LEGAL NOTICE:

This technical proposal is proprietary to the NVM Express, Inc. (also referred to as "Company") and/or its successors and assigns.

NOTICE TO USERS WHO ARE NVM EXPRESS, INC. MEMBERS: Members of NVM Express, Inc. have the right to use and implement this technical proposal subject, however, to the Member's continued compliance with the Company's Intellectual Property Policy and Bylaws and the Member's Participation Agreement.

NOTICE TO NON-MEMBERS OF NVM EXPRESS, INC.: If you are not a Member of NVM Express, Inc. and you have obtained a copy of this document, you only have a right to review this document or make reference to or cite this document. Any such references or citations to this document must acknowledge NVM Express, Inc. copyright ownership of this document. The proper copyright citation or reference is as follows: **"© 2008 to 2024 NVM Express, Inc. ALL RIGHTS RESERVED."** When making any such citations or references to this document you are not permitted to revise, alter, modify, make any derivatives of, or otherwise amend the referenced portion of this document in any way without the prior express written permission of NVM Express, Inc. Nothing contained in this document shall be deemed as granting you any kind of license to implement or use this document or the specification described therein, or any of its contents, either expressly or impliedly, or to any intellectual property owned or controlled by NVM Express, Inc., including, without limitation, any trademarks of NVM Express, Inc.

LEGAL DISCLAIMER:

THIS DOCUMENT AND THE INFORMATION CONTAINED HEREIN IS PROVIDED ON AN "**AS IS**" BASIS. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, NVM EXPRESS, INC. (ALONG WITH THE CONTRIBUTORS TO THIS DOCUMENT) HEREBY DISCLAIM ALL REPRESENTATIONS, WARRANTIES AND/OR COVENANTS, EITHER EXPRESS OR IMPLIED, STATUTORY OR AT COMMON LAW, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, VALIDITY, AND/OR NONINFRINGEMENT.

All product names, trademarks, registered trademarks, and/or servicemarks may be claimed as the property of their respective owners.

NVM Express Workgroup
c/o VTM, Inc.
3855 SW 153rd Drive
Beaverton, OR 97003 USA
info@nvmexpress.org

NVM Express® Technical Proposal

Technical Proposal ID	**TP 4159 PCIe Infrastructure for Live Migration**
Revision Date	**2024.07.30**
Builds on Specification(s)	**NVM Express Base Specification 2.0d** **NVM Express NVM Command Set Specification 1.0d** **NVM Express Management Interface Specification 1.2d**
References	**TP4165 Tracking LBA Allocation with Granularity** **TP4074a Defining Scope for Features**

Technical Proposal Author(s)

Name	**Company**
Mike Allison, Judy Brock, Bill Martin, Dan Helmick, Hanjae Lee, Vipin Agrawal, Jiwon Chang, Mark Gaertner, Nate Thornton	Samsung
Lee Prewitt, Jake Oshins, Scott Lee	Microsoft
Amber Huffman, Chris Sabol, David Dunn, Nicolae Mogoreanu, Yuliya Tarnikova	Google
Paul Suhler, Fred Knight	Kioxia
Ben Walker, Max Gurtovoy, Chaitanya Kulkarni, Jason Gunthorpe	NVIDIA
Peter Onufryk, Yadong Li	Intel

Technical Proposal Overview

This proposal adds capabilities for a host to manage the live migration of a controller to either another controller within the same NVM subsystem or to a controller in a different NVM subsystem in a manner that is transparent to host connected to the controller being migrated.

Revision History

Revision Date	Change Description
2023.10.12	Initial draft
2023.10.24	Made the following changes: • Per the Technical WG meeting review, moved the log pages specific to controller state into a Send/Receive command pair. • The Tracking Action bit is moved to the Management Operation Specific field and the bit polarity is inverted as requested by Jiwon Chang. • Editorial corrections from Jiwon Chang. • Removed TBDs from figure titles and section titles. • Replaced the use of LBA for User Data in the NVMe Base Specification. • Using new title: Track Memory Changes. • Added a Start bit, an End bit, offset and size to the For the Migration Send command Set Controller State management operation to allow for the transfer of controller state that exceeds MDTS.
2023.10.31	• Moved the Management Operation Specific field from bits 15:08 to bits 31:16 in the migration Send command • Fixed the definition of the End bit in Figure SCS-FIG2 when the bit is cleared to ‘0’. • Fixed misspellings • Fixed the index offsets to the Controller State UUID Supported List • Fixed the figure numbering and ordering for the Migration Receive command section. • Changed the name of the Delete User Data Migration Queue to Delete Migration Queue.
2023.11.08	• Renamed Supported Controller Migration State log page to Supported Controller State Formats log page. • Corrected spelling, grammar, references to figures, and references to sections. • Updated the NV field and the NUUID field in the Supported Controller State Formats log page to refer to the counts in the appropriate list. • Added references to figures in fields that reference fields in different data structures. • Updated the Migration Attributes field with a global definition of the field being cleared to 0h if the HMLMS bit is cleared to ‘0’. • Corrected the use of identifies/specifies.
2023.11.09	• Addressed editorial items provided by Phil Colline.
2023.11.09	• Separated Tracking and Migrating commands per the request of the Technical WG. Still need to update descriptions.
2023.11.17	• Moved the Supported Controller State Formats data structure from a log page to a CNS value.
2023.12.06	• Editorial edits from Dan Helmick and Jiwon Chang.
2023.12.07	• Made decision that reading controller state requires the controller to be paused. • Editorial updates based on review and removed discussed/resolved comments.
2023.12.14	• Made the decision to allow reading controller state but the data returned is not consistent between fields.
2024.01.04	• Resolved the decision choices in many comments.
2024.01.10	• Changed “Memory Range” to “Tracking Memory Range” to distinguish from other memory ranges defined by NVMe.
2024.01.13	• Aligned to NVM Express Base Specification 2.0d, NVM Express NVM Command Set Specification 1.0d, and NVM Express Management Interface Specification 1.2d.

2024.02.06	• Lots of clean-up and resolved editorial comments. Per Technical WG decision on 2/1/2024, updating TP for the following: • Abandon the user data logging as a result of the LBA Migration Queue becoming full and place an entry in the LBA Migration Queue indicating the condition. • If the LBA Migration Queue becomes full during the processing of a Migration Send command then controller is paused, a specific status code is generated indicating the condition, and the LBA Migration Queue has an entry of the condition. • Update the Track Management command for the VM memory changes to not use the CC.MPS setting of the controller being migrated. • Added an event to notify the host of the LBA Migration Queue becomes full.
2024.02.07	• Addressed comments from Phil Colline
2024.02.08	• Editorial wording changes
2024.02.13	• Removed the attachment of private namespaces • Deleted the requirement that tracking attributes only allowed if migration is supported (i.e., support is independent. • Added a recommendation that the controller needs to arbitrate across different Controller Data Queues when reporting Controller Data Queue Tail Pointer events. • Added a Paused bit to the Tracked Memory Change Data Structure so the host can determine when all data has been reported after a controller is paused. • Updated the Migration Send command with the Pause management option to allow the reporting of information in the LBA Migration Queue and the Tracked Memory changes to be allow to occur after the completion of the command. • Remove the unpausing of a controller if a Controller Level Reset occurs on that controller. • Add the abort of a Controller Data Queue command when creating a queue is the size of the queue is not aligned to the entry size. • Updated the LBA Migration Queue Entry Type 0 to identify when the last entry for a paused controller s posted and if an entry is posted if that controller is resumed from a Paused state. • Moved the Pause Filled User Data Queue status code from the Track Send command completion section to the Migration Send command completion section.
2024.02.14	Integrated items identified by Jiwon Chang: • Corrected references and section numbering • Clarified the command dwords used by the Track Send command • Moved the Pause Filled User Data Queue status code from the Migration Receive command to the Migration Send command. • Clarified that the controller being paused during a Migration Receive command to obtain the controller state. • Fixed the name for Controller Memory Buffer • Moved the behavior text in the Controller State Size field to outside the data structure definition. • Removed figure SCS-FIG2 since figure SCS-FIG3 is the same information. • Added text describing the Pause Notification value in the Pause Type field in section 8.LM. • Removed references to Stop Track Memory Changes bit as that bit was previously removed. • Clarified the word “restarted” in section 5.MQ in the NVM Command Set Specification.

2024.02.14	Integrated items identified by Nate Thornton: • Add the Maximum Controller Data Queue Contiguous Memory Pages (MCDQCMOP) field to the Identify Controller data structure so the controller can indicate the number of non-contiguous memory chucks allowed in creating a Controller Data Queue. • Updated the Set Controller State management operation specific field to allow a single Migration Send command to transfer all of the controller state.
2024.02.15	• Made the Event Specific Parameter 32-bits (i.e., the whole Completion Queue Entry Dword 1) as other TPs need to use it for other events.
2024.02.21	• Fixed the duplicate SCS-FIG2 definitions which caused renumbering of Figure SCS-FIGx. This caused a renaming to the NVMe defined controller state. • The I/O Queue definitions in the NVMe Controller State have been updated to reflect the data is the data passed by the command that created the queue so as to not repeat the information. • Per the request of Solidigm I add the Maximum Controller Data Queue PRP Count field to the Identify Controller data structure. • Incorporated editorial items reported by Judy Brock and Nicole Ross. • Fixed the byte offsets for the Tracked Memory Changed Descriptor x fields in the Tracked Memory Change data structure.
2024.02.22	• Updated the Log Used Data Changes management operation to clear any pending Controller Data Queue Full Error event on a successful command completion. • Added a requirement that the Migration Send command be aborted if I/O queues exist in the controller that are not defined by the controller state data. • Clarified the committed state to include creating I/O queues defined by the controller state. • Added a requirement to clear any pending Controller Data Queue Tail Pointer event on a successful Set Features command specifying the Controller Data Queue feature. • Renamed the Pause Filled User Data Queue status code to User Data Queue Not Logging.
2024.02.23	• Added Phase 3 notes. Removed comments to needed for Phase 3. • Changed the use of “arbitrate” to “balance”
2024.02.27	Feedback from Dan Hubbard: • Clarified the text about the Controller Data Queue Full Error event. • Phase 3 editorial items • Added a requirement to clear the Maximum User Data Migration Queues field to 0h if the tracking user data is not supported. • Changed the name of “Host Managed Live Migration Support” to “Host Managed Live Migration Support” • Changed the Get Controller State management operation to being mandatory. • Added the behavior associated to the User Data Logging Started (UDLS) bit. • Added the ability to report if opcode in the LBA Migration Queue Entry is an I/O command or an Admin command.
2024.02.29	Feedback from Jonathon Hughes: • Add NVM subsystem maximum number of User Data Migration Queues Feedback from Gordon Waidhofer: • Editorial changes • Allowed the CDQ size to be specified in dwords • Increased the sizes of the head and tail pointers for the CDQ feature. • Changed the Number of Dwords field in the Track Receive command to be a 0’s based number. • Removed the Number of Dwords field on the requirement to abort a Migration Send command for controller state data on the offset being past the end of the actual controller state data. • The NVMe Controller State Size field and Vendor Specific Size field in the Controller State data structure define the size of the associated data fields.

2024.02.29	• Moved the new events to a new One-Shot event type.
2024.03.05	• Made the Migration Send command be able to return successful if the controller is paused independent of the LBA Migration Queue becoming Full. This means the User Data Queue Not Logging status code is no longer needed. • Added a statement that the posting of entries into the LBA Migration Queue is not tied to the Track Send command completion. • Added NVM subsystem scope controller support counts for the fields added to the Identify Controller data structure to allow an implementation to pool resources. • Editorial changes.
2024.03.07	• Removed the text associated to the User Data Migration queue not being full from the Migration Send command for the Pause management operation.
2024.03.13	• Added the requirement to abort a CDQ command if a User Data Migration Queue exists for the specified controller. Changes due to issues raised by Randy Jennings: • Added the Controller Data Queue command to the list of commands that uses the Invalid Controller Identifier status code. • Editorial • Added Get Features command since new feature returns data. • Marked that CDQ feature uses memory buffer for attributes. • Identified CDQ Feature command dwords fields not used by Get Features command • Deleted unused field in the Track Received command and added CDW11 for the Tracked Memory Changes management operation • MTR bit is cleared to ‘0’ if NTMCD filed cleared to 0h • Added the requirement to abort a CDQ command if a User Data Migration Queue exists for the specified controller.
2024.03.19	• Number of Versions field made 0-based. • Made the Logging Action bit a 4-bit field for scalability. • Clarified that the controller state data is not consistent if the migrating controller is reset while processing the command to get the controller state. • Removed the opcode and command type from the LBA Migration Queue entry and added a bit to define if the LBAs being reported are deallocated.
2024.03.19	• Forbid a Sanitize command if any controller in the NVM subsystem is paused.
2024.03.28	• Corrected “about” being “abort” • Editorial fixes
2024.04.02	• Per Technical WG on 3/28/2024 made the following technical changes: ○ Any reset of a migratable controller cause that controller to be reset and the Pause state is removed. ○ A reset of a controller managing migrations does not affect the state of a migratable controller. ○ Added requirements around the sequencing of Migration Send commands for setting the controller state. ○ Added the ability for a controller to limit the size of the memory ranges allowed to be tracked. ○ Added that controller state can be set if the controller is Paused, disabled (CC.EN is cleared to ‘0’), or a secondary controller that is offline. ○ Clarified that the Address field for a Track Memory Changes management operation for the Tracking Send command is for the controller associated by the specified controller. • Minor Phase 3 clean up. • As reported by Hanjae Lee, If a Track Send command is sent to a controller where the LBA Migration Queue is already full, then the controller may need to abort the command. • Updated the LBA Migration Queue Entry to be able to indicate that all LBAs in the namespace have changed.

2024.04.03	• As reported by Hanjae Lee, If a Track Send command is sent to a controller where the LBA Migration Queue is already full, then the controller may need to abort the command. • Updated the LBA Migration Queue Entry to be able to indicate that all LBAs in the namespace have changed.
2024.04.04	• Put a note that the starting address for the VM memory tracking is the PCIe Address.
2024.05.01	Incorporating Phase 3 feedback • From Kioxia posted Phase 2 response. Incorporating the following functional changes: • The Track Send command and Track Receive command for the VM Memory tracking are only allow host memory (i.e., no CMB and PRM support) • HMB is not allowed to be supported on a migratable controller.
2024.05.09	Incorporating more Phase 3 feedback • From Kioxia posted Phase 2 response.
2024.05.16	• Incorporating more Phase 3 feedback from Kioxia posted Phase 2 response. • Updated the LBA Migration Queue entry Phase Tag bit to indicate that bit has to be last update to the entry. • Reordered bytes and sections to align to specifications • Corrected the NVM Command Set spec to update the specific identify controller data structure and opposed to the namespace data structure.
2024.05.21	• Aligned to ECN116 • Defined the Controller Data Queue Identifier.
2024.05.23	• Updated section 8.LM to use named NVM subsystems, host, and controllers. • Editorial changes • Feedback from Kioxia addressed.
2024.05.28	• Changed Pause to Suspend • Added text to allow controller to ignore tracking addresses of MSI and MSI-X registers. • Renamed hosts and controllers in section 8.LM • Added a requirement to the CDQ Feature to abort the command if a head pointer slot passes the tail pointer (i.e., a valid slot, but an invalid value). • Changed “Invalid Field In Command” to “Invalid Field in Command” • Changed section 8.CDQ to match the NEXT file Completion Queue description as previous section was based on NVMe Base specification 2.0. • Simplified the text in section 4.1.NEW.
2024.06.x07	• Corrected grammar in section 5.MSC.1.1.
2024.07.10	• 30-day member review modifications: many editorial cleanup • Added the Track Receive, Migration Send, and Migration Receive commands as being affected by the Invalid Controller Identifier status code • Deleted repeated requirements related to the MCUDMQ and MNSUDMQ fields. • Updated the MCNMR and NMCNMR fields to make clear that the controller may require a single memory range. • Updated commands use of the Data Pointer field • Made the NUMDL field of the Migration Receive command a 0’s based number. • Fields in the Identify Controller were changed form a 2-bute field to a 1-byte field • Clarified that the User Data Migration Queue is deleted if the DUDMQ bit is set to ‘1’ and the command completes successful. • The Controller Data Queue Identifier by the Controller Data Queue command is only for the create operation • Made the NMCNMR field be greater than or equal to the MCNMR field • Bits 13:12 were a gap in the I/O Completion Queue State Data Structure so reformatted the data to fix. • Allowed the Tail Pointer Trigger to be set if the queue is empty. • Clarified the address relationship in the Memory Range Tracking Descriptor • Updated section 8.LM to uniquely name the MMHs between the source and destination •

2024.07.11	• Used consistent text on status code requirements when aborting commands. • Modified the Get Controller State management operation of the Migration Receive command to move the Controller Identifier field into Command Dword 11 so the Controller Identifier field is consistently in the same place for the Migration Receive command, Migration Send command, Track Receive command, and the Track Send command. • Corrected step 7 in the 8.LM.1 Process for Migrating a Controller section to state the actual field with a reference.
2024.07.11	• Replaced several wrong references in Tracking Attributes (TRATTR) field – saved as "…2024.07.11b …JB" version
2024.07.12	• Change the abbreviations MCNMR and NMCNMR to MCMR and NMCMR, respectively, because the word associated with the "N" was removed from the long names. Also removed the word "noncontiguous" from uses of the field name in running text because it was removed from the name of the field and is now incorrect. • Change "a MMC" to "an MMC"; and "a MMH" to "an MMH" • Change MMCs to singular (eliminate confusion with "source") • Change MMHs to singular (eliminate confusion with MMHS – meaning "source") • Add definitions for MMH, MMHS and MMHD.
2024.07.13	• Added 07.12.2024 changes above (in that version, they were comments but no actual changes) • Minor editorial fixes
2024.07.15	• Additional editorial changes in section 8.LM
2024.07.16	• Added definition for MMC
2024.07.27	• Editorial updates
2024.07.28	• Fixed the naming of the NVMe Controller State Version list in 2 places. • Changed inserted text from red to blue • Changes paragraph text from size 9 to size 10.
2024.07.30	• Clarified the race condition that occurs if the host issues a Set Features command to cause a Tail Pointer Trigger if a Tail Pointer Trigger is already enabled.

