFFFh Reserved Bytes[3-2] Extended Status Field (Vendor Defined)
SMART and Health Data	128	4	Output formatted as shown in Table 4-2.

Table 4-2 SMART and Health Data Format

Bytes	Description
03-00	Validation Flags – if the corresponding validation flag is not set in this field, it is indication to software that the corresponding field is not valid and must not be interpreted. Bit[0] – if set to 1, indicates that Health Status field is valid Bit[1] – if set to 1, indicates that Current Temperature field is valid Bit[2] – if set to 1, indicates that Spare Blocks field is valid Bit[3] – if set to 1, indicates that Alarm Trips field is valid Bit[4] – if set to 1, indicates that Percentage Used field is valid Bit[5] – if set to 1, indicates that Last Shutdown Status field is valid Bit[6] – if set to 1, indicates that Size of Vendor-specific Data field is valid. If this field is not valid, the software will ignore the vendor-specific data fields.



NVDIMM DSM Interface Example

	Bits[31:7] – Reserved
07-04	Reserved
08	Health Status (HS): Overall health summary Bit[0] – if set to 1, indicates Non-Critical condition, maintenance required but no data loss detected Bit[1] – if set to 1, indicates Critical condition, features or performance degraded due to failures but no data loss detected Bit[2] – if set to 1, indicates fatal condition, data loss is detected or is imminent Bits[7:3] - Reserved
10-09	Current Temperature: Current temperature of the NVDIMM device Bits[14:0] - Temperature in 1/16 th Celsius resolution. Bit[15] – Sign bit for temperature (1 = negative, 0 = positive)
11	Spare Blocks: Remaining Spare Capacity as % of factory configured space Valid range 0 to 100.
12	Alarm Trips: Bits to signify if values have tripped their respective alarm thresholds Bit[0] – Temperature Trip - If set, then the temperature value has reached the pre-programmed threshold limit Bit[1] – Spare Blocks Trip - If set, then the spare block value has reached the pre-programmed threshold limit Bits[7:2] - Reserved
13	Percentage Used: Device life span as percentage, 100 = the warranted life span of the device has been reached
14	Last Shutdown Status: status of last shutdown 0 – Clean shutdown 1 - OFFh – Not Clean Shutdown, indicates that there was either a platform or memory device-related failure occurred when saving data targeted for this memory device.
15	Reserved
19-16	Size of Vendor-specific Data. If set to 0, indicates that there is no vendor specific data that follows. Otherwise, indicates size of the Vendor-specific data that follows.
127-20	Vendor-specific Data

4.2 Get SMART Threshold (Function Index 2)

This function provides SMART related threshold information.

4.2.1 Input (Arg3)

None



4.2.2 Output

The return value for this function is a buffer formatted as shown in Table 4-3.

Table 4-3 Get SMART Threshold – Output Format

Field	Byte Length	Byte Offset	Description
Status	4	0	Bytes[1-0] 0 – Success 1 – Not Supported 2 – Non-Existing Memory Device 3 – Reserved 4 – Vendor-specific Error(details in Extended Status Field) 5–FFFFh Reserved Bytes[3-2] Extended Status Field (Vendor-defined)
SMART Threshold Data	8	4	Output formatted as shown in Table 4-4.

Table 4-4 SMART Threshold Data Format

Bytes	Description
1-0	Threshold Alarm Control – If this bit is set to 1, the specific alarm is enabled and the corresponding Alarm Trip bit in the SMART Health Status output payload will be set when a specific threshold outlined below has been reached. Bit[0] – Temperature Threshold Alarm Bit[1] – Spare Block Threshold Alarm Bits[15:2] – Reserved
3-2	Temperature Threshold Bits[14:0] – Temperature in 1/16 th Celsius resolution. Bit[15] – Sign bit for temperature (1 = negative, 0 = positive) If the <i>Temperature Threshold Alarm</i> bit is enabled and when the temperature goes above this value, the <i>Temperature Trip</i> bit will be set in the SMART and Health Data structure defined in Table 4-2.
4	Spare Block Threshold: Remaining spare capacity as % of factory configured space. Valid range 0 to 100. If the <i>Spare Block Threshold Alarm</i> bit is enabled and when the space block capacity goes below this threshold, the <i>Spare Blocks Trip</i> bit will be set in the SMART and Health Data structure defined in Table 4-2.
7-5	Reserved



4.3 Get Block NVDIMM Flags (Function Index 3)

This function that is only applicable if block mode is enabled in the NVDIMM (i.e., the Number of Block Control Windows field set is set to a non-zero value in the NVDIMM Control Region Structure).

4.3.1 Input (Arg3)

None

4.3.2 Output

The return value for this function is a buffer formatted as shown in Table 4-5.

Table 4-5 Get Block NVDIMM Flags - Output Format

Field	Byte Length	Byte Offset	Description
Status	4	0	Bytes[1-0] 0 – Success 1 – Not Supported 2 – Non-Existing Memory Device 3 – Reserved 4 – Vendor-specific Error(details in Extended Status Field) 5-FFFFh Reserved Bytes[3-2] Extended Status Field (Vendor-defined)
NVDIMM Flags	4	4	Byte[0] Bit[0] – Block Data Window Invalidation Required – If this bit is set to 1, indicates that the NVDIMM requires the driver to flush previous data from cache lines that will be moved through the Block Data Window, before re-using the Block Data Window for read. If set to '0', flushing of previous data from cachelines that will be moved through the Block Data Window are handled by the platform or VMM. Typical usage of this flag is in a virtualized environment. Bit[1] – Command Register in Block Control Window Latch – If this bit is set to 1,



			<p>indicates that after a write to the Command Register in Block Control Windows, the NVDIMM requires the software to read the same Command Register to ensure that the command is latched before reading contents from Block Data Window.</p> <p>If this bit is set to 0, software is allowed to read the contents of the Block Data Window immediately after writing to the Command Register of Block Control Window.</p> <p>Bits[7:2] – Reserved Note: If this command is not implemented, then the software should assume bit[0] and bit[1] are clear.</p> <p>Bytes[3-1] – Reserved</p>
--	--	--	--

4.4 Get Namespace Label Size (Function Index 4)

The usage of this function is detailed in *NVDIMM Namespace Specification*.

4.4.1 Input (Arg3)

None

4.4.2 Output

The return value for this function is a buffer formatted as shown in Table 4-6.

Table 4-6 Get Namespace Label Size – Output Format

Field	Byte Length	Byte Offset	Description
-------	-------------	-------------	-------------



NVDIMM DSM Interface Example

Status	4	0	Bytes[1-0] 0 – Success 1 – Not Supported 2 – Non-Existing Memory Device 3 - Reserved 4 – Vendor-specific Error(details in Extended Status Field) 5-FFFFh Reserved Bytes[3-2] Extended Status Field (Vendor-defined)
Size of Namespace Label Area	4	4	Size returned in bytes
Max Namespace Label Data Length	4	8	In bytes, Maximum size of the namespace label data length supported by the platform in <i>Get/Set Namespace Label Data</i> functions

4.5 Get Namespace Label Data (Function Index 5)

The usage of this function is detailed in *NVDIMM Namespace Specification*.

4.5.1 Input (Arg3)

The input is a package containing a single buffer, where the buffer is formatted as shown in Table 4-7.

Table 4-7 Get Namespace Label Data – Input Format

Field	Byte Length	Byte Offset	Description
Offset	4	0	In bytes Indicates the offset in the namespace label data area, to which the namespace label data is to be read from the target NVDIMM
Length	4	4	In bytes

4.5.2 Output

The return value for this function is a buffer formatted as shown in Table 4-8.

Table 4-8 Get Namespace Label Data – Output Format

Field	Byte Length	Byte Offset	Description
-------	-------------	-------------	-------------



Status	4	0	Bytes[1-0] 0 – Success 1 – Not Supported 2 – Non-Existing Memory Device 3 – Invalid Input Parameters(Offset + Length is > size of Namespace Label Data Area) 4 – Vendor-specific Error(details in Extended Status Field) 5-FFFFh Reserved Bytes[3-2] Extended Status Field (Vendor-defined)
Namespace Label Data	Varies	4	The size of the output is equal to input's <i>Length</i> if <i>Status</i> is Success; otherwise, the contents of rest of the output buffer are not valid.

4.6 Set Namespace Label Data (Function Index 6)

The usage of this function is detailed in *NVDIMM Namespace Specification*.

4.6.1 Input (Arg3)

Input is a package containing a single buffer, where the buffer is formatted as shown in Table 4-9.

Table 4-9 Set Namespace Label Data – Input Format

Field	Byte Length	Byte Offset	Description
Offset	4	0	In bytes Indicates the offset in the namespace label data area, to which the <i>Namespace Label Data</i> is to be written to the target NVDIMM
Length	4	4	In bytes
Namespace Label Data	Varies	8	Namespace label data. Size of the namespace label data is as indicated by <i>Length</i> field above.



4.6.2 Output

The return value for this function is a buffer formatted as shown in Table 4-10.

Table 4-10 Set Namespace Label Data – Output Format

Field	Byte Length	Byte Offset	Description
Status	4	0	Bytes[1-0] 0 – Success 1 – Not Supported 2 – Non-Existing Memory Device 3 – Invalid Input Parameters(Offset + Length is > size of Namespace Label Data Area) 4 – Vendor-specific Error(details in Extended Status Field) 5-FFFFh Reserved Bytes[3-2] Extended Status Field (Vendor-defined)

4.7 Get Vendor-Specific Command Effect Log Size (Function Index 7)

This function returns the maximum data size of output buffer for retrieving the Vendor-Specific Command Effect Log.

4.7.1 Input (Arg3)

None

4.7.2 Output

The return value for this function is a buffer formatted as shown in Table 4-11.