Description for Changes Document for NVM Express Base Specification 2.0d

New Features/Feature Enhancements/Required Changes:

- Added the ability for a host to manage the live migration of a controller from one NVM subsystem to another NVM subsystem.
    - Description of change.
        - Defined a new optional Track Receive command and Track Send command to track changes to:
            - user data; and
            - host memory due to the process of commands.
        - Defined a new optional Migration Receive command and Migration Send command to manage the migration of a controller by allowing the suspending of a controller, resuming of a controller, and transfer of controller state.
        - Defined a new optional Controller Data Queue command to allow a host to provide host memory for queuing information from the controller to the host. An LBA Migration Queue can be created to allow the controller to post logical block changes from a different controller once user data tracking has started as a result of a Track Send command.
        - Defined a new optional Controller Data Queue feature to allow a host to specify the head pointer value and to identify a slot that when posted with an entry by the controller causes the controller to issue the new Controller Data Queue Tail Pointer event.
    - References
        - Technical Proposal TP4159

Description for Changes Document for NVM Express NVM Command Set Specification 1.0d

New Features/Feature Enhancements/Required Changes:

- Defined an LBA Migration Queue that is associated with a User Data Migration Queue.
    - Description of change.
        - Defined the requirements and entries for an LBA Migration Queue (i.e., a type of Controller Data Queue) that is used to track the changes to logical blocks in namespaces attached to a controller. The queue is created and deleted using the

Controller Data Queue command. The posting of entries is enabled and disabled using the Migration Send command.

- References
    - Technical Proposal TP4159

Description for Changes Document for NVM Express Management Interface Specification 1.2d

New Features/Feature Enhancements/Required Changes:

- Defined the support requirements for the new commands and features associated with the support for controller migration.
    - Description of change.
        - Defined the support requirements for the following Admin commands:
            - Track Receive command
            - Track Send command
            - Migration Receive command
            - Migration Send command
        - Defined the support requirements for the following features:
            - Controller Data Queue
    - References
        - Technical Proposal TP4159

Markup Conventions:

Black:	Unchanged (however, hot links are removed)
~~Red Strikethrough~~:	Deleted
Blue:	New
Blue Highlighted:	TBD values, anchors, and links to be inserted in new text.
~~Purple Strikethrough~~:	Existing text moved to a different location.
Purple:	Inserted existing text moved from a different location.
Orange:	From another Technical Proposal
<Green Bracketed>:	Notes to editor

Modify a portion of section 1.5 as shown below:

1.5 Definitions

…

1.5.TBD7 MMC

A Migration Management Controller. Refer to section 8.LM.

1.5.TBD8 MMH

A Migration Management Host. Refer to section 8.LM.

1.5.TBD9 MMHD

A Migration Management Host associated with a Migration Management Controller in a Destination NVM Subsystem. Refer to section 8.LM.

1.5.TBD10 MMHS

A Migration Management Host associated with a Migration Management Controller in a Source NVM Subsystem. Refer to section 8.LM.

Modify a portion of section 3 as shown below:

…

3 NVM Express Architecture

…

3.1 NVM Controller Architecture

…

3.1.2 Controller Types

…

3.1.2.1 I/O Controller

…

3.1.2.1.1 Command Support

…

Figure 22: I/O Controller – Admin Command Support

Command	Command Support Requirements [1]	Reference
…		
Sanitize	O	5.24
Track Receive	O	5.TRC
Track Send	O	5.TSC
Migration Receive	O	5.MRC
Migration Send	O	5.MSC
Controller Data Queue	O	5.CDQC
Property Set	M[3]	6.6
…		

…

3.1.2.1.3 Features Support

…

Figure 25: I/O Controller – Feature Support

Feature Name	Feature Support Requirements [1]	Logged in Persistent Event Log [1]
…		
Endurance Group Event Configuration	O	O
Controller Data Queue	O	O
I/O Command Set Profile	O	O
…		

…

3.1.2.2 Administrative Controller

…

3.1.2.2.1 Command Support

…

Figure 28: Administrative Controller – Admin Command Support

Command	Command Support Requirements [1]	Reference
…		
Sanitize	O	5.24
Track Receive	O	5.TRC
Track Send	O	5.TSC
Migration Receive	O	5.MRC
Migration Send	O	5.MSC
Controller Data Queue	O	5.CDQC
Property Set	M[3]	6.6
…		

…

3.1.2.2.3 Features Support

…

Figure 30: Administrative Controller – Feature Support

Feature Name	Feature Support Requirements [1]	Logged in Persistent Event Log [1]
…		
Endurance Group Event Configuration	O	O
Controller Data Queue	O	O
I/O Command Set Profile	P	P
…		

…

3.1.2.3 Discovery Controller

…

3.1.2.3.2 Command Support

…

Figure 32: Discovery Controller – Admin Command Support

Command	Command Support Requirements [1]	Reference
…		
Sanitize	O	5.24
Track Receive	P	5.TRC
Track Send	P	5.TSC
Migration Receive	P	5.MRC
Migration Send	P	5.MSC
Controller Data Queue	P	5.CDQC
Property Set	M[3]	6.6
…		

…

3.1.2.3.4 Features Support

…

Figure 34: Discovery Controller – Feature Support

Feature Name	Feature Support Requirements [1]	Logged in Persistent Event Log [1]
…		
Endurance Group Event Configuration	P	P
Controller Data Queue	P	P
Vendor Specific	O	O
…		

…

3.3 NVM Queue Models

…

3.3.3 Queueing Data Structures

…

3.3.3.2 Common Completion Queue Entry

…

3.3.3.2.1 Status Field Definition

…

3.3.3.2.1.2 Command Specific Status Definition

…

Figure 96: Status Code – Command Specific Status Values

Value	Description	Commands Affected
…		
IFh	Invalid Controller Identifier	Virtualization Management, Track Send, Track Receive, Migration Send, Migration Receive, Controller Data Queue
…		
2Dh	Identifier Unavailable	Capacity Management
37h	Invalid Controller Data Queue	Set Features, Get Features, Track Send, Controller Data Queue
38h	Not Enough Resources	Track Send, Controller Data Queue
39h	Controller Suspended	Track Send, Sanitize
3Ah	Controller Not Suspended	Track Send
3Bh	Controller Data Queue Full	Track Send
3Ch~~2Eh~~ to 6Fh	Reserved	
70h to 7Fh	Directive Specific	NOTE 1
…		

…

3.10 Privileged Actions

Privileged actions are actions (e.g., command, property write) that affect or have the potential to affect the state beyond the controller and attached namespaces.

Examples of privileged actions are:

- Admin commands including Namespace Management, Namespace Attachment, Virtualization Management, Format NVM, Set Features with Feature Identifier 17h (i.e., Sanitize Config, refer to section 5.27.1.19), Sanitize, ~~and~~ Capacity Management, Controller Data Queue, Migration Receive, Migration Send, Track Send, and Track Receive;
- Property Writes including NVM Subsystem Reset; and
- Some Vendor specific commands and properties.

Modify a portion of section 5 as shown below:

…

5 Admin Command Set

…

Figure 140: Opcodes for Admin Commands

Opcode by Field			Combined Opcode [1]	Namespace Identifier Used [2]	Command	Command Set Specific [8]
(07)	(06:02)	(01:00)				
Generic Command	Function	Data Transfer[3]				
…						
0b	010 00b	00b	20h	No	Capacity Management	No
0b	010 01b	00b	24h	No	Lockdown	No
0b	011 11b	01b	3Dh	No	Track Send	No
0b	011 11b	10b	3Eh	No	Track Receive	No
0b	100 00b	01b	41h	No	Migration Send	No
0b	100 00b	10b	42h	No	Migration Receive	No
0b	100 01b	01b	45h	No	Controller Data Queue	No
0b	111 11b	00b	7Ch	No	Doorbell Buffer Config	No
…						

…

5.2 Asynchronous Event Request command

Asynchronous events are used to notify a host ~~software~~ of status, error, and health information as these events occur. To enable asynchronous events to be reported by the controller, a host ~~software~~ needs to submit one or more Asynchronous Event Request commands to the controller. The controller specifies an event to the host by completing an Asynchronous Event Request command. A host ~~Host software~~ should expect that the controller may not execute the command immediately; the command should be completed when there is an event to be reported.

The Asynchronous Event Request command is submitted by a host ~~software~~ to enable the reporting of asynchronous events from the controller. This command has no timeout. The controller posts a completion queue entry for this command when there is an asynchronous event to report to the host. If Asynchronous Event Request commands are outstanding when the controller is reset, then each of those commands is aborted and should not return a CQE.

All command specific fields are reserved.

A host ~~Host software~~ may submit multiple Asynchronous Event Request commands to reduce event reporting latency. The total number of simultaneously outstanding Asynchronous Event Request commands is limited by the value indicated in the Asynchronous Event Request Limit field in the Identify Controller data structure in Figure 276.

Asynchronous events are grouped into event types. The event type is indicated in the Asynchronous Event Type field in Dword 0 of the completion queue entry for the Asynchronous Event Request command. When the controller posts a completion queue entry for an outstanding Asynchronous Event Request command and thus reports an asynchronous event, subsequent events of that event type are automatically masked by the controller until the host clears that event. Unless otherwise stated, an event is cleared by reading the log page associated with that event (refer to section 5.16). If that log page is not accessible because media is not ready (i.e., the controller aborts the Get Log Page command with a status code of Admin Command Media Not Ready), then the controller shall not post a completion queue entry for that asynchronous event until the controller is able to successfully return the log page that is required to be read to clear the asynchronous event.

The following event types are defined:

a) **Error event:** Indicates a general error that is not associated with a specific command (refer to Figure 146). The controller shall set the Log Page Identifier field to the identifier of the Error Information log page (i.e., 01h). To clear this event, a host ~~software~~ reads that log page (i.e., the

Error Information log page (refer to section 5.16.1.2)) using the Get Log Page command with the Retain Asynchronous Event bit cleared to ‘0’;

b) **SMART / Health Status event:** Indicates a SMART or health status event (refer to Figure 147). The controller shall set the Log Page Identifier field to the identifier of the SMART/Health Information log page (i.e., 02h). To clear this event, a host ~~software~~ reads that log page (i.e., the SMART / Health Information log (refer to section 5.16.1.3)) using the Get Log Page command with the Retain Asynchronous Event bit cleared to ‘0’. The SMART / Health conditions that trigger asynchronous events may be configured in the Asynchronous Event Configuration feature using the Set Features command (refer to section 5.27.1.8);

c) **Notice event:** Indicates a general event (refer to Figure 148). The controller shall set the Log Page Identifier field to the log page identifier of the appropriate log page as described in Figure 148. To clear this event, a host ~~software~~ reads that log page (i.e., the appropriate log page as described in Figure 148). The conditions that trigger asynchronous events may be configured in the Asynchronous Event Configuration feature using the Set Features command (refer to section 5.27.1.8);

d) **I/O Command Specific Status events:** Events that are specific to an I/O command (refer to Figure 149);

e) **Immediate events:** Events that are only reported when an outstanding Asynchronous Event Request command exists at the time the event occurs. If the event occurs and there is no outstanding Asynchronous Event Request command, then the event shall not be reported. No log page is associated with these events. These events include:

    A. Normal NVM Subsystem Shutdown event;

f) **One-Shot events:** Events that are only reported once and the event is cleared by the controller when that event is reported in the completion of an Asynchronous Event Request command. Refer to the description of each event to determine how to clear any pending events (e.g., refer to the Controller Data Queue feature in section 5.27.1.CDQF); and

g) **Vendor Specific event:** Indicates a vendor specific event. To clear this event, a host ~~software~~ reads the indicated vendor specific log page using the Get Log Page command with the Retain Asynchronous Event bit cleared to ‘0’.

The Sanitize Operation Completed With Unexpected Deallocation asynchronous event shall be supported if the controller supports the Sanitize Config feature (refer to section 5.27.1.19).

Asynchronous events are reported due to a new entry being added to a log page (e.g., Error Information log page) or a status update (e.g., status in the SMART / Health log page). A status change may be permanent (e.g., the media has become read only) or transient (e.g., the temperature reached or exceeded a threshold for a period of time). A host ~~Host software~~ should modify the event threshold or mask the event for transient and permanent status changes before issuing another Asynchronous Event Request command to avoid repeated reporting of asynchronous events.

If an event occurs for which reporting is enabled and there are no Asynchronous Event Request commands outstanding, the controller should retain the event information for that Asynchronous Event Type (i.e., a pending event) and use that information as a response to the next Asynchronous Event Request command that is received. If a Get Log Page command clears the event prior to receiving the Asynchronous Event Request command or if a Controller Level Reset occurs, then a notification is not sent. If multiple events of the same type occur that have identical responses to the Asynchronous Event Request command, then those events may be reported as a single response to an Asynchronous Event Request command. If multiple events occur that are of different types or have different responses to the Asynchronous Event Request command, then the controller should retain a queue of those events for reporting in responses to subsequent Asynchronous Event Request commands.

…

5.2.1 Command Completion

…

Dword 0 and Dword 1 of the completion queue entry contains information about the asynchronous event. The definition of Dword 0 of the completion queue entry is in Figure 145. The definition of Dword 1 of the completion queue entry is in Figure AER_FIG.

Figure 145: Asynchronous Event Request – Completion Queue Entry Dword 0

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:24</td><td>Reserved</td></tr>
<tr><td>23:16</td><td><b>Log Page Identifier:</b> Indicates the log page associated with the asynchronous event. This log page needs to be read by the host to clear the event.
For asynchronous events not associated with a log page (refer to section 5.2), this field should be ignored by the host.</td></tr>
<tr><td>15:08</td><td><b>Asynchronous Event Information:</b> Refer to Figure 146, Figure 147, Figure 148, ~~and ~~Figure 149, Figure 150, and Figure AER-OS for detailed information regarding the asynchronous event.</td></tr>
<tr><td>07:03</td><td>Reserved</td></tr>
<tr><td>02:00</td><td><b>Asynchronous Event Type:</b> Indicates the event type of the asynchronous event. More specific information on the event is provided in the Asynchronous Event Information field.
<table>
<tr><th>Value</th><th>Definition</th><th>Reference</th></tr>
<tr><td>000b</td><td>Error status</td><td>Figure 146</td></tr>
<tr><td>001b</td><td>SMART / Health status</td><td>Figure 147</td></tr>
<tr><td>010b</td><td>Notice</td><td>Figure 148</td></tr>
<tr><td>011b</td><td>Immediate</td><td>Figure 150</td></tr>
<tr><td>100b</td><td>One-Shot</td><td>Figure AER-OS</td></tr>
<tr><td>101b</td><td>Reserved</td><td>~~-~~</td></tr>
<tr><td>110b</td><td>I/O Command specific status</td><td>Figure 149</td></tr>
<tr><td>111b</td><td>Vendor specific</td><td>~~-~~</td></tr>
</table>
</td></tr>
</table>

Figure AER-FIG: Asynchronous Event Request – Completion Queue Entry Dword 1

Bits	Description
31:00	**Event Specific Parameter (EVNTSP):** This field specifies information for a specific event. If this field is not defined by a specific event, then this field is reserved for that specific event.

…

~~The information in either Figure 146, Figure 147, Figure 148, or Figure 149 is returned in the Asynchronous Event Information field, depending on the Asynchronous Event Type.~~

…

Figure 150: Asynchronous Event Information – Immediate

Value	Description
00h	**NVM Subsystem Normal Shutdown:** This controller has started performing a normal NVM Subsystem Shutdown that is due to: • the value 4E726D6Ch ("Nrml") has been written to the NSSD.NSSC field within the NVM subsystem or Domain; or • an NVMe-MI Shutdown command (refer to the NVM Express Management Interface Specification) being processed. Refer to section 3.6.3.
01h to FFh	Reserved

Figure AER-OS: Asynchronous Event Information – One Shot

Value	Definition
00h	**Controller Data Queue Tail Pointer:** Indicates that an entry in a Controller Data Queue was posted at the tail pointer location specified by the Controller Data Queue feature (refer to section 5.27.1.CDQF). Figure AER-CDQID defines the Event Specific Parameter field in Completion Queue Entry Dword 1 field for this event.
01h	**Controller Data Queue Full Error:** The controller detected that a Controller Data Queue became full. Figure AER-CDQID defines the Event Specific Parameter field in Completion Queue Entry Dword 1 for this event.
02h to FFh	Reserved

Figure AER-CDQID: Controller Data Queue Tail Pointer – Event Specific Parameter

Bits	Description
31:16	Reserved
15:00	**Controller Data Queue Identifier (CDQID):** This field indicates the identifier for the Controller Data Queue associated with this event (refer to Figure CDQC-FIG9).

…

5.CDQC Controller Data Queue command

The Controller Data Queue command is used to manage queues in host memory that a controller posts a specific type of data (refer to section 8.CDQ).

The Controller Data Queue command uses the Command Dword 10 field. The use of the PRP1 field, Command Dword 11 field, and Command Dword 12 field is specific to the management operation specified by the Select field. All other command specific fields are reserved.

The Select field defined in Figure CDQC-FIG1 determines which management operation is to be performed by this command and refer to section 5.CDQC.1 for a description of each management operation.

Figure CDQC-FIG1: Controller Data Queue – Command Dword 10

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:16</td><td><b>Management Operation Specific (MOS):</b> The definition of this field is specific to a management operation (refer to the Select field). If a management operation does not define the use of this field, then this field is reserved.</td></tr>
<tr><td>15:08</td><td>Reserved</td></tr>
<tr><td>07:00</td><td><b>Select (SEL):</b> This field specifies the type of management operation to perform.
<table>
<tr><th>Value</th><th>Management Operation</th><th>Reference Section</th></tr>
<tr><td>0h</td><td>Create Controller Data Queue</td><td>5.CDQC.1.1</td></tr>
<tr><td>1h</td><td>Delete Controller Data Queue</td><td>5.CDQC.1.2</td></tr>
<tr><td>2h to BFh</td><td colspan="2">Reserved</td></tr>
<tr><td>C0h to FFh</td><td colspan="2">Vendor Specific</td></tr>
</table></td></tr>
</table>

5.CDQC.1 Controller Data Queue Management Operations

5.CDQC.1.1 Create Controller Data Queue (Management Operation 0h)

The Create Controller Data Queue management operation of the Controller Data Queue command is used to create a queue in host memory to which a controller posts the information specified by the Queue Type field (refer to Figure CDQC-FIG5).

The Create Controller Data Queue management operation uses the PRP Entry 1 field, Command Dword 11 field, and Command Dword 12 field.

If a PRP List is provided to describe the Controller Data Queue, then the PRP List shall be maintained by the host at the same location in host physical memory and the values in the PRP List shall not be modified until the Controller Data Queue is deleted (refer to section 5.CDQC.1.2) or the controller is reset. If the PRP List values are modified, then the behavior is undefined.

If the number of memory ranges specified by the PRP Entry 1 field is greater than the value defined by the Maximum CDQ Memory Ranges (MCMR) field in the Identify Controller data structure (refer to Figure 276), then the controller shall abort the command with a status code of Invalid Field in Command.

If the number of memory ranges specified by the PRP Entry 1 field plus the number of memory ranges that exists for all of the Controller Data Queues in the NVM subsystem is greater than the value defined by the NVM Subsystem Maximum Controller CDQ Memory Ranges (NMCMR) field in the Identify Controller data structure (refer to Figure 276), then the controller shall abort the command with a status code of Invalid Field in Command.

If the current number of User Data Migration Queues that exist in the controller is equal to the value in the Maximum Controller User Data Migration Queues (MCUDMQ) field in the Identify Controller data structure, then the controller shall abort the command with a status code of Invalid Field in Command.

If the current number of User Data Migration Queues that exist in the NVM subsystem is equal to the value in the Maximum NVM Subsystem User Data Migration Queues (MNSUDMQ) field in the Identify Controller data structure, then the controller shall abort the command with a status code of Invalid Field in Command.

Figure CDQC-FIG2: Create Controller Data Queue – PRP Entry 1

Bits	Description
63:00	**PRP Entry 1 (PRP1):** If the PC bit is set to '1', then this field specifies a 64-bit base host memory (refer to section 1.5.28) address pointer of the Controller Data Queue that is physically contiguous. The address pointer is memory page aligned (based on the value in CC.MPS field) unless otherwise specified. If the PC bit (refer to Figure CDQC-FIG3) is cleared to '0', then this field specifies a PRP List pointer that describes the list of pages that constitute the Controller Data Queue. The list of pages is host memory (refer to section 1.5.28) that is page aligned (based on the value in CC.MPS field) unless otherwise specified. In both cases the PRP Entry shall have an offset of 0h. In a non-contiguous Controller Data Queue, each PRP Entry in the PRP List shall have an offset of 0h. If there is a PRP Entry with a non-zero offset, then the controller shall return an error of PRP Offset Invalid. If the memory specified by this field is not host memory (e.g., CMB or PMR), then the controller shall abort the command with a status code of Invalid Field in Command.

Figure CDQC-FIG3: Create Controller Data Queue – Command Dword 11

Bits	Description
31:16	**Create Queue Specific (CQS):** The definition of this field is specific to the type of queue being created (refer to Queue Type (QT) field in Figure CDQC-FIG5). If the type of queue being created does not define the use of this field, then this field is reserved.
15:01	Reserved
00	**Physically Contiguous (PC):** If set to '1', then the Controller Data Queue is physically contiguous and PRP Entry 1 (PRP1) is the address of a contiguous physical buffer. If cleared to '0', then the Controller Data Queue is not physically contiguous and PRP Entry 1 (PRP1) is a PRP List pointer.

Figure CDQC-FIG4: Create Queue – Command Dword 12

Bits	Description
31:00	**Controller Data Queue Size (CDQSIZE):** This field specifies the size of the queue to be created in dwords.

If the Controller Data Queue Size field value is not a multiple of the size of the entries for the type of Controller Data Queue as specified by the Queue Type field (refer to Figure CDQC-FIG5), then the controller shall abort the command with a status code of Invalid Field in Command.

The Management Operation Specific field (refer to Figure CDQC-FIG1) for the Create Controller Data Queue management operation is specified in Figure CDQC-FIG5.

Figure CDQC-FIG5: Create Controller Data Queue – Management Operation Specific

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>15:08</td><td>Reserved</td></tr>
<tr><td>07:00</td><td><b>Queue Type (QT):</b> This field specifies the type of queue to be created which defines the information that the controller posts into the entries of the queue.
<table>
<tr><th>Value</th><th>Definition</th><th>Reference Section</th></tr>
<tr><td>0h</td><td>User Data Migration Queue</td><td>5.CDQC.1.1.1</td></tr>
<tr><td>1h to BFh</td><td colspan="2">Reserved</td></tr>
<tr><td>C0h to FFh</td><td colspan="2">Vendor Specific</td></tr>
</table></td></tr>
</table>

If the PRP Entry 1 field has a non-zero PRP offset, then the controller shall abort the command with a status code of PRP Offset Invalid.

5.CDQC.1.1.1 User Data Migration Queue (Queue Type 0h)

The User Data Migration Queue of the Controller Data Queue command is used to create a queue for logging of user data modified during the migration of the controller specified by the Controller Identifier field as defined by Figure CDQC-FIG6.

If a User Data Migration Queue is associated with the same controller specified by the Controller Identifier field (refer to Figure CDQC-FIG6), then the controller shall abort the command with a status code of Invalid Field in Command.

If the number of User Data Migration Queues that exist in the controller is equal to the value defined in the Maximum Controller User Data Migration Queues (MCUDMQ) field in the Identify Controller data structure (refer to Figure 276), then the controller shall abort this command with a status code of Not Enough Resources.

If the number of User Data Migration Queues that exist in the NVM subsystem is equal to the value defined in the Maximum NVM Subsystem User Data Migration Queues (MNSUDMQ) field in the Identify Controller data structure (refer to Figure 276), then the controller shall abort this command with a status code of Not Enough Resources.

Figure CDQC-FIG6: User Data Migration Queue – Create Queue Specific

Bits	Description
15:00	**Controller Identifier (CNTLID):** This field specifies the controller associated with the information contained in the User Data Migration Queue.

Refer to the applicable I/O Command Set specification for the requirements and formats of the entries for the User Data Migration Queue.

5.CDQC.1.2 Delete Controller Data Queue (Management Operation 1h)

The Delete Controller Data Queue management operation of the Controller Data Queue command is used to delete a CDQ.

The Delete Controller Data Queue management operation uses the Command Dword 11 field.

Figure CDQC-FIG8: Delete Controller Data Queue – Command Dword 11

Bits	Description
31:16	Reserved
15:00	**Controller Data Queue Identifier (CDQID):** This field specifies the identifier (refer to Figure CDQC-FIG9) of the existing User Data Migration Queue to delete.

If the value in the Controller Data Queue Identifier field specifies a CDQ that does not exist in the controller processing the command, then the controller shall abort the command with a status code of Invalid Controller Data Queue.

5.CDQC.2 Command Completion

Upon completion of the Controller Data Queue command, the controller posts a completion queue entry to the Admin Completion Queue indicating the status for the command. Section 5.CDQC.1 describes completion details for each management operation.

If a status code of Successful Completion is returned for the Create Controller Data Queue management operation, then Dword 0 of the completion queue entry contains the identifier of the Controller Data Queue created as defined in Figure CDQC-FIG9. The indicated identifier is unique to the controller that processed the command.

Figure CDQC-FIG9: Controller Data Queue – Completion Queue Entry Dword 0

Bits	Description
31:16	Reserved
15:00	**Controller Data Queue Identifier (CDQID):** This field indicates the identifier for the Controller Data Queue created (refer to section 8.CDQ).

Controller Data Queue command specific status values (i.e., SCT field set to 1h) are shown in Figure CDQC-FIG10.

Figure CDQC-FIG10: Controller Data Queue – Command Specific Status Values

Value	Definition
1Fh	**Invalid Controller Identifier:** The Controller Identifier field contains an invalid value.
37h	**Invalid Controller Data Queue:** This error indicates that the specified Controller Data Queue Identifier is invalid for the controller processing the command.

…

5.15 Get Features command

…

Figure 195: Get Features – Feature Identifiers

Description	Section Defining Format of Attributes Returned
…	
Spinup Control	5.27.1.22

Figure 195: Get Features – Feature Identifiers

Description	Section Defining Format of Attributes Returned
Controller Data Queue	5.27.1.CDQF
…	

…

5.16 Get Log Page command

…

5.16.1 Log Specific Information

…

5.16.1.18 Feature Identifiers Supported and Effects (Log Page Identifier 12h)

…

Figure 257: FID Supported and Effects Data Structure

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:20</td><td><b>FID Scope (FSP):</b> This field defines the scope for the associated feature identifier. If the value of this field is 0h, then no scope is reported. If this field is non-zero, then only one bit shall be set to ‘1’.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>11: 7~~6~~</td><td>Reserved</td></tr>
<tr><td>6</td><td><b>Controller Data Queue (CDQSCP)</b>: If set to ‘1’, then modifying the attributes of the feature identified by this FID may impact a Controller Data Queue (refer to 8.CDQ). If cleared to ‘0’ and the FSP field is non-zero, then modifying the attributes of the feature identified by this FID does not impact a Controller Data Queue.</td></tr>
<tr><td>5</td><td><b>NVM Subsystem Scope</b>: If set to ‘1’, then modifying the attributes of the feature identified by this FID may impact the whole NVM subsystem. If cleared to ‘0’ and the FSP field is non-zero, then modifying the attributes of the feature identified by this FID does not impact the whole NVM subsystem.</td></tr>
<tr><td colspan="2">…</td></tr>
<tr><td>0</td><td><b>Namespace Scope</b>: If set to ‘1’, then modifying the attributes of the feature identified by this FID may impact namespaces. If cleared to ‘0’ and the FSP field is non-zero, then modifying the attributes of the feature identified by this FID does not impact namespaces.</td></tr>
</table></td></tr>
<tr><td colspan="2">…</td></tr>
</table>

…

5.17 Identify command

…

Figure 274: Identify – CNS Values

<table>
<tr><th>CNS Value</th><th>O/M[1]</th><th>Definition</th><th>NSID[2]</th><th>CNTID[3]</th><th>CSI[4]</th><th>Reference Section</th></tr>
<tr><td colspan="7"><b>Active Namespace Management</b></td></tr>
<tr><td colspan="7">…</td></tr>
</table>

Figure 274: Identify – CNS Values

CNS Value	O/M[1]	Definition	NSID[2]	CNTID[3]	CSI[4]	Reference Section
Controller and Namespace Management						
…						
1Ch	O	I/O Command Set data structure	N	Y	N	5.17.2.21
20h	O	Supported Controller State Formats	N	N	N	5.17.2.SCSF
21h~~18h~~ to 1Fh		Reserved				
Future Definition						
…						

…

5.17.2 Identify Data Structures

5.17.2.1 Identify Controller Data Structure (CNS 01h)

…

Figure 276: Identify – Identify Controller Data Structure, I/O Command Set Independent

<table>
<tr><th>Bytes</th><th>I/O[1]</th><th>Admin[1]</th><th>Disc[1]</th><th>Description</th></tr>
<tr><td colspan="5">…</td></tr>
<tr><td>255</td><td>M</td><td>M</td><td>M</td><td><b>Management Endpoint Capabilities (MEC):</b> This field indicates the capabilities of the Management Endpoint in the NVM subsystem. Refer to the NVM Express Management Interface Specification for details.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7:2</td><td>Reserved</td></tr>
<tr><td>1</td><td><b>PCIe Port Management Endpoint (PCIEME):</b> If set to ‘1’, then the NVM subsystem contains a Management Endpoint on a PCIe port.</td></tr>
<tr><td>0</td><td><b>SMBus/I2C Port Management Endpoint (SMBUSME):</b> If set to ‘1’, then the NVM subsystem contains a Management Endpoint on an SMBus/I2C port.</td></tr>
</table></td></tr>
<tr><td colspan="5"><b>Admin Command Set Attributes & Optional Controller Capabilities</b></td></tr>
</table>

Figure 276: Identify – Identify Controller Data Structure, I/O Command Set Independent

<table>
<tr><th>Bytes</th><th>I/O[1]</th><th>Admin[1]</th><th>Disc[1]</th><th>Description</th></tr>
<tr><td>257:256</td><td>M</td><td>M</td><td>R</td><td><b>Optional Admin Command Support (OACS):</b> This field indicates the optional Admin commands and features supported by the controller. Refer to section 3.1.2.

~~Bits 15:11 are reserved.~~

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>15:~~11~~12</td><td>Reserved</td></tr>
<tr><td>11</td><td><b>Host Managed Live Migration Support (HMLMS):</b> If this bit is set to ‘1’, then the controller supports the Host Managed Live Migration capability (refer to section 8.LM).

If this bit is cleared to ‘0’, then the controller does not support the Host Managed Live Migration capability.

Secondary controllers shall clear this bit to ‘0’.</td></tr>
</table>

<Note to reader: this field is already a table in the NEXT file but not a table in the NVM Express Base Specification 2.0d file. Just showing the new row entries to the table>

Bit 10 if set to ‘1’, then the controller supports the Command and Feature Lockdown capability (refer to section 8.4). If cleared to ‘0’, then the controller does not support the Command and Feature Lockdown capability. This value shall be the same for all controllers in the NVM subsystem.
…</td></tr>
<tr><td>258</td><td>M</td><td>M</td><td>R</td><td><b>Abort Command Limit (ACL):</b> This field is used to convey the maximum number of concurrently executing Abort commands supported by the controller (refer to section 5.1). This is a 0’s based value. It is recommended that implementations support concurrent execution of a minimum of four Abort commands.</td></tr>
<tr><td colspan="5"><b>…</b></td></tr>
<tr><td>275:272</td><td>O</td><td>O</td><td>R</td><td><b>Host Memory Buffer Preferred Size (HMPRE):</b> This field indicates the preferred size that the host is requested to allocate for the Host Memory Buffer feature in 4 KiB units. This value shall be greater than or equal to the Host Memory Buffer Minimum Size. If this field is non-zero, then the Host Memory Buffer feature is supported. If this field is cleared to 0h, then the Host Memory Buffer feature is not supported.

If the Host Managed Live Migration Support (HMLMS) bit is set to ‘1’ in one or more controllers in the NVM subsystem; then this field shall be cleared to 0h.</td></tr>
<tr><td colspan="5">…</td></tr>
<tr><td>357:356</td><td>O</td><td>O</td><td>O</td><td><b>Domain Identifier:</b> This field indicates the identifier of the domain (refer to section 3.2.4) that contains this controller. If the MDS bit is set to ‘1’, then this field shall be set to a non-zero value. If the NVM subsystem does not support multiple domains (i.e., the NVM subsystem consists of a single domain), then this field shall be cleared to 0h.</td></tr>
<tr><td colspan="5"><b>…</b></td></tr>
<tr><td>571:570</td><td>O</td><td>O</td><td>P</td><td><b>Controller Maximum Memory Range Tracking Descriptors (CMMRTD):</b> This field indicates the maximum number of Memory Range Tracking Descriptors (refer to Figure TMC-FIG2) the controller supports (refer to section 5.TSC.1.2).

If the THMCS bit is cleared to ‘0’, then this field shall be cleared to 0h and the host should ignore this field.</td></tr>
</table>

Figure 276: Identify – Identify Controller Data Structure, I/O Command Set Independent

<table>
<tr><th>Bytes</th><th>I/O[1]</th><th>Admin[1]</th><th>Disc[1]</th><th>Description</th></tr>
<tr><td>573:572</td><td>O</td><td>O</td><td>P</td><td><b>NVM Subsystem Maximum Memory Range Tracking Descriptors (NMMRTD):</b> This field indicates the maximum number of Memory Range Tracking Descriptors (refer to Figure TMC-FIG2) the NVM subsystem supports (refer to section 5.TSC.1.2).
If the THMCS bit is cleared to ‘0’, then this field shall be cleared to 0h and the host should ignore this field.</td></tr>
<tr><td>574</td><td>O</td><td>O</td><td>P</td><td><b>Minimum Memory Range Tracking Granularity (MINMRTG):</b> This field indicates the minimum value supported in the Requested Memory Range Tracking Granularity (RMRTG) field (refer to Figure TMC-FIG2) of the Track Memory Ranges data structure.
If the THMCS bit is cleared to ‘0’, then this field shall be cleared to 0h and the host should ignore this field.</td></tr>
<tr><td>575</td><td>O</td><td>O</td><td>P</td><td><b>Maximum Memory Range Tracking Granularity (MAXMRTG):</b> This field indicates the maximum value supported in the Requested Memory Range Tracking Granularity (RMRTG) field (refer to Figure TMC-FIG2) of the Track Memory Ranges data structure.
If the THMCS bit is cleared to ‘0’, then this field shall be cleared to 0h and the host should ignore this field.</td></tr>
<tr><td>576</td><td>O</td><td>O</td><td>P</td><td><b>Tracking Attributes (TRATTR):</b> This field indicates supported attributes for the Track Send command and Track Receive command.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7:3</td><td>Reserved</td></tr>
<tr><td>2</td><td><b>Memory Range Tracking Length Limit (MRTLL):</b> If this bit is set to ‘1’, then the controller only supports the length as specified by the Requested Memory Range Tracking Granularity field and the Length field in a Track Memory Changes data structure (refer to Figure TMC-FIG2) indicating a value that is a power of 2. If this bit is cleared to ‘0’, then the length indicated by the Requested Memory Range Tracking Granularity field and the Length field in a Track Memory Changes data structure is not required to be a value that is a power of 2.</td></tr>
<tr><td>1</td><td><b>Track User Data Changes Support (TUDCS):</b> If this bit is set to ‘1’, then the controller supports tracking user data changes as defined by section 5.TSC.1.1. If this bit is cleared to ‘0’, then the controller does not support tracking user data changes as defined by section 5.TSC.1.1.</td></tr>
<tr><td>0</td><td><b>Track Host Memory Changes Support (THMCS):</b> If this bit is set to ‘1’, then the controller supports tracked memory changes as defined by section 5.TRC.1.1 and section 5.TSC.1.2. If this bit is cleared to ‘0’, then the controller does not support the tracked memory changes as defined by section 5.TRC.1.1 and section 5.TSC.1.2.</td></tr>
</table></td></tr>
<tr><td>577</td><td></td><td></td><td></td><td>Reserved</td></tr>
<tr><td>579:578</td><td>O</td><td>O</td><td>P</td><td><b>Maximum Controller User Data Migration Queues (MCUDMQ):</b> This field indicates the maximum number of User Data Migration Queues supported by the controller (i.e., allowed to be created in this controller).
If the TUDCS bit is cleared to ‘0’, then this field shall be cleared to 0h and the host should ignore this field.</td></tr>
</table>

Figure 276: Identify – Identify Controller Data Structure, I/O Command Set Independent

Bytes	I/O[1]	Admin[1]	Disc[1]	Description
581:580	O	O	P	**Maximum NVM Subsystem User Data Migration Queues (MNSUDMQ):** This field indicates the maximum number of User Data Migration Queues supported by the NVM subsytem (i.e., allowed to be created in this NVM subsystem). If the TUDCS bit is cleared to '0', then this field shall be cleared to 0h and the host should ignore this field.
583:582	O	O	P	**Maximum CDQ Memory Ranges (MCMR):** This field indicates the maximum number of memory ranges allowed to be specified by the PRP1 field of a Controller Data Queue command. A value of 0h indicates that a maximum is not reported.
585:584	O	O	P	**NVM Subsystem Maximum CDQ Memory Ranges (NMCMR):** This field indicates the maximum number of memory ranges for all Controller Data Queues in the NVM subsystem. A value of 0h indicates that a maximum is not reported. The value of this field shall be greater than or equal to the value of the MCMR field.
587:586	O	O	P	**Maximum Controller Data Queue PRP Count (MCDQPC):** This field indicates the maximum number of PRPs allowed to be specified in the PRP list in the Controller Data Queue command. A value of 0h indicates that a maximum is not reported.
…				

…

5.17.2.SCSF Supported Controller State Formats (CNS 20h)

A Supported Controller State Formats data structure (refer to Figure SCSF-FIG1) is returned to the host identifying the supported NVMe Controller State data structures (refer to Figure SCS-FIG5) and a list of UUIDs that identifies the vendor specific controller state information.