Table 4-11 Get Vendor Specific Command Effect Log Size – Output Format

Field	Byte Length	Byte Offset	Description
-------	-------------	-------------	-------------



Status	4	0	Bytes[1-0] 0 – Success 1 – Not Supported 2 – Non-Existing Memory Device 3 - Reserved 4 – Vendor-specific Error(details in Extended Status Field) 5-FFFFh Reserved Bytes[3-2] - Extended Status Field (Vendor-defined)
Max Command Effect Log Data Length	4	8	In bytes, Maximum size of the Vendor-specific command effect log data buffer supported by the platform

4.8 Get Vendor-Specific Command Effect Log (Function Index 8)

This function returns the Command Effect Log for all of the Vendor-Specific Commands. If the OpCode is not in the Command Effect Log, OSPM may block the Vendor-Specific call for that OpCode.

4.8.1 Input (Arg3)

None

4.8.2 Output

The return value for this function is a buffer formatted as shown in Table 4-12.

Table 4-12 Get Vendor Specific Command Effect Log Size – Output Format

Field	Byte Length	Byte Offset	Description
-------	-------------	-------------	-------------



NVDIMM DSM Interface Example

Status	4	0	<p>Bytes[1-0]</p> <p>0 – Success 1 – Not Supported 2 – Non-Existing Memory Device 3 - Reserved 4 – Vendor-specific Error(details in Extended Status Field) 5-FFFFh Reserved</p> <p>Bytes[3-2] Extended Status Field (Vendor-defined)</p>
OpCode Count	2	4	Number of OpCode command effect logs returned
Reserved	2	6	
Command Effect Data	Varies	8	<p>The command effect data for each OpCode.</p> <p>The Fields in Table 4-13 are repeated <i>OpCode Count</i> times.</p>

Table 4-13 Command Effect Data - Format

Field	Byte Length	Byte Offset	Description
OpCode	4	0	OpCode representing a Vendor-specific command
OpCode Command Effect	4	4	<p>Bit[0] – No Effects (NE) If set to 1, execution of this OpCode does not change DIMM state. If this bit is set, all the following bits should be clear.</p> <p>Bit[1] – Security State Change (SSC) If set to 1, execution of this Opcode results in immediate security state change of the NVDIMM.</p> <p>Bit[2] – DIMM Configuration Change after Reboot (DCC) If set to 1, execution of this Opcode results in change to the configuration of the NVDIMM or data contained within persistent memory regions of the NVDIMM. The change does not take effect until the system reboots.</p>



			<p>Bit[3] – Immediate DIMM Configuration Change (IDCC) If set to 1, execution of this Opcode results in immediate change to the configuration of the NVDIMM or data contained within persistent memory regions of the NVDIMM.</p> <p>Bit[4] – Quiesce All IO (QIO) If set to 1, execution of this Opcode may disrupt on-going operations of the memory region covered by this NVDIMM. The outstanding IO operations corresponding to this NVDIMM must be quiesced before executing this command; otherwise, undefined system behavior will result.</p> <p>Bit[5] - Immediate DIMM Data Change (IDDC) If set to 1, execution of this Opcode results in immediate change to the data written to the NVDIMM.</p> <p>Bit[6] – Test Mode (TM) If set to 1, execution of this Opcode activates a test feature that may disrupt on-going operations. This may result in errors or error recovery operations.</p> <p>Bit[7] – Debug Mode (DM) If set to 1, execution of this Opcode activates a debug feature that is non-disruptive, but may alter performance characteristics of the NVDIMM.</p> <p>Bits[31:8] – Reserved</p>
--	--	--	---



4.9 Vendor-Specific Command (Function Index 9)

This function provides access to the vendor specific commands. Refer to the vendor specific document for the format of the input and output data buffers.

4.9.1 Input (Arg3)

Input is a package containing a single buffer, where the buffer is formatted as shown in Table 4-14.

Table 4-14 Vendor Specific Command – Input Format

Field	Byte Length	Byte Offset	Description
OpCode	4	0	Vendor-specific command OpCode
OpCode Parameters Data Length	4	4	In bytes Length of OpCode parameters data
OpCode Parameters Data	Varies	8	Vendor-specific command input data

4.9.2 Output

The return value for this function is a buffer formatted as shown in Table 4-15.

Table 4-15 Vendor Specific Command – Output Format

Field	Byte Length	Byte Offset	Description
Status	4	0	Bytes[1-0] 0 – Success 1 – Not Supported 2 – Non-Existing Memory Device 3 - Reserved 4 – Vendor-specific Error(details in Extended Status Field) 5-FFFFh Reserved Bytes[3-2] Extended Status Field (Vendor-defined)
Output Data Length	4	4	In bytes. If <i>Status</i> is not <i>Success</i> , output data length returned is 0.
Output Data	Varies	8	The <i>Output Data</i> is valid only when the <i>Output Data Length</i> is non-zero.



NVDIMM DSM Interface Example