A host uses an index into each list to identify the format and information returned by a Migration Receive command specifying the Get Controller State management operation (refer to section 5.MRC.1.1).

A host uses an index into each list to identify the format and information contained in a Migration Send command specifying the Set Controller State management operation of the Migration Send command (refer to section 5.MSC.1.3).

Figure SCSF-FIG1: Supported Controller State Formats Data Structure

Bytes	Description
Header	
0	**Number of Versions (NV):** This field indicates the number of entries in the NVMe Controller State Version list. A value of 0h indicates that no entries exist in the NVMe Controller State Version list.
1	**Number of UUIDs (NUUID):** This field indicates the number of entries in the Vendor Specific Controller State UUID Supported list. A value of 0h indicates that no entries exist in the Vendor Specific Controller State UUID Supported List.
NVMe Controller State Version list	
3:2	**NVMe Controller State Data Structure Version 1:** This field contains a version of the NVMe Controller State data structure (refer to Version field in Figure SCS-FIG6) that is supported by this controller and is associated with index 1h of this list, if any. This field is the first entry of the NVMe Controller State Version list.
5:4	**NVMe Controller State Data Structure Version 2:** This field contains a version of the NVMe Controller State data structure that is supported by this controller and is associated with index 2h of this list, if any. This field is the second entry of the NVMe Controller State Version list.

Figure SCSF-FIG1: Supported Controller State Formats Data Structure

Bytes	Description
…	
(NV*2)+1:(NV*2)	**NVMe Controller State Data Structure Version NV:** This field contains a version of the NVMe Controller State data structure that is supported by this controller and is associated with index NV of this list, if any. This field is the last entry of the NVMe Controller State Version list.
Vendor Specific Controller State UUID Supported List	
((NV+1)*2)+15: ((NV+1)*2)	**Vendor Specific Controller State UUID 1:** This field contains a 128-bit Universally Unique Identifier (UUID), as specified in RFC 9562 (refer to section 4.3.6), that identifies the format of the Vendor Specific Data field the Controller State data structure (refer to Figure SCS-FIG5) that is supported by this controller and is associated with index 1h of this list, if any. This field is the first entry of the Vendor Specific Controller State UUID Supported list. The value in this field and the associated format of the Vendor Specific Data field the Controller State data structure is out of scope for this specification.
((NV+1)*2)+31 : ((NV+1)*2)+16	**Vendor Specific Controller State UUID 2:** This field contains a 128-bit Universally Unique Identifier (UUID), as specified in RFC 9562, that identifies the format of the Vendor Specific Data field the Controller State data structure that is supported by this controller and is associated with index 2h of this list, if any. This field is the second entry of the Vendor Specific Controller State UUID Supported list. The value in this field and the associated format of the Vendor Specific Data field the Controller State data structure is out of scope for this specification.
…	
((NUUID)*16)+ ((NV+1)*2)-1 : ((NUUID-1)*16)+ ((NV+1)*2)	**Vendor Specific Controller State UUID NUUID:** This field contains a 128-bit Universally Unique Identifier (UUID), as specified in RFC 9562, that identifies the format of the Vendor Specific Data field the Controller State data structure that is supported by this controller and is associated with index NUUID of this list, if any. This field is the last entry of the Vendor Specific Controller State UUID Supported list. The value in this field and the associated format of the Vendor Specific Data field the Controller State data structure is out of scope for this specification.

…

5.MRC Migration Receive command

The Migration Receive command is used to obtain information from the controller processing the command that the host may use to manage a migratable controller (refer to section 8.LM).

The Migration Receive command uses the Data Pointer field, Command Dword 10 field, Command Dword 12 field, Command Dword 13 field, Command Dword 14 field, and Command Dword 15 field. The use of the Command Dword 11 field is specific to the management operation specified by the Select field. All other command specific fields are reserved.

The Select field defined in Figure MRC-FIG1 specifies the management operation to be performed. Refer to section 5.MRC.1 for a description of each management operation.

Figure MRC-FIG1: Migration Receive – Command Dword 10

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:16</td><td><b>Management Operation Specific (MOS):</b> The definition of this field is specific to a management operation (refer to the Select field). If a management operation does not define the use of this field, then this field is reserved.</td></tr>
<tr><td>15:08</td><td>Reserved</td></tr>
<tr><td>07:00</td><td><b>Select (SEL):</b> This field specifies the type of management operation to perform.
<table>
<tr><th>Value</th><th>M/O[1]</th><th>Management Operation</th><th>Reference Section</th></tr>
<tr><td>0h</td><td>M</td><td>Get Controller State</td><td>5.MRC.1.1</td></tr>
<tr><td>1h to FFh</td><td></td><td colspan="2">Reserved</td></tr>
<tr><td colspan="4">Notes:
1. O/M definition: O = Optional, M = Mandatory</td></tr>
</table>
</td></tr>
</table>

Figure MRC-FIG2: Migration Receive – Command Dword 12

Bits	Description
31:00	**Offset Lower (OL):** This field specifies the least-significant 32-bits of the offset, in bytes, within the data available to be returned and specifies the starting point for that data for what is actually returned to the host. The offset is required to be dword aligned and if bits 1:0 are not cleared to 00b, then the controller shall abort the command with a status code of Invalid Field in Command. If the host specifies an offset (i.e., OL field and OU field) that is greater than the size of the returned data requested (e.g., the returned data contains 100 bytes is requested starting at offset 200), then the controller shall abort the command with a status of Invalid Field in Command.

Figure MRC-FIG3: Migration Receive – Command Dword 13

Bits	Description
31:00	**Offset Upper (OU):** This field specifies the most-significant 32-bits of the offset, in bytes, within the data available to be returned and specifies the starting point for that data for what is actually returned to the host. Refer to the Offset Lower field definition in Figure MRC-FIG2.

If the controller supports selection of a UUID by the Migration Receive command (refer to Figure 212 and section 8.25), then Command Dword 14 is used to specify a UUID Index value (refer to Figure MRC-FIG4).

Figure MRC-FIG4: Migration Receive – Command Dword 14

Bits	Description
31:07	Reserved
06:00	**UUID Index (UIDX):** Refer to Figure 479.

Figure MRC-FIG5: Migration Receive – Command Dword 15

Bits	Description
31:00	**Number of Dwords (NUMDL):** This field specifies the number of dwords to return. If the host specifies a value that is greater than the number of dwords returned by the specified management operation of the command, then the controller returns the number of dwords defined by the specified management operation of the command with undefined results for the remaining dwords. This is a 0's based value.

5.MRC.1 Migration Receive Management Operations

5.MRC.1.1 Get Controller State (Management Operation 0h)

The Get Controller State management operation of the Migration Receive command allows the host to retrieve from the controller processing the command state data for the controller specified in the Controller Identifier field in the Management Operation Specific field (refer to Figure CS-FIG1). The data returned in the data buffer is the Controller State data structure as defined by Figure SCS-FIG5.

If the controller specified in the Controller Identifier field in the Management Operation Specific field (i.e., the specified controller):

- is suspended (refer to section 5.MSC.1.1); and
- remains suspended during the processing of the Migration Receive command,

then:

- the specified controller has stopped processing commands; and
- if:
  - a Controller Level Reset has not occurred on the specified controller during the processing of the Migration Receive command; or
  - the host does not modify controller properties on the specified controller during the processing of the Migration Receive command,

  then the returned Controller State data structure is consistent between each field reported in that data structure.

If the specified controller:

- is not suspended during the processing of the Migration Receive command; or
- has a Controller Level Reset during the processing of the Migration Receive command,

then the specified controller state may be changing during the processing of the Migration Receive command and:

- each field in the reported Controller State data structure contains the value that reflects the state of the specified controller at the time the value was obtained by the controller processing the Migration Receive command; and
- state of the specified controller may be at a different state when a value is obtained between different fields (i.e., the returned Controller State data structure may or may not be internally consistent between each field).

The Get Controller State management operation uses the Management Operation Specific field as defined in Figure CS-FIG1 and the Command Dword 11 field as defined in Figure CS-FIG2.

Figure CS-FIG1: Get Controller State Management Operation – Management Operation Specific

Bits	Description
15:08	Reserved
07:00	**Controller State Version Index (CSVI):** A non-zero value in this field specifies the index to a specific entry in the NVMe Controller State Version list of the Supported Controller State Formats data structure (refer to section 5.17.2.SCSF). The contents of that entry specify the format of the Controller State field in the returned data. If this field is cleared to 0h, then no Controller State field is returned in the returned data.

The Get Controller State management operation of the Migration Receive command uses Command Dword 11 as defined in Figure CS-FIG2.

Figure CS-FIG2: Get Controller State Management Operation – Command Dword 11

Bits	Description
31:24	**Controller State UUID Index Parameter (CSUIDXP)**: This field is vendor specific. If the Controller State UUID Index field is cleared to 0h, then the controller shall ignore this field.
23:16	**Controller State UUID Index (CSUUIDI):** A non-zero value in this field specifies the index to a specific entry in the Vendor Specific Controller State UUID Supported list of the Supported Controller State Formats data structure (refer to section 5.17.2.SCSF). The contents of that entry specify the format of the Vendor Specific field in the returned data. If this field is cleared to 0h, then no Vendor Specific field is returned in the returned data.
15:00	**Controller Identifier (CNTLID):** This field specifies the identifier of the controller on which the management operation is performed.

If the CSVI field (refer to Figure CS-FIG2) specifies a non-zero index that is not defined by the Supported Controller State Formats data structure (refer to Figure SCSF-FIG1), then the controller shall abort the command with a status code of Invalid Field in Command.

If the CSVI field is cleared to 0h, then the Controller State Size field in the returned data shall be cleared to 0h.

If the CSUUIDI field specifies a non-zero index that is not defined by the Supported Controller State Formats data structure, then the controller shall abort the command with a status code of Invalid Field in Command.

If the CSUUIDI field is cleared to 0h, then the Vendor Specific Size field in the returned data shall be cleared to 0h.

If a status code of Successful Completion is returned, then the completion queue entry Dword 0 contains additional information about the command as defined in Figure CS-FIG3.

Figure CS-FIG3: Get Controller State Management Operation – Completion Queue Entry Dword 0

Bits	Description
31:01	Reserved
00	**Controller Suspended (CSUP):** If this bit is set to '1', then the controller associated with the returned data structure was suspended for the entire duration of the processing of this command. If this bit is cleared to '0', then the controller associated with this data structure was not suspended for the entire duration of the processing of this command.

5.MRC.2 Command Completion

Upon completion of the Migration Receive command, the controller posts a completion queue entry to the Admin Completion Queue indicating the status for the command. Section 5.MRC.1 describes completion details for each management operation.

Migration Receive command specific status values (i.e., SCT field set to 1h) are shown in Figure MRC-CC.

Figure MRC-CC: Migration Receive – Command Specific Status Values

Value	Definition
1Fh	**Invalid Controller Identifier:** The specified controller is not in a condition to set the controller state.

5.MSC Migration Send command

The Migration Send command is used to manage the migration of a controller (refer to section 8.LM).

The Migration Send command uses the Command Dword 10 and Command Dword 14. The use of the Data Pointer field, Command Dword 11 field, Command Dword 12 field, Command Dword 13 field, and Command Dword 15 field is specific to the management operation specified by the Select field. All other command specific fields are reserved.

The Select field defined in Figure MSC-FIG1 specifies the management operation to be performed. Refer to section 5.MSC.1 for a description of each management operation.

Figure MSC-FIG1: Migration Send – Command Dword 10

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:16</td><td><b>Management Operation Specific (MOS):</b> The definition of this field is specific to a management operation (refer to the Select field). If a management operation does not define the use of this field, then this field is reserved.</td></tr>
<tr><td>15:08</td><td>Reserved</td></tr>
<tr><td>07:00</td><td><b>Select (SEL):</b> This field specifies the type of management operation to perform.
<table>
<tr><th>Value</th><th>M/O[1]</th><th>Management Operation</th><th>Reference Section</th></tr>
<tr><td>0h</td><td>M</td><td>Suspend</td><td>5.MSC.1.3</td></tr>
<tr><td>1h</td><td>M</td><td>Resume</td><td>5.MSC.1.4</td></tr>
<tr><td>2h</td><td>M</td><td>Set Controller State</td><td>5.MSC.1.5</td></tr>
<tr><td>3h to FFh</td><td></td><td colspan="2">Reserved</td></tr>
<tr><td colspan="4">Notes:
1. O/M definition: O = Optional, M = Mandatory</td></tr>
</table>
</td></tr>
</table>

If the controller supports selection of a UUID by the Migration Send command (refer to Figure 211 and section 8.25), then Command Dword 14 is used to specify a UUID Index value (refer to Figure MSC-FIG2).

Figure MSC-FIG2: Migration Send – Command Dword 14

Bits	Description
31:07	Reserved
06:00	**UUID Index (UIDX):** Refer to Figure 479.

5.MSC.1 Migration Send Management Operations

5.MSC.1.1 Suspend (Management Operation 0h)

The Suspend management operation of the Migration Send command allows a host to specify the type of suspend to be done on the specified controller.

The Suspend management operation uses the Command Dword 11 field as defined in Figure SUSPEND-FIG1 and does not use the Management Operation Specific field.

Figure SUSPEND-FIG1: Suspend – Command Dword 11

Bits	Description
31	**Delete User Data Migration Queue (DUDMQ):** If this bit is set to '1' and the command completes successfully, then the User Data Migration Queue associated with the controller specified in the CNTLID field, if any, is deleted (refer to the applicable I/O Command Set specification). If this bit is cleared to '0', then the User Data Migration Queue associated the controller specified in the CNTLID field is retained after processing the command, if any.
30:24	Reserved

Figure SUSPEND-FIG1: Suspend – Command Dword 11

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>23:16</td><td><b>Suspend Type (STYPE):</b> This field specifies the type of suspend.
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>0h</td><td><b>Suspend Notification:</b> The specified controller is going to be suspended in the future with a subsequent Migration Send command specifying a Suspend management operation and a non-zero value in the Suspend Type field.</td></tr>
<tr><td>1h</td><td><b>Suspend:</b> Suspend the controller specified in the CNTLID field as defined in this section.</td></tr>
<tr><td>All Others</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>15:00</td><td><b>Controller Identifier (CNTLID):</b> This field specifies the identifier of the controller to which the Suspend management operation is performed.</td></tr>
</table>

A Suspend Type field value of 0h (i.e., Suspend Notification) allows a host to notify a controller that the controller specified by the Controller Identifier field is going to be requested to suspend with a subsequent Migration Send command specifying a Suspend management operation and a non-zero value in the Suspend Type field.

A Suspend Type field value of 1h (i.e., Suspend) specifies that the controller specified in the Controller Identifier field (refer to Figure SUSPEND-FIG1) is to be suspended. That controller shall perform the following actions and then stop initiating transport transactions:

1. Stop fetching commands until the specified controller is no longer suspended (refer to section 8.LM).
2. Complete the processing of all previously fetched commands, if any, except any outstanding Asynchronous Event Request commands.

If:

- the command causes the specified controller to be suspended; and
- there is a User Data Migration Queue associated with that specified controller,

then:

- If the DUDMQ bit is set to ‘1’, then the controller processing the command before posting the completion queue entry to the command shall delete that User Data Migration Queue; and
- after the specified controller completes the processing of all previously fetched commands, if any, except any outstanding Asynchronous Event Request commands, the controller processing the Migration Send command shall, at a vendor specific amount of time, posts an entry in that User Data Migration Queue that indicates the specified controller is suspended as specified by the applicable I/O Command Set specification.

It is not an error if a suspend operation occurs on a controller that is already suspended with the same Suspend Type value.

If the result of this command causes the specified controller to be suspended, then the specified controller remains suspended until:

- a Migration Send command specifying the Resume management operation (refer to section 5.MSC.1.2) on the specified controller is completed successfully; or
- a Controller Level Reset occurs on the controller processing this command.

If a controller is suspended, then:

- accesses to controller properties for that suspended controller shall behave normally except a host modification of the CC property that sets CC.EN bit from ‘0’ to ‘1’ (i.e., enabling that suspended controller) has implementation specific behavior (i.e., the suspended controller may or may not set the CSTS.RDY bit). A host should avoid modifications to controller properties, CMB, and PMR to a controller that is suspended;

- the Controller Memory Buffer (CMB), if supported, shall behave as defined in section 8.5 (i.e., CMB on a suspended controller continues to function normally); and
- the Persistent Memory Region (PMR), if supported, shall behave as defined in section 8.14 (i.e., PMR on a suspended controller continues to function normally).

5.MSC.1.2 Resume (Management Operation 1h)

The Resume management operation of the Migration Send command is a request for the controller processing the command to resume the operation of a different controller that is suspended (refer to section 5.MSC.1.1).

The Resume management operation uses the Command Dword 11 field as defined in Figure RESUME-FIG7 and does not use the Management Operation Specific field.

Figure RESUME-FIG7: Resume – Command Dword 11

Bits	Description
31:16	Reserved
15:00	**Controller Identifier (CNTLID):** This field specifies the identifier of the controller to resume operations.

The Resume management operation shall cause the controller specified by the Controller Identifier field (refer to Figure Resume-FIG7) to start fetching and processing commands.

If the specified controller is not suspended (refer to section 5.MSC.1.1), then the controller shall abort the command with a status code of Controller Not Suspended.

If the controller processing the command has received controller state for the specified controller to be resumed and that controller state has not been verified and committed (refer to section 5.MSC.1.3), then the controller shall abort the command with a status code of Command Sequence Error.

5.MSC.1.3 Set Controller State (Management Operation 2h)

The Set Controller State management operation of the Migration Send command allows the host to set the state of the controller specified by the Controller Identifier (CNTLID) field in Command Dword 11 (refer to Figure SCS-FIG1).

The data buffer contains the Controller State data structure specified in Figure SCS-FIG5 which specifies the state to be set in the specified controller.

To set the controller state of the specified controller, the specified controller is required to be in one or more of these conditions:

- Suspended (refer to section 5.MSC.1.1);
- enabled (i.e., CC.EN bit is set to ‘1’); or
- offline, if that specified controller is a secondary controller.

If the specified controller is not in one of these conditions, then the controller processing the command shall abort the command with a status code of Invalid Controller Identifier.

The host may submit one or more Migration Send commands to transfer the controller state. If more than one Migration Send command is submitted to transfer the controller state, then:

- The host first submits a Migration Send command with:
  - the Select field set to Set Controller State (i.e., 2h);
  - the Sequence Indicator (SEQIND) field (refer to Figure SCS-FIG1) set to 01b; and
  - the Controller Identifier field set to the controller whose state is being set,

  to specify to the controller that the command is the first of a sequence of Migration Send commands of controller state data being transferred.
- The host may submit Migration Send commands with:

- ○ the Select field set to Set Controller State (i.e., 2h);
  - ○ the SEQIND field cleared to 00b; and
  - ○ the Controller Identifier field set to the controller whose state is being set,

  to specify to the controller that the command is not the last of the sequence of Migration Send commands of controller state data being transferred, if any.
- The host is required to wait for the completion queue entries to be posted for the previous submitted Migration Send commands in the sequence of Migration Send commands.
- Finally, the host submits the last Migration Send command with:
  - ○ the Select field set to Set Controller State (i.e., 2h);
  - ○ the SEQIND field set to 10b; and
  - ○ the Controller Identifier field set to the controller whose state is being set,

to specify to the controller that this is the last command of the sequence of Migration Send commands of the controller state data being transferred. This last Migration Send command may or may not have the Number of Dwords (NUMD) field cleared to 0h.

If a sequence of Migration Send commands of the controller state data being transferred and the controller processes a Migration Send command with:

- the Select field set to Set Controller State (i.e., 2h);
- the Sequence Indicator (SEQIND) field (refer to Figure SCS-FIG1) set to 01b; and
- the Controller Identifier field set to the controller whose state is being set,

then the command is specifying a new sequence of Migration Send commands of the controller state data is being transferred and the previous sequence of Migration Send commands of the controller state data is discarded.

If a sequence of Migration Send commands of the controller state data is not being transferred to the specified controller and the controller processes a Migration Send command with the Sequence Indicator (SEQIND) field (refer to Figure SCS-FIG1) not set to 01b, clearer to 0b; the controller shall abort that command with a status code of Command Sequence Error.

If a single Migration Send command is submitted to transfer the entire controller state, then the host submits a Migration Send command with the SEQIND field set to 11b.

Figure SCS-FIG1: Set Controller State – Management Operation Specific Field

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>15:02</td><td>Reserved</td></tr>
<tr><td>01:00</td><td><b>Sequence Indicator (SEQIND):</b> This field identified the sequences of this Migration Send command in relation to other Migration Send commands.
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>00b</td><td>This command is not the first or last of a sequence of two or more Migration Send commands with this management operation used to transfer the controller state from the host to the controller.</td></tr>
<tr><td>01b</td><td>This command is the first of a sequence of two or more Migration Send commands with this management operation used to transfer the controller state from the host to the controller<b>.</b></td></tr>
<tr><td>10b</td><td>This Migration Send command is the last command of a sequence of two or more Migration Send commands with this management operation used to transfer the controller state from the host to the controller.</td></tr>
<tr><td>11b</td><td>This Migration Send command is the only command and contains the entire controller state for this management operation used to transfer the controller state from the host to the controller.</td></tr>
</table></td></tr>
</table>

Figure SCS-FIG2: Set Controller State – Command Dword 11

Bits	Description
31:24	**Controller State UUID Index (CSUUIDI):** A non-zero value in this field specifies the index to a specific entry in the Vendor Specific Controller State UUID Supported list of the Supported Controller State Formats data structure (refer to section 5.17.2.SCSF). The contents of that entry specify the format of the Vendor Specific field in the specified Controller State data structure. If this field is cleared to 0h, then no Vendor Specific field is contained in the Controller State data structure.
23:16	**Controller State Version Index (CSVI):** A non-zero value in this field specifies the index to a specific entry in the NVMe Controller State Version list of the Supported Controller State Formats data structure (refer to section 5.17.2.SCSF). The contents of that entry specify the format of the Controller State field in the specified Controller State data structure. If this field is cleared to 0h, then no NVMe Controller State field is specified in the Controller State data structure.
15:00	**Controller Identifier (CNTLID):**. This field specifies the identifier of the controller whose state is being set.

Figure SCS-FIG3: Set Controller State – Command Dword 12

Bits	Description
31:00	**Controller State Offset Lower (CSOL):** This field specifies the least-significant 32-bits of starting offset within the Controller State data structure that the host is setting. The offset is required to be dword aligned and if bits 1:0 are not cleared to 00b, then the controller shall abort the command with a status code of Invalid Field in Command.

Figure SCS-FIG4: Set Controller State – Command Dword 13

Bits	Description
31:00	**Controller State Offset Upper (CSOU):** This field specifies the most-significant 32-bits of starting offset within the Controller State data that the host is setting. Refer to the CSOL field definition in Figure SCS-FIG3.

Figure SCS-FIG4: Set Controller State – Command Dword 15

Bits	Description
31:00	**Number of Dwords (NUMD):** This field specifies the number of dwords being transferred. This field is only allowed to be cleared to the value of 0h if the Sequence Indicator field is set to 10b.

If the host specifies an offset (i.e., the CSLO field and the CSUO field) that is greater than the size of the Controller State data, then the controller shall abort the command with a status code of Invalid Field in Command.

Figure SCS-FIG5: Controller State Data Structure

Bytes	Description
Header	
01:00	**Version (VER):** This field indicates the version of this data structure. This field shall be cleared to 0h.

Figure SCS-FIG5: Controller State Data Structure

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>02</td><td><b>Controller State Attributes (CSATTR):</b> This field specifies attributes associated with the controller state.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>07:01</td><td>Reserved</td></tr>
<tr><td>00</td><td><b>Controller Suspended (CP):</b> If this bit is set to ‘1’, then the controller associated with this data structure was suspended for the entire duration of the processing of the Migration Receive command that reported in contents this data structure. If this bit is cleared to ‘0’, then the controller associated with this data structure was not suspended for the entire duration of the processing of the Migration Receive command that reported the contents in this data structure.</td></tr>
</table></td></tr>
<tr><td>15:03</td><td>Reserved</td></tr>
<tr><td>31:16</td><td><b>NVMe Controller State Size (NVMECSS):</b> This field indicates the number of dwords in the NVMe Controller State field.</td></tr>
<tr><td>47:32</td><td><b>Vendor Specific Size (VSS):</b> This field indicates the number of dwords in the Vendor Specific Data field.</td></tr>
<tr><th colspan="2">Controller State Data</th></tr>
<tr><td>(NVMECSS *4)+47:48</td><td><b>NVMe Controller State (NVMECS):</b> If the NVMECSS field is non-zero, then this field contains the controller state data as defined in Figure SCS-FIG6. If the NVMECSS field is cleared to 0h, then this field does not exist in the data structure.</td></tr>
<tr><td>((VSS+ NVMECSS)*4)+47:
(NVMECSS *4)+48</td><td><b>Vendor Specific Data (VSD):</b> If the VSS field is non-zero, then this field contains vendor specific data. If the VSS field is cleared to 0h, then this field does not exist in the data structure.</td></tr>
</table>

If the CSVI field (refer to Figure SCS-FIG2) specifies a non-zero index that is not defined by the Supported Controller State Formats data structure (refer to Figure SCSF-FIG1), then the controller shall abort the command with a status code of Invalid Field in Command.

If the CSVI field (refer to Figure SCS-FIG2) is cleared to 0h and the NVMe Controller State Size field is non-zero, then the controller shall abort the command with a status code of Invalid Field in Command.

If the CSUUIDI field (refer to Figure SCS-FIG2) specifies a non-zero index that is not defined by the Supported Controller State Formats data structure, then the controller shall abort the command with a status code of Invalid Field in Command.

If the NVMe Controller State field exists (i.e., the NVMECSS field is non-zero) and any I/O Submission Queue or I/O Completion Queue exists in the controller specified by the Controller Identifier (CNTLID) field, then the controller shall abort the command with a status code of Invalid Field in Command.

Figure SCS-FIG6 defines the NVMe Controller State data structure that identifies the state of the all the I/O Submission Queues and I/O Completion Queues for a controller. Any controller state defined by this specification that is not included in the NVMe Controller State data structure is either included in the Vendor Specific Data field in the Controller State data structure (refer to Figure SCS-FIG5) or is obtained in a vendor specific manner.

Figure SCS-FIG6: NVMe Controller State Data Structure

Bytes	Description
Header	
01:00	**Version (VER):** This field indicates the version of this data structure. This field shall be cleared to 0h.
03:02	**Number of I/O Submission Queues (NIOSQ):** This field indicates the number I/O Submission Queues contained in this data structure.

Figure SCS-FIG6: NVMe Controller State Data Structure

Bytes	Description
05:04	**Number of I/O Completion Queues (NIOCQ):** This field indicates the number I/O Completion Queues contained in this data structure.
07:06	Reserved
I/O Submission Queue List	
31:08	**I/O Submission Queue State 0:** This field contains the state of the first I/O Submission Queue reported, if any. The contents of this field are defined in the I/O Submission Queue State data structure (refer to Figure SCS-FIG7).
55:32	**I/O Submission Queue State 1:** This field contains the state of the second I/O Submission Queue reported, if any. The contents of this field are defined in the I/O Submission Queue State data structure (refer to Figure SCS-FIG7).
…	
((NIOSQ) * 24) + 7: ((NIOSQ-1) * 24) + 8	**I/O Submission Queue State NIOSQ-1:** This field contains the state of the last I/O Submission Queue reported, if any. The contents of this field are defined in the I/O Submission Queue State data structure (refer to Figure SCS-FIG7).
I/O Completion Queue List	
((NIOSQ+1) * 24) + 7: ((NIOSQ) * 24) + 8	**I/O Completion Queue State 0:** This field contains the state of the first I/O Completion Queue reported, if any. The contents of this field are defined in the I/O Completion Queue State data structure (refer to Figure SCS-FIG8).
((NIOSQ+2) * 24) + 7: ((NIOSQ+1) * 24) + 8	**I/O Completion Queue State 1:** This field contains the state of the second I/O Completion Queue reported, if any. The contents of this field are defined in the I/O Completion Queue State data structure (refer to Figure SCS-FIG8).
…	
((NIOCQ) * 24) + ((NIOSQ) * 24) + 15: ((NIOCQ-1) * 24) + ((NIOSQ) * 24) + 8	**I/O Completion Queue State NIOSQ-1:** This field contains the state of the last I/O Completion Queue reported, if any. The contents of this field are defined in the I/O Completion Queue State data structure (refer to Figure SCS-FIG8).

The I/O Submission Queue list shall be listed in ascending order by I/O Submission Queue Identifier.

The I/O Completion Queue list shall be listed in ascending order by I/O Completion Queue Identifier.

Figure SCS-FIG7: I/O Submission Queue State Data Structure

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>07:00</td><td><b>I/O Submission PRP Entry 1 (IOSQPRP1):</b> This field contains the contents of the PRP1 field (refer to Figure 160) from the Create I/O Submission Queue command that created this I/O Submission Queue.</td></tr>
<tr><td>09:08</td><td><b>I/O Submission Queue Size (IOSQQSIZE):</b> This field contains the contents of the QSIZE field (refer to Figure 161) from the Create I/O Submission Queue command that created this I/O Submission Queue.</td></tr>
<tr><td>11:10</td><td><b>I/O Submission Queue Identifier (IOSQQID):</b> This field contains the contents of the QID field (refer to Figure 161) from the Create I/O Submission Queue command that created this I/O Submission Queue.</td></tr>
<tr><td>13:12</td><td><b>I/O Completion Queue Identifier (IOSQCQID):</b> This field contains the contents of the CQID field (refer to Figure 162) from the Create I/O Submission Queue command that created this I/O Submission Queue.</td></tr>
<tr><td>15:14</td><td><b>I/O Submission Queue Attributes (IOSQA):</b> This field contains various attributes associated with the I/O Submission Queue.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>15:03</td><td>Reserved</td></tr>
<tr><td>02:01</td><td><b>I/O Submission Queue Queue Priority (IOSQQPRIO):</b> This field contains the contents of the QPRIO field (refer to Figure 162) from the Create I/O Submission Queue command that created this I/O Submission Queue.</td></tr>
<tr><td>00</td><td><b>I/O Submission Queue Physically Contiguous (IOSQPC):</b> This bit contains the contents of the PC bit (refer to Figure 162) from the Create I/O Submission Queue command that created this I/O Submission Queue.</td></tr>
</table></td></tr>
<tr><td>17:16</td><td><b>I/O Submission Queue Head Pointer (IOSQHP):</b> This field indicates the value of the I/O Submission Queue head pointer.</td></tr>
</table>

Figure SCS-FIG7: I/O Submission Queue State Data Structure

Bytes	Description
19:18	**I/O Submission Queue Tail Pointer (IOSQTP):** This field indicates the value of the I/O Submission Queue tail pointer.
23:20	Reserved

Figure SCS-FIG8: I/O Completion Queue State Data Structure

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>07:00</td><td><b>I/O Completion Queue Size PRP Entry 1 (IOCQPRP1):</b> This field contains the contents of the PRP1 field (refer to Figure 156) from the Create I/O Completion Queue command that created this I/O Completion Queue.</td></tr>
<tr><td>09:08</td><td><b>I/O Completion Queue Size (IOCQQSIZE):</b> This field contains the contents of the QSIZE field (refer to Figure 157) from the Create I/O Completion Queue command that created this I/O Completion Queue.</td></tr>
<tr><td>11:10</td><td><b>I/O Completion Queue Identifier (IOCQQID):</b> This field contains the contents of the QID field (refer to Figure 157) from the Create I/O Completion Queue command that created this I/O Completion Queue.</td></tr>
<tr><td>13:12</td><td><b>I/O Completion Queue Head Pointer (IOCQHP):</b> This field indicates the value of the I/O Completion Queue head pointer.</td></tr>
<tr><td>15:14</td><td><b>I/O Completion Queue Tail Pointer (IOCQTP):</b> This field indicates the value of the I/O Completion Queue tail pointer.</td></tr>
<tr><td>19:16</td><td><b>I/O Completion Queue Attributes (IOCQA):</b> This field contains various attributes associated with the I/O Completion Queue.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:16</td><td><b>I/O Completion Queue Interrupt Vector (IOCQIV):</b> This field contains the contents of the IV field (refer to Figure 158) from the Create I/O Completion Queue command that created this I/O Completion Queue.</td></tr>
<tr><td>15:03</td><td>Reserved</td></tr>
<tr><td>02</td><td><b>Slot 0 Phase Tag (S0PT):</b> This bit states the value of the Phase Tag bit for slot 0 of this I/O Completion Queue.</td></tr>
<tr><td>01</td><td><b>I/O Completion Queue Interrupts Enabled (IOCQIEN):</b> This bit contains the contents of the IEN bit (refer to Figure 158) from the Create I/O Completion Queue command that created this I/O Completion Queue.</td></tr>
<tr><td>00</td><td><b>I/O Completion Queue Physically Contiguous (IOCQPC):</b> This bit contains the contents of the PC bit (refer to Figure 158) from the Create I/O Completion Queue command that created this I/O Completion Queue.</td></tr>
</table></td></tr>
<tr><td>23:20</td><td>Reserved</td></tr>
</table>

If the CSVI field (refer to Figure SCS-FIG2) is cleared to 0h and the NVMe Controller State Size field (refer to Figure SCS-FIG5) is not cleared to 0h, then the controller shall abort the command with a status code of Invalid Field in Command.

If the CSUUIDI field (refer to Figure SCS-FIG2) is cleared to 0h and the Vendor Specific Size field (refer to Figure SCS-FIG5) is not cleared to 0h, then the controller shall abort the command with a status code of Invalid Field in Command.

If the CSVI field (refer to Figure SCS-FIG2) is cleared to 0h and the CSUUIDI field (refer to Figure SCS-FIG2) is cleared to 0h, then the controller shall abort the command with a status code of Invalid Field in Command.

If the Migration Send command for this management operation specifies the SEQIND field set to 10b and that command completes successfully, then the controller state shall be verified and committed to the specified controller which includes:

- Creating any I/O Submission Queues and I/O Completion Queues specified by the NVMe Controller State field; and
- setting the queue state for each I/O Submission Queues and I/O Completion Queues specified by the NVMe Controller State field.

If any Migration Send command for this management operation is not successful, then the host should attempt to re-transfer the entire controller state data so that any existing controller state is overwritten with the desired controller state.

5.MSC.2 Command Completion

Upon completion of the Migration Send command, the controller posts a completion queue entry to the Admin Completion Queue indicating the status for the command. Section 5.MSC.1 describes completion details for each management operation.

Migration Send command specific status values (i.e., SCT field set to 1h) are shown in Figure MSCC-FIG1.

Figure MSCC-FIG1: Migration Send – Command Specific Status Values

Value	Definition
1Fh	**Invalid Controller Identifier:** The specified controller is not in a condition to set the controller state.
37h	**Invalid Controller Data Queue:** This error indicates that the specified Controller Data Queue Identifier is invalid for the controller processing the command.
38h	**Not Enough Resources:** This error indicates that there is not enough resources in the controller to process the command.
3Ah	**Controller Not Suspended:** The operation requested is not allowed if the specified controller is not suspended (refer to section 5.MSC.1.1).

…

5.24 Sanitize command

…

Activation of new firmware is prohibited during a sanitize operation (refer to section 8.21.1).

If one or more controllers in the NVM subsystem is suspended (refer to section 5.MSC.1.1), then the controller shall abort the command with a status code of Controller Suspended.

…

Figure 306: Sanitize – Command Specific Status Values

Value	Description
…	
23h	**Sanitize Prohibited While Persistent Memory Region is Enabled:** A sanitize operation is prohibited while the Persistent Memory Region is enabled.
39h	**Controller Suspended:** One or more controllers in the NVM subsystem have been suspended by a Migration Send command.

…

5.27 Set Features command

…

.5.27.1 Feature Specific Information

…

Figure 317: Set Features – Feature Identifiers

Feature Identifier	Current Setting Persists Across Power Cycle and Reset [2]	Uses Memory Buffer for Attributes	Feature Name	Scope[6]
…				
1Ah	Yes	No	Spinup Control	NVM subsystem
21h	No	Yes	Controller Data Queue	Controller Data Queue per controller
22h~~1Bh~~ to 1Fh	Reserved			
…				

Notes:
1. The behavior of a controller in response to an inactive namespace ID to a vendor specific Feature Identifier is vendor specific.
2. This column is only valid if the feature is not saveable (refer to section 4.2). If the feature is saveable, then this column is not used.
3. The controller does not save settings for the Host Memory Buffer feature across power states and reset events, however, host software may restore the previous values. Refer to section 8.9.
4. The feature does not use a memory buffer for Set Features commands and does use a memory buffer for Get Features commands. Refer to section 8.9.
5. Selection of a UUID may be supported. Refer to section 8.25.
6. Refer to Feature Identifiers Supported and Effects log page in section 5.16.1.18 for how scope is reported to the host.
7. For NVM Subsystems with multiple controllers in the same domain, specifying different power states results in an unspecified power state for that domain.

…

5.27.1.CDQF Controller Data Queue (Feature Identifier 21h)

This Feature allows a host to update the status of the head pointer of a Controller Data Queue (CDQ) and specify the configuration of a Controller Data Queue Tail event. The CDQ is specified by the Controller Data Queue Identifier (CDQID) field in Command Dword 11 (refer to Figure CDQF-FIG1).

The Head Pointer field specifies the current slot of the head pointer in the queue (refer to section 8.CDC). The controller uses this value to determine if CDQ entries have been freed by the host.

If the Enable Tail Pointer Trigger (ETPT) bit is set to ‘1’, then when the slot specified by the Tail Pointer Trigger (TPT) field for the specified CDQ is posted with an entry, the controller shall generate a Controller Data Queue Tail Pointer event to the host (refer to section 8.CDQ).

If the Set Features command is successful and there is a pending Controller Data Queue Tail Pointer event for the specified Controller Data Queue specified by the CDQID field, then the controller shall clear that pending event.

A controller should report Controller Data Queue Tail Pointer events in the order of occurrence to avoid reporting the same Controller Data Queue when a Controller Data Queue Tail Pointer event for that Controller Data Queue is being repeatedly triggered.

If the Enable Tail Pointer Trigger (ETPT) bit is set to ‘1’ in the current value of this Feature (refer to section 4.2) and the controller processes a Set Features command for this Feature that specifies:

- the Enable Tail Pointer Trigger (ETPT) bit set to ‘1’; and
- a slot in the Tail Pointer Trigger (TPT) field,

then prior to posting the completion for that Set Features command, the controller may post an entry into the specified CDQ in the slot specified by the value of the TPT field that had been set prior to processing that Set Features command. To detect this condition, after receiving the completion of that Set Features command, a host should examine the CDQ to determine if the requested tail pointer trigger has already occurred. If the requested tail pointer trigger has occurred, then a subsequent Set Features command should be submitted by the host to disable the Tail Pointer Trigger event or request a different tail pointer trigger.

If a Get Features command is submitted for this Feature, the attributes described in Figure CDQF-FIG1 are returned in Dword 0 and the attributes described in Figure CDQF-FIG3 are returned in the data buffer for that command.

Figure CDQF-FIG1: Controller Data Queue – Command Dword 11

Bits	Description
31	**Enable Tail Pointer Trigger (ETPT):** If this bit is set to ‘1’, then the controller is to generate a Controller Data Queue Tail Pointer event when that controller posts the entry into the slot specified by the TPT field for the specified CDQ. If this bit is cleared to ‘0’, then there is no request for a Controller Data Queue Tail Pointer event to be sent by the controller. If a Controller Data Queue Tail Pointer event is generated for the specified Controller Data Queue, then the current value of this bit for this Feature shall be cleared to ’0’ (i.e., the current value for this bit only enables a single occurrence of a Controller Data Queue Tail Pointer event for the specified Controller Data Queue). For a Get Features command, this field shall be ignored by the controller.
30:16	Reserved
15:00	**Controller Data Queue Identifier (CDQID):** This field contains the identifier associated with the CDQ.

Figure CDQF-FIG2: Controller Data Queue – Command Dword 12

Bits	Description
31:00	**Head Pointer (HP):** This field specifies the slot of the head pointer for the specified CDQ.

Figure CDQF-FIG2a: Controller Data Queue – Command Dword 13

Bits	Description
31:00	**Tail Pointer Trigger (TPT):** If the ETPT bit is set to ‘1’, then this field specifies a slot in the CDQ that when posted with an entry causes the controller to issue a Controller Data Queue Tail Pointer event. For a Set Features command, if the ETPT bit is cleared to ‘0’, then this field shall be ignored by the controller. For a Get Features command, this field shall be ignored by the controller.

If the CDQ is empty (refer to section 3.3.1.4) and the Head Pointer field specifies a value that is not the same value as the current value, then the controller shall abort the command with a status code of Invalid Field in Command.

If the CDQ is not empty and the Head Pointer field specifies a slot not associated with an entry that was posted by the controller within the specified CDQ, then the controller shall abort the command with a status code of Invalid Field in Command.

If the Enable Tail Pointer Trigger bit is set to ‘1’ and the Tail Pointer Trigger field specifies a slot not associated with an entry within the specified CDQ, then the controller shall abort the command with a status code of Invalid Field in Command.

If the value in the Controller Data Queue Identifier field specifies a CDQ that does not exist in the controller processing the command, then the controller shall abort the command with a status code of Invalid Controller Data Queue.

Figure CDQF-FIG3: Controller Data Queue – Data Structure

Bytes	Description
03:00	**Head Pointer (HP):** This field indicates the slot of the head pointer for the specified CDQ.
07:04	**Tail Pointer Trigger (TPT):** If the ETPT bit is set to '1', then this field indicates the slot in the CDQ that when posted with an entry causes the controller to issue a Controller Data Queue Tail Pointer event. If the ETPT bit is cleared to '0', then this field shall be cleared to 0h and should be ignored by the host.
511:08	Reserved

…

5.27.2 Command Completion

…

Figure 371: Set Features – Command Specific Status Values

Value	Description
…	
15h	**I/O Command Set Combination Rejected:** This error indicates that the controller did not accept the request to select the requested I/O Command Set Combination.
37h	**Invalid Controller Data Queue:** This error indicates that the specified Controller Data Queue Identifier is invalid for the controller processing the command.

…

5.TRC Track Receive command

The Track Receive command is used to obtain the information being tracked by the controller that was enabled by a Track Send command (refer to section 5.TSC).

The Track Receive command uses the Data Pointer field, Command Dword 10 field, and Command Dword 12 field. The use of the Command Dword 11 field is specific to the management operation specified by the Select field. All other command specific fields are reserved.

The Select field defined in Figure TRC-FIG1 specifies the management operation to be performed. Refer to section 5.TRC.1 for a description of each management operation.

FigureTRC-FIG1: Track Receive – Command Dword 10

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:08</td><td>Reserved</td></tr>
<tr><td>07:00</td><td><b>Select (SEL):</b> This field specifies the type of management operation to perform.
<table>
<tr><th>Value</th><th>M/O[1]</th><th>Management Operation</th><th>Reference Section</th></tr>
<tr><td>0h</td><td>O</td><td>Tracked Memory Changes</td><td>5.TRC.1.1</td></tr>
<tr><td>1h to FFh</td><td></td><td>Reserved</td><td></td></tr>
<tr><td colspan="4">Notes:
1. O/M definition: O = Optional, M = Mandatory</td></tr>
</table>
</td></tr>
</table>

Figure TRC-FIG5: Track Receive – Command Dword 12

Bits	Description
31:00	**Number of Dwords (NUMDL):** This field specifies the number of dwords to return. If the host specifies a size larger than defined by the specified management operation of the command, then the controller returns the data specified by that management operation with undefined results for dwords beyond the end of the data specified by that management operation. This is a 0's based value.

5.TRC.1 Track Receive Management Operations

5.TRC.1.1 Tracked Memory Changes (Management Operation 0h)

The Tracked Memory Changes management operation of the Track Receive command is used to report host memory changes by the controller specified by the Controller Identifier (CNTLID) field (refer to Figure TRC-FIG) processing any and all commands (e.g., Admin commands, I/O command) as a result of tracking host memory modifications being started by a Tracked Memory Changes management operation of the Track Send command (refer to section 5.TSC.1.2).

The data returned is a Tracked Memory Change data structure defined by Figure TMCDS-FIG1. The identifier for the controller that is tracked is specified in the Controller Identifier field in the Command Dword 11 field as defined in Figure TRC-FIG.

Figure TRC-FIG: Tracked Memory Changes – Command Dword 11

Bits	Description
31:16	Reserved
15:00	**Controller Identifier (CNTLID):**. This field specifies the identifier of the controller for which the tracked memory changes, if any, are to be returned.

If the Track Receive command for this management operation completes successfully, then the controller shall remove the tracked memory changes that were reported in the returned data. The reported memory ranges identify the host memory that has changed due to the specified controller processing commands since:

- Track Memory Changes was enabled (refer to section 5.TSC.1.2) and no tracked memory changes have been reported by the controller; or
- the controller last reported tracking change while the memory tracking remained enabled.

If tracked memory changes occur for the same memory address at a high frequency, the controller reporting of the Tracking Memory Changed Descriptors for that address may be delayed by a vendor specific amount of time so that Tracking Memory Changed Descriptors for other addresses are able to be reported during that vendor specific amount of time.

If the Suspended bit is set to ‘1’ and the More To Report (MTR) bit is cleared to ‘0’ in the returned Track Memory Changed data structure, then all host memory changes have been reported to the host. If the controller specified by the CNTLID field remains suspended, then the controller has no more tracked host memory changes to report to the host until the controller specified by the CNTLID field is resumed.

If the Suspended bit is cleared to ‘0’ and the More To Report (MTR) bit is cleared to ‘0’ in the returned Track Memory Changed data structure, then no additional tracked host memory changes are available to report to the host in a subsequent commands at the time of processing this command.

- **Figure TMCDS-FIG1: Tracked Memory Change Data Structure**

Bytes	Description
Header	
00	**Version (VER):** This field indicates the version of this data structure. This field shall be cleared to 0h.

- **Figure TMCDS-FIG1: Tracked Memory Change Data Structure**

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>01</td><td><b>Attributes (ATTRB)</b>: This field identifies attributes associated with the tracked memory changes.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7:2</td><td>Reserved</td></tr>
<tr><td>1</td><td><b>Suspended (SUSP):</b> If this bit is set to ‘1’, then the controller was suspended during the processing of the command. If this bit is cleared to ‘0’, then the controller was not suspended during the processing of the command.</td></tr>
<tr><td>0</td><td><b>More To Report (MTR):</b> If this bit is set to ‘1’, then the controller was not able to report all of the tracked memory changes available to be reported in this returned data structure. The host should issue a subsequent Track Receive command to obtain any tracked memory changes not reported.
If this bit is cleared to ‘0’, the controller reported all of the tracked memory changes available to be reported, if any, at the time the Track Receive command was processed.
If the NTMCD field is cleared to 0h, then this bit shall be cleared to ‘0’.</td></tr>
</table></td></tr>
<tr><td>03:02</td><td><b>Controller Identifier (CNTLID)</b>: This field indicates the identifier for the controller whose memory modifications are being tracked and reported in the Tracked Memory Changed Descriptor list. (i.e., the same controller identifier as specified by the host in the Management Operation Specific field (refer to Figure TRC-FIG)).</td></tr>
<tr><td>07:04</td><td><b>Number of Tracked Memory Changed Descriptors (NTMCD):</b> This field indicates the number of Tracked Memory Changed Descriptors in this data structure. A value of 0h indicates that at the time of processing the Track Receive command, there was no tracked memory changes to report.</td></tr>
<tr><td>09:08</td><td><b>Reported Memory Range Granularity (RPMPG):</b> This field specifies the granularity unit size of the Length field in each Tracked Memory Changed Descriptor containing in this data structure. The value specified is a power of two (2^n) times 4 KiB. For example, a value of 8h indicates that the granularity of tracking is (2^8) * 4 KiB which is equal to 1 MiB.</td></tr>
<tr><td>15:10</td><td>Reserved</td></tr>
<tr><th colspan="2">Tracked Memory Changed Descriptor List</th></tr>
<tr><td>31:16</td><td><b>Tracked Memory Changed Descriptor 1:</b> This field contains the first Tracked Memory Change Descriptor as defined in Figure TMCDS-FIG2, if any.</td></tr>
<tr><td>47:32</td><td><b>Tracked Memory Changed Descriptor 2:</b> This field contains the second Tracked Memory Changed Descriptor as defined in Figure TMCDS-FIG2, if any.</td></tr>
<tr><td colspan="2">…</td></tr>
<tr><td>(NTMCD*16)+15:
(NTMCD-1)*16+16</td><td><b>Tracked Memory Changed Descriptor NTMCD:</b> This field contains the last Tracked Memory Changed Descriptor as defined in Figure TMCDS-FIG2, if any.</td></tr>
<tr><td colspan="2">…</td></tr>
</table>

Figure TMCDS-FIG2 defines a Tracked Memory Changed Descriptor.

Figure TMCDS-FIG2: Tracked Memory Changed Descriptor

Bytes	Description
07:00	**Start Address (SADDR):** This field indicates the 64-bit starting address of a memory range in which one or more bytes of data have been written. The address in this field shall be aligned to the granularity specified by the RPMPG field. For example, if the value in the RPMPG field is 8h, then the value in this field is required to be aligned to 1 MiB (i.e., bits 19:0 are cleared to 0h).
11:08	**Length (LEN):** This field specifies the length of this memory range in units of the tracking granularity as indicated in the RPMPG field. For example, if the value in the RPMPG field is 8h and the value of this field is 4h then the length of this length of this memory range is 1 MiB * 4h which is 4 MiB.
15:12	Reserved

5.TRC.2 Command Completion

Upon completion of the Track Receive command, the controller posts a completion queue entry to the Admin Completion Queue indicating the status for the command. Section 5.TRC.1 describes completion details each for management operation.

Track Receive command specific status values (i.e., SCT field set to 1h) are shown in Figure TSC-CC.

Figure TSC-CC: Track Receive – Command Specific Status Values

Value	Definition
1Fh	**Invalid Controller Identifier:** The controller for the specified Controller Identifier is already tracking host memory changes.

5.TSC Track Send command

The Track Send command is used to manage the tracking of information by a controller (e.g., what to track during the migration of a controller in an NVM subsystem).

The Track Send command uses the Command Dword 10 field. The use of the Data Pointer field and Command Dword 11 field is specific to the management operation specified by the Select field. All other command specific fields are reserved.

The Select field defined in Figure TSC-FIG1 specifies the management operation to be performed. Refer to section 5.TSC.1 for a description of each management operation.

Figure TSC-FIG1: Track Send – Command Dword 10

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:16</td><td><b>Management Operation Specific (MOS):</b> The definition of this field is specific to a management operation (refer to the Select field). If a management operation does not define the use of this field, then this field is reserved.</td></tr>
<tr><td>15:08</td><td>Reserved</td></tr>
<tr><td>07:00</td><td><b>Select (SEL):</b> This field specifies the type of management operation to perform.
<table>
<tr><th>Value</th><th>M/O[1]</th><th>Management Operation</th><th>Reference Section</th></tr>
<tr><td>0h</td><td>O</td><td>Log User Data Changes</td><td>5.TSC.1.1</td></tr>
<tr><td>1h</td><td>O</td><td>Track Memory Changes</td><td>5. TSC.1.2</td></tr>
<tr><td>2h to FFh</td><td></td><td colspan="2">Reserved</td></tr>
<tr><td colspan="4">Notes:
1. O/M definition: O = Optional, M = Mandatory</td></tr>
</table></td></tr>
</table>

5.TSC.1 Track Send Management Operations

5.TSC.1.1 Log User Data Changes (Management Operation 0h)

The Log User Data Changes management operation of the Track Send command is used to start or stop logging user data changes to namespaces attached to the controller associated with the User Data Migration Queue (refer to section 5.CDQC.1.1.1) specified in Controller Data Queue Identifier field in Command Dword 11 (refer to Figure LUDC-FIG2).

Refer to the applicable I/O command Set specification for additional requirements.

Figure LUDC-FIG1: Log User Data Changes – Management Operation Specific Field

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>15:04</td><td>Reserved</td></tr>
<tr><td>03:00</td><td><b>Logging Action (LACT):</b> This field specifies the logging action to be performed.
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>0h</td><td><b>Stop Logging:</b> The controller shall stop logging user data changes to namespaces attached to the controller associated with the User Data Migration Queue specified in the Controller Data Queue Identifier field.</td></tr>
<tr><td>1h</td><td><b>Start Logging:</b> The controller shall start logging user data changes to namespaces attached to the controller associated with the User Data Migration Queue (refer to the CDQID field in Figure LUDC-FIG2) into that User Data Migration Queue where those user data changes are caused by the controller associated with that User Data Migration Queue processing commands.</td></tr>
<tr><td>2h to Fh</td><td>Reserved</td></tr>
</table>
</td></tr>
</table>

The Log User Data Changes management operation uses Command Dword 11 as defined in Figure LUDC-FIG2.

Figure LUDC-FIG2: Log User Data Changes – Command Dword 11

Bits	Description
31:16	Reserved
15:00	**Controller Data Queue Identifier (CDQID):** This field specifies the Controller Data Queue Identifier of the User Data Migration Queue.

Refer to the applicable I/O Command Set specification for any requirements on the posting of entries into the User Data Migration Queue if this command is successful.

If the value in the Controller Data Queue Identifier field specifies a User Data Migration Queue that does not exist in the controller processing the command, then the controller shall abort the command with a status code of Invalid Controller Data Queue.

If the LACT field is set to 1h and any controller in the NVM subsystem is already logging user data for the controller associated with the Controller Data Queue specified by the CDQID field, then the controller shall abort the command with a status code of Invalid Controller Data Queue.

If the controller associated with the User Data Migration Queue specified by the CDQID field is suspended as a result of a Migration Send command (refer to section 5.MSC.1.1), then the controller shall abort the command with a status code of Controller Suspended.

5.TSC.1.2 Track Memory Changes (Management Operation 1h)

The Track Memory Changes management operation of the Track Send command requests the controller processing that Track Send command to:

- start tracking host memory changes described by the specified host memory ranges that are caused by the processing commands in the controller specified by Controller Identifier field (refer to Figure TMC-FIG1); or
- stop tracking the host memory changes that are caused by the processing commands by the controller specified by Controller Identifier field.

If the TACT bit is set to ‘1’ (refer to Figure TMC-FIG), then:

- The data buffer contains a Track Memory Changes data structure (refer to Figure TMC-FIG2) that specifies one or more host memory ranges associated with the controller specified by the Controller Identifier field.
- If the value in the Requested Memory Range Tracking Granularity (RMRTG) field is greater than the value in the Maximum Memory Range Tracking Granularity (MAXMRTG) field in the Identify

Controller data structure (refer to Figure 276), then the controller shall abort the command with a status code of Invalid Field in Command.

- If the value in the Requested Memory Range Tracking Granularity (RMRTG) field is less than the value in the Minimum Memory Range Granularity (MINMRG) field in the Identify Controller data structure (refer to Figure 276), then the controller shall abort the command with a status code of Invalid Field in Command.
- If the sum of:
    - the value in the Requested Number of Memory Range Tracking Descriptors (RNMRTD) field; and
    - the number of Memory Range Tracking Descriptors currently being tracked by the controller processing the command for other controllers in the NVM subsystem,

  is greater than the value in the Controller Maximum Memory Range Tracking Descriptors (CMMRTD) field in the Identify Controller data structure (refer to Figure 276), then the controller processing the command shall abort the command with a status code of Invalid Field in Command.
- If the sum of the value in the Requested Number of Memory Range Tracking Descriptors (RNMRTD) field plus the number of Memory Range Tracking Descriptors currently being tracked by all controllers in the NVM subsystem is greater than the value in the NVM subsystem Maximum Memory Range Tracking Descriptors (NMMRTD) field in the Identify Controller data structure (refer to Figure 276), then the controller shall abort the command with a status code of Invalid Field in Command.
- If any controller in the NVM subsystem is already tracking host memory changes for the controller specified by the CNTLID field, then the controller processing the command shall abort the command with a status code of Invalid Controller Identifier.
- If:
    - the Memory Range Tracking Length Limit (MRTLL) bit in the Identify Controller data structure is set to ‘1’; and
    - the length as specified by the Requested Memory Range Tracking Granularity field and any Length field in a Track Memory Changes data structure is not a value that is a power of 2,

  then the controller shall abort the command with a status code of Invalid Field in Command.

If the TACT bit is cleared to ‘0’, then the data buffer shall be ignored by the controller.

Figure TMC-FIG: Track Memory Changes – Management Operation Specific Field

Bits	Description
15:01	Reserved
00	**Tracking Action (TACT):** If this bit is set to ‘1’, then the data buffer specifies the host memory ranges associated with the controller specified by the Controller Identifier field. Any writes to those host memory ranges by the controller specified by the Controller Identifier field shall be tracked by the controller processing the command (i.e., tracking starts). If this bit is cleared to ‘0’, then controller processing the command shall stop tracking host memory changes by the controller specified by the Controller Identifier field.

Figure TMC-FIG1: Track Memory Changes – Command Dword 11

Bits	Description
31:16	Reserved
15:00	**Controller Identifier (CNTLID):**. This field specifies the identifier of the controller for which changes to host memory is being tracked. If the value of this field is the controller identifier of the controller processing the command, then the controller processing the command shall abort the command with a status code of Invalid Controller Identifier.

If the controller specified by the CNTLID field is suspended (refer to section 5.MSC.1.1), then the controller shall abort the command with a status code of Controller Suspended.

Figure TMC-FIG2: Track Memory Changes Data Structure

Bytes	Description
Header	
00	**Version (VER):** This field specifies the version of this data structure. If this field is not cleared to 0h, then the controller shall abort this command with a status code of Invalid Field in Command.
02:01	Reserved
03	**Requested Memory Range Tracking Granularity (RMRTG):** This field specifies the granularity for the Length field in each Memory Range Tracking Descriptor contained in this data structure. The value specified is a power of two (2^n) times 4 KiB. For example, a value of 8h indicates that the granularity of tracking is (2^8) * 4 KiB which in 1 MiB.
07:04	**Number of Memory Range Tracking Descriptors (RNMRTD):** This field specifies the number of Memory Range Tracking Descriptors in the Memory Range Tracking Descriptor list. This field is a 1's based number.
Memory Range Tracking Descriptor List	
19:08	**Memory Range Tracking Descriptor 0:** This field contains the first Memory Range Tracking Descriptor as defined in Figure TMC-FIG3.
31:20	**Memory Range Tracking Descriptor 1:** This field contains the second Memory Range Tracking Descriptor as defined in Figure TMC-FIG3, if any.
...	
(RNMRTD*12)+7: (RNMRTD-1)*12+8	**Memory Range Tracking Descriptor RNMRTD-1:** This field contains the last Memory Range Tracking Descriptor as defined in Figure TMC-FIG3, if any.

Figure TMC-FIG3 defines a Memory Range Tracking Descriptor. The address range specified by a Memory Range Tracking Descriptor describes addresses specified in an SGL or PRP for a command submitted by a host to the controller specified in the CNTLID field (refer to Figure TMC-FIG1).

Figure TMC-FIG3: Memory Range Tracking Descriptor

Bytes	Description
07:00	**Address (SADDR):** This field specifies the starting 64-bit address of the memory range where the address is host memory (refer to section 1.5.28) associated with the controller specified by the CNTLID field (refer to Figure TMC-FIG1). The address is aligned to the granularity specified by the RMRTG field. For example, if the value in the RMRTG field is 8h, then the value in this field is required to the be aligned to 1 MiB (i.e., bits 19:0 are cleared to 0h).
11:08	**Length (LEN):** This field specifies the length of this host memory range in units of the tracking granularity as defined by RMRTG field. For example, if the value in the RMRTG field is 8h and the value of this field is 4h, then the length of this memory range is 1 MiB * 4h which is 4 MiB.

If the controller is unable to track the requested number of Memory Range Tracking Descriptors (i.e. specified in the RNMRTD field) or unable to track the memory changes at the specified granularity (i.e., RMRTG field), then:

- the controller shall abort the command with a status code of Not Enough Resources; and
- Dword 0 and Dword 1 of the completion queue entry shall be returned as defined in Figure TMC-FIGF4 and Figure TMC-FIG5.

If any Address field is not aligned to the granularity specified by the RMRTG field, then the controller shall abort the command with a status code of Invalid Field in Command.

If the address range specified by the Address field and the Length field specifies an address that is not host memory (refer to section 1.5.28) associated with the controller specified by the CNTLID field, then the controller processing the command shall abort the command with a status code of Invalid Field in Command. If:

- the command is successful; and
- the configuration within the controller associated with the controller specified by the CNTLID field changes resulting in that address range is no longer specifying host memory associated with the controller specified by the CNTLID field (e.g., the address range overlaps with PMR or CMB), then the behavior for tracking host memory changes for the controller specified by the CNTLID field is undefined.

A host should not specify MSI or MSI-X registers in the address range. If the address range specified by the Address field and the Length field contains addresses for MSI or MSI-X registers on the controller specified by the CNTLID field, then the controller may or may not report modification to those MSI or MSI-X register addresses.

Figure TMC-FIG4: Track Memory Changes – Completion Queue Entry Dword 0

Bits	Description
31:16	Reserved
15:00	**Memory Range Tracking Granularity (MRTG):** This field indicates the lowest granularity that the controller is able track the memory changes with the specified Memory Range Tracking Descriptor list (refer to Figure TMC-FIG2). This units for this field is defined by the RMRTG field.

Figure TMC-FIG5: Track Memory Changes – Completion Queue Entry Dword 1

Bits	Description
31:00	**Number of Memory Range Tracking Descriptors (NMRTD):** This field indicates the number of Memory Range Tracking Descriptors for which the controller is able to track the memory changes with the requested granularity specified in the RMRTG field (refer to Figure TMC-FIG2).

5.TSC.2 Command Completion

Upon completion of the Track Send command, the controller posts a completion queue entry to the Admin Completion Queue indicating the status for the command. Section 5.TSC.1 describes completion details each for management operation.

The use of Dword 0 and Dword 1 of the completion queue entry is specific to the management operation specified in the Select field (refer to Figure TSC-FIG1). Refer to each management operation description in section 5.TSC.1 for details.

Track Send command specific status values (i.e., SCT field set to 1h) are shown in Figure MSCC-FIG1.

Figure MSCC-FIG1: Track Send – Command Specific Status Values

Value	Definition
1Fh	**Invalid Controller Identifier:** The controller for the specified Controller Identifier is already tracking host memory changes.
37h	**Invalid Controller Data Queue:** This error indicates: • that the specified Controller Data Queue Identifier is invalid for the controller processing the command; or • the controller associated with the Controller Data Queue specified by the Controller Data Queue Identifier is already logging user data changes.
38h	**Not Enough Resources:** This error indicates that there are not enough resources in the controller to process the command successfully.
39h	**Controller Suspended:** The requested operation is not allowed if the controller specified in the Controller Identifier field is suspended by a Migration Send command (refer to section 5.MSC.1.1).

…

Modify a portion of section 8 as shown below:

…

8 Extended Capabilities

…

8.CDQ Controller Data Queue

A Controller Data Queue (CDQ) is used to post information from the controller to the host that is specific to the type of queue being created (refer to Figure CDQC-FIG5) by the Controller Data Queue command that specifies the Create Controller Data Queue management operation (refer to section 5.CDQC.1.1). A CDQ is a circular buffer with a fixed slot size with entries posted by a controller.

A CDQ is deleted:

- by a Controller Data Queue command that specifies the Delete Controller Data Queue management operation (refer to section 5.CDQC.1.2);
- on a Controller Level Reset on the controller that processed the Controller Data Queue command that created the CDQ; or
- by a Migration Send command that specifies the Suspend management operation and the Delete User Data Migration Queue (DUDMQ) bit is set to ‘1’ (refer to section 5.MSC.1.1).

On the successful completion of a Controller Data Queue command, a Controller Data Queue Identifier (CDQID) is returned in the completion queue entry (refer to Figure CDQC-FIG9). A CDQID is used to identify the created Controller Data Queue on the controller that processed the command. The returned value for that CDQID is any value except the CDQID values for any Controller Data Queue that exists on the controller that processed the command (i.e., a controller is permitted to return the value of a CDQID value of a Controller Data Queue that was previously deleted). The scope of a CDQID is per controller.

The CDQ Head pointer is updated by the host by issuing a Set Features command specifying the Controller Data Queue feature and the Controller Data Queue Identifier (refer to section 5.27.1.CDQF) after processing an entry indicating the last free CDQ slot. A Controller Data Queue Phase Tag (CDQP) bit is defined in the CDQ entry (refer to the applicable I/O command set specification) to indicate whether an entry has been newly posted without the host relying on the Controller Data Queue Tail Pointer event (refer to Figure AER-OS). The Controller Data Queue Phase Tag bit enables the host to determine whether entries are new or not.

8.CDQ.1 Controller Data Queue Usage

The controller uses the current Tail entry pointer to identify the next open CDQ slot. The controller increments the Tail entry pointer after placing the new entry to the next open CDQ slot. If the Tail entry pointer increment exceeds the CDQ size, then the Tail entry pointer shall roll to zero. The controller may continue to place entries in free CDQ slots as long as the Full queue condition is not met (refer to section 3.3.1.5). The controller shall take CDQ wrap conditions into account.

The host uses the current Head entry pointer to identify the slot containing the next entry to be consumed. The host increments the Head entry pointer after consuming the next entry from the CDQ. If the Head entry pointer increment exceeds the CDQ size, the Head entry pointer shall roll to zero. The host may continue to consume entries from the CDQ as long as the Empty queue condition is not met (refer to section 3.3.1.3).

Note: The host shall take CDQ wrap conditions into account.

A host issues a Set Features command specifying the Controller Data Queue feature to communicate a new value of the Head entry pointer to the controller. If the host specifies an invalid value to the Head entry pointer, then that Set Features command is aborted. This condition may be caused by a host attempting to remove an entry from an empty CDQ.

A host checks a posted entries Controller Data Queue Phase Tag (CDQP) bits in memory to determine whether new CDQ entries have been posted (refer to section 8.CDQ.2). The CDQ Tail pointer is only used internally by the controller and is not visible to the host.

An entry is posted to the CDQ when the controller write of that entry to the next free CDQ slot inverts the Controller Data Queue Phase Tag (CDQP) bit from its previous value in memory (refer to section 8.CDQ.2). The controller generates a Controller Data Queue Tail Pointer event (refer to Figure AER-OS) to the host to indicate that the CDQ slot specified by the host in the Controller Data Queue feature (refer to section 5.27.1.CDQF) has been posted if in the Controller Data Queue feature associated with the CDQ:

- the Tail pointer value matches the value in the Tail Pointer Trigger (TPT) field; and
- the Enable Tail Pointer Trigger (ETPT) bit is set to ‘1’.

A CDQ entry has been consumed by the host when the host submits a Set Features command specifying the Controller Data Queue feature with a new value that indicates that the CDQ Head Pointer has moved past the slot in which that CDQ entry was placed. A Set Features command specifying the Controller Data Queue feature may indicate that one or more CDQ entries have been consumed.

Altering a CDQ entry after that entry has been posted but before that entry has been consumed results in undefined behavior.

Refer to the specific type of CDQ to determine the behavior of a CDQ that has become full.

Refer to section 3.3.1.4 for the definition of an empty CDQ. Refer to section 3.3.1.5 for the definition of a full CDQ.

If a CQD entry is constructed via multiple writes, the Controller Data Queue Phase Tag (CDQP) bit shall be updated in the last write of that CDQ entry.

Refer to the applicable I/O Command Set specifications for any specific requirements on the use of CDQs.

8.CDQ.2 Controller Data Queue Phase Tag

The Controller Data Queue Phase Tag (CDQP) bit indicates whether a CDQ entry is new. The CDQP bit for each CDQ entry in the CDQ shall be initialized to ‘0’ by the host before creating the CDQ by submitting a Controller Data Queue command (refer to section 5.MSC) specifying the Create Queue management operation (refer to section 5.MSC.1.1).

When the controller posts a new CDQ entry to the CDQ, the controller shall invert the CDQP bit in that CDQ entry (i.e., the inverting of the CDQP bit enables the host to detect the new CDQ entry).

When a CDQ entry is posted to a CDQ slot in the CDQ for the first time after the CDQ is created, the CDQP bit for that completion queue entry is set to ‘1’.

This continues for each CDQ entry that is posted until the controller posts a CDQ entry to the highest numbered CDQ slot and wraps to CDQ slot number 0 as described in section 8.CDQ.1. When that CDQ wrap condition occurs, the CDQP bit is then cleared to ‘0’ in each CDQ entry that is posted. This continues until another CDQ wrap condition occurs. Each time a CDQ wrap condition occurs, the value of the CDQP bit is inverted (i.e., changes from ‘1’ to ‘0’ or changes from ‘0’ to ‘1’).

…

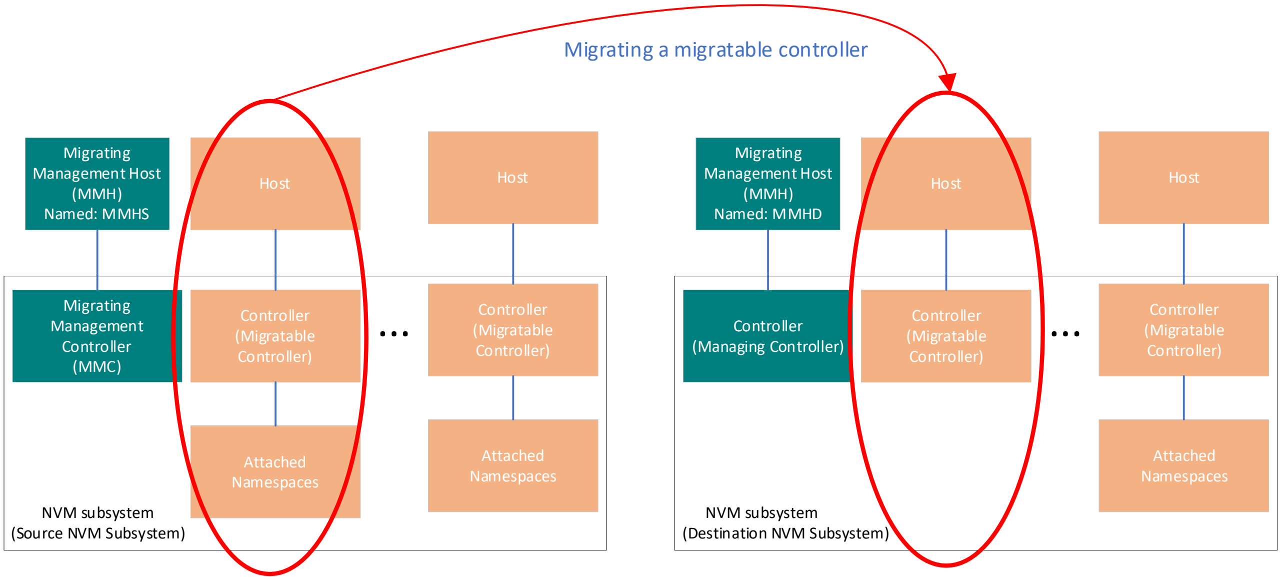
8.LM Host Managed Live Migration

The migration of a controller, by a host that is using the Host Managed Live Migration capability, involves one or more NVM subsystems, multiple hosts, and multiple controllers. In this section, to differentiate between the multiple hosts and controllers, each is given a unique name as illustrated in Figure LM-FIG1. In an NVM subsystem that supports Host Managed Live Migration, each controller that sets the HMLMS bit to ‘1’ is referred to as a Migration Management Controller (MMC) and each host associated with an MMC is referred to as a Migration Management Host (MMH). In that same NVM subsystem, all other controllers (i.e., those that clear the HMLMS bit to ‘0’) are referred to as Migratable Controllers.

The Host Managed Live Migration capability allows an MMH to use an MMC to migrate a Migratable Controller from a Source NVM Subsystem to a Migratable Controller in a Destination NVM Subsystem. During that migration, a host is actively submitting commands to a Migratable Controller in the Source NVM Subsystem and that controller is processing those commands.

The MMH that is managing the migration of a Migratable Controller from the Source NVM Subsystem is responsible for ensuring that the Migratable Controller in the Destination NVM Subsystem is compatible with controller state of the Migratable Controller from the Source NVM Subsystem. It is also the responsibility of the MMH to transfer the migrating data between the NVM subsystems.

Figure LM-FIG1: Host Managed Live Migration Host and Controller Naming


Migrating a migratable controller
Migrating Management Host (MMH) Named: MMHS
Host
Host
Migrating Management Controller (MMC)
Controller (Migratable Controller)
...
Controller (Migratable Controller)
Attached Namespaces
Attached Namespaces
NVM subsystem (Source NVM Subsystem)
Migrating Management Host (MMH) Named: MMHD
Host
Host
Controller (Managing Controller)
Controller (Migratable Controller)
...
Controller (Migratable Controller)
Attached Namespaces
NVM subsystem (Destination NVM Subsystem)

An MMC is an I/O controller or an Administrative controller that supports the Host Managed Live Migration capability and provides the ability for the MMH to use privileged actions (refer to section 3.10) to:

- suspend the processing of commands on the Migratable Controller in the Source NVM Subsystem being migrated (refer to the Migration Send command in section 5.MSC.1.1);
- obtain the state of the Migratable Controller in the Source NVM Subsystem being migrated (refer to the Migration Receive command in section 5.MRC.1.1);
- set that controller state in the Migratable Controller in the Destination NVM Subsystem (refer to the Migration Send command in section 5.MSC.1.3); and
- resume the operation on the Migratable Controller in the Destination NVM Subsystem (refer to the Migration Send command in section 5.MSC.1.2).

A controller indicates support for the Host Managed Live Migration capability by setting the Host Managed Live Migration Support (HMLMS) bit to ‘1’ in the Optional Admin Command Support (OACS) field in the Identify Controller data structure (refer to Figure 276).

An MMC is not permitted to be migrated and:

- shall support:
  - the Migration Send command with the Management Operations,
    - Suspend (i.e., 0h);
    - Resume (i.e., 1h); and
    - Set Controller State (i.e., 2h);

- the Migration Receive command with the Management Operations,
  - Controller State (i.e., 1h);

  and
- the CNS value of 21h in the Identify command (i.e., Supported Controller State Formats data structure (refer to section 5.17.2.SCSF)).

If the NVM subsystem contains one or more Migration Management Controllers, then:

- each controller (i.e., MMC and Migratable Controller) in that NVM subsystem does not support the Host Memory Buffer (refer to the Host Memory Buffer Preferred Size (HMPRE) field in the Identify Controller data structure in Figure 276); and
- each Migratable Controller in that NVM subsystem is allowed to be migrated and shall not support:
  1. the Migration Send command (refer to section 5.MSC);
  2. the Migration Receive command (refer to section 5.MRC);
  3. the CNS value of 21h in the Identify command (i.e., returning Supported Controller State Formats data structure (refer to section 5.17.2.SCSF)); and
  4. the Controller Data Queue command with the Create Queue management operation specifying the User Data Migration Queue (refer to section 5.CQDC.1.1).

A Controller Level Reset on an MMC shall cause that MMC to:

- delete all User Data Migration Queues on that MMC; and
- stop tracking all host memory changes on that MMC;

A Controller Level Reset on a Migratable Controller shall cause that Migratable Controller to:

- remove a suspended state if that Migratable Controller is suspended by a Migration Send command specifying the Suspend management operation (refer to section 5.MSC.1.1); and
- retain any persistent controller state that has been committed to that Migratable Controller from a Migration Send command specifying the Set Controller State management operation (refer to section 5.MSC.1.3).

During the migration of a Migratable Controller in the Source NVM Subsystem while that Migratable Controller is processing commands, it may be necessary for the MMHS to obtain the user data modifications to namespaces attached to that Migratable Controller and modifications made by that Migratable Controller to host memory in the host. The Track Send command (refer to section 5.TSC) and the Track Receive command (refer to section 5.TRC) allows the host to use privileged actions (refer to section 3.10) to:

- determine user data in namespaces attached to a controller that has changed; and
- determine host memory that has changed due to commands processed by a controller.

A controller indicates support for tracking changes to user data in namespaces attached to a controller by setting the Track User Data Changes Support (TUDCS) bit to ‘1’ in the Tracking Attributes (TRATTR) field in the Identify Controller data structure (refer to Figure 276).

A controller indicates support for tracking changes to host memory (refer to section 1.5.28) due to the processing of commands by a controller by setting the Track Host Memory Changes Support (THMCS) bit to ‘1’ in the TRATTR field.

8.LM.1 Process for Migrating a Controller

This section provides an example sequence of steps to allow the MMHS to migrate a Migratable Controller in the Source NVM Subsystem and the attached namespaces to a Migratable Controller in the Destination NVM Subsystem (refer to Figure LM-FIG1). Additionally, the steps include obtaining the host memory modifications made to the host due to the Migratable Controller in the Source NVM Subsystem processing commands during the migration.

In this example the MMC in the Source NVM Subsystem has:

- the Track User Data Changes Support (TUDCS) bit set to '1' in the Tracking Attributes (TRATTR) field in the Identify Controller data structure (refer to Figure 276); and
- the Track Host Memory Changes Support (THMCS) bit set to '1' in the Tracking Attributes (TRATTR) field in the Identify Controller data structure.

If attached namespaces are not required to be migrated by the MMHS, then those steps may be ignored. If host memory modifications are not required to be obtained by the MMHS, then those steps may be ignored.

1. The MMHS copies (i.e., migrates) the user data in namespaces attached to the Migratable Controller in the Source NVM Subsystem while the host is submitting commands and the Migratable Controller is processing those submitted commands.

    a. The MMHS creates a User Data Migration Queue in the MMC in the Source NVM Subsystem by submitting a Controller Data Queue command to that MMC (refer to section 5.CDQC) specifying:

    - the Select field set to the Create Queue management operation (i.e., 1h);
    - the Queue Type field set to User Data Migration Queue; and
    - the Controller Identifier field (refer to Figure CDQC-FIG6) set to the controller identifier for the Migratable Controller in the Source NVM Subsystem.

    b. If the Controller Data Queue command is successful, the completion queue entry contains the Controller Data Queue Identifier for the created User Data Migration Queue. The MMHS causes the MMC in the Source NVM Subsystem to post user data modifications to namespaces attached to the Migratable Controller by submitting a Track Send command to that MMC specifying:

    - the Select field set to Log User Data Changes management operation (i.e., 0h);
    - the Logging Action bit set to '1' (i.e., start logging); and
    - the Controller Data Queue Identifier field set to the Controller Data Queue Identifier from the completion queue entry for the Controller Data Queue command.

    It is the responsibility of the MMHS to make sure that the User Data Migration Queue is sized properly and there may be more restrictions on that MMHS as specified by the appropriate I/O command set (e.g., the NVM Command Set requires the MMHS to not allow the User Data Migration Queue to become full). Refer to section 8.CDQ for a description of the User Data Migration Queue (i.e., a Controller Data Queue).

    c. If the Track Send command is successful, then the MMHS starts copying the namespaces attached to the Migratable Controller in the Source NVM Subsystem to the Destination NVM Subsystem. Refer to the applicable I/O Command Set specifications for capabilities that may reduce the amount of user date that is required to copied (e.g., the Get LBA Status command in the NVM Command Set Specification).

2. The MMHS causes the MMC in the Source NVM Subsystem to start tracking the modifications to host memory in the host due to the Migratable Controller processing submitted commands.

    a. The MMHS submits a Track Send command to the MMC in the Source NVM Subsystem (refer to section 5.CDQC) specifying:

    - the Select field set to the Track Memory Changes management operation (i.e., 0h);
    - the Tracking Action (TACT) bit set to '1';
    - the Controller Identifier field specifying the identifier for the Migratable Controller in the Source NVM Subsystem being migrated; and
    - a Track Memory Changes data structure specifying the host memory to be tracked as described by the set of host memory ranges.

    b. If the Track Send command is successful, then the MMHS may query the host memory modifications that have resulted from the Migratable Controller processing commands by submitting a Track Receive command to the MMC in the Source NVM Subsystem specifying:

        - the Select field set to the Track Memory Changes management operation (i.e., 0h); and
        - the Controller Identifier field specifying the identifier for the Migratable Controller in the Source NVM Subsystem being migrated.

3. Once the MMHS has copied the namespaces attached to the Migratable Controller in the Source NVM Subsystem, that MMHS determines a time to suspend that Migratable Controller. During this period that MMHS is migrating the user data in the namespaces attached to that Migratable Controller that are identified in the posted entries in the User Data Migration Queue. That MMHS may handle the modified host memory in the host as reported by the Track Receive command.
4. The MMHS suspends the Migratable Controller in the Source NVM Subsystem by submitting a Migration Send command to the MMC in the Source NVM Subsystem specifying:

    - the Select field set to the Suspend management operation (i.e., 0h);
    - the Suspend Type field set to Suspend;
    - Delete User Data Migration Queue bit set to '1', if the MMHS desires that the MMC delete the User Data Migration Queue being used to log user data changes of the Migratable Controller being migrated; and
    - the Controller Identifier field (refer to Figure CDQC-FIG6) set to the controller identifier for the Migratable Controller to be suspended (i.e., the Migratable Controller being migrated).

    Prior to completing that Migration Send command:

    - the specified Migratable Controller being suspended:

        - stops fetching commands (i.e., from the Admin queue and I/O queues); and
        - completes all previously fetched commands;

      and

    - the MMC processing that Migration Send command:

        - if user data is being logged into a User Data Migration Queue that is associated with the Migratable Controller being migrated, an entry is placed into the User Data Migration Queue that indicates the user data logging has suspended, and then deletes the User Data Migration Queue, if the Delete User Data Migration Queue bit is set to '1'.

5. Since the Migratable Controller in the Source NVM Subsystem is suspended, the MMHS migrates the user data in the namespaces attached to the Migratable Controller that are identified in the posted entries in the User Data Migration Queue. Refer to the appropriate I/O command set specification to determine how the entries identify when entries have completed being posted. That MMHS may handle the modified host memory in the host as reported by the Track Receive command.
6. The MMHS is able to obtain the controller state of the Migratable Controller in the Source NVM Subsystem by submitting one or more Migration Receive commands to the MMC in the Source NVM Subsystem specifying:

    a. the Select field set to the Get Controller State management operation (i.e., 0h);
    b. the Sequence Indicator field set to the proper sequence value;
    c. the Controller Identifier field set to the controller identifier for the Migratable Controller in the Source NVM Subsystem;
    d. the Controller State Version Index field set to an index into the NVMe Controller State Version list in the Supported Controller State Formats data structure for MMC in the Source

NVM Subsystem (refer to Figure SCSF-FIG1) if the NVMe defined controller state is required; and

e. the Controller State UUID Index field set to an index in the Vendor Specific Controller State UUID Supported list in the Supported Controller State Formats Data Structure for MMC in the Source NVM Subsystem (refer to Figure SCSF-FIG1), if vendor specific controller state is required.

7. After the MMHS has transferred the obtained controller state information to the MMHD, the MMHD sets that controller state into a Migratable Controller by submitting a sequence of one or more Migration Send commands to the MMC in the Destination NVM Subsystem specifying:

   - the Select field set to Set Controller State management operation (i.e., 3h);
   - the Sequence Indicator field set to the proper sequence value;
   - the Controller Identifier field set to the controller identifier for the Migratable Controller in the Destination NVM Subsystem;
   - the Controller State Version Index field set to an index into the NVMe Controller State Version list in the Supported Controller State Formats data structure for MMC in the Destination NVM Subsystem (refer to Figure SCSF-FIG1) if the value of the NVMe Controller State Size (NVMECSS) field is non-zero (refer to Figure SCS-FIG5); and
   - the Controller State UUID Index field set to an index in the Vendor Specific Controller State UUID Supported list in the Supported Controller State Formats Data Structure for MMC in the Source NVM Subsystem (refer to Figure SCSF-FIG1), if the value of the Vendor Specific Size (VSS) field is non-zero (refer to Figure SCS-FIG5).

8. The MMHD resumes operation on the Migratable Controller in the Destination NVM Subsystem (i.e., the migrated controller) by submitting a Migration Send command to the MMC in the Destination NVM Subsystem specifying:

   - the Select field set to Resume management operation (i.e., 2h);
   - the Controller Identifier field (refer to Figure CDQC-FIG6) set to the controller identifier for the Migratable Controller to be resumed.

Description of Specification Changes for the NVM Express NVM Command Set Specification 1.0d

Modify a portion of section 4 as shown below:

…

4 Admin Commands for the NVM Command Set

4.1 Admin Command behavior for the NVM Command Set

…

4.1.5 Identify Command

…

4.1.5.4 I/O Command Set Specific Identify Controller Data Structure (CNS 06h, CSI 00h)

…

Figure 102: I/O Command Set Specific Identity Controller Data Structure for the NVM Command Set

<table>
<tr><th>Bytes</th><th>O/M [1]</th><th>Description</th></tr>
<tr><td colspan="3">…</td></tr>
<tr><td>15:08</td><td>O</td><td><b>Dataset Management Size Limit (DMSL):</b> …</td></tr>
<tr><td colspan="3"><b>…</b></td></tr>
<tr><td>24</td><td>O</td><td><b>LBA Migration Queue Format (LBAMQF):</b> This field indicates the format supported by this controller for User Data Migration Queue entries. Refer to section 5.MQ.
<table>
<tr><th>Value</th><th>Definition</th><th>Reference</th></tr>
<tr><td>0h</td><td>LBA Migration Queue Entry Type 0.</td><td>Figure NVMCS-FIG1</td></tr>
<tr><td>1h to BFh</td><td>Reserved</td><td></td></tr>
<tr><td>C0h to FFh</td><td>Vendor Specific</td><td></td></tr>
</table></td></tr>
<tr><td>11:26</td><td></td><td>Reserved</td></tr>
<tr><td colspan="3">…</td></tr>
</table>

…

4.1.NEW Track Send command

Upon posting the successful completion to a Track Send command (refer the NVM Express Base Specification) that specifies:

- the Log User Data Changes management operation;
- the Logging Action (LACT) bit set to ‘1’; and
- a Controller Data Queue Identifier for a created LBA Migration Queue,

then the controller shall post entries into that LBA Migration Queue that identifies the logical block modifications to the attached namespaces on the controller associated with the LBA Migration Queue (refer to section 5.MQ). Posting of these entries into the LBA Migration Queue continues until the logging is stopped as specified by the NVM Express Base specification.

If the Reporting Allocated LBA Supported (RALBAS) bit is set to ‘1’ (refer to Figure 102) (Editor: This is from TP4165), then after that Track Send command completes successfully, any logical block modifications to the attached namespaces on the controller associated with the LBA Migration Queue (refer to section

5.MQ) that occur during the processing of that Track Send command are reported by a Get LBA Status command with the Action Type field set to 02h (i.e., Return Allocated LBAs) (Editor: this is from TP4165). Those logical block modifications to the attached namespaces on the controller associated with the LBA Migration Queue (refer to section 5.MQ) that occur during the processing of that Track Send command may also be logged into the LBA Migration Queue.

If the RALBAS bit is cleared to '0', then after that Track Send command completes successfully, any logical block modifications to the attached namespaces on the controller associated with the LBA Migration Queue (refer to section 5.MQ) that occur during the processing of that Track Send command may be logged into the LBA Migration Queue.

The posting of an entry into the LBA Migration Queue as a result of a Track Send command to indicate:

- the start of logging user data changes; or
- the stop of logging user data changes,

is not required to be posted prior to posting the completion of that Track Send command.

If:

- the specified LBA Migration Queue in Track Send command with the Log User Data Changes management operation and the Logging Action (LACT) bit set to '1'; and
- that specified LBA Migration Queue is currently full,

then the controller may abort the command with a status code of Controller Data Queue Full.

If the Track Send command is successful and there is a pending Controller Data Queue Full Error event for the LBA Migration Queue specified by the CDQID field, then the controller shall discard that pending event with no effects to the state of the LBA Migration Queue.

…

Modify a portion of section 5 as shown below:

5 Extended Capabilities

…

5.MQ LBA Migration Queue

As defined by the NVM Express Base Specification, a User Data Migration Queue allows a controller to post entries that identify user data modifications due to the processing of commands by the controller associated to the User Data Migration Queue. For the NVM Command Set:

- user data is logical blocks and posted entries indicate:
  - data in the logical blocks has changed (e.g., modified by a Write command);
  - data in the logical blocks may have changed (e.g., a Write command was processed that targeted the logical blocks but there was no change to the data in those logical blocks);
  - are deallocated (i.e., the logical block transitioned from being allocated to deallocated); or
  - may have been deallocated (i.e., the command requested to deallocate logical blocks that were already deallocated);

  and
- the User Data Migration Queue is referred to as the LBA Migration Queue.

If logging has been started in an LBA Migration Queue due to a Track Send command (refer to the NVM Express Base Specification), the first entry posted indicates that the logging has started. Subsequent entries are able to indicate if an entry is:

- the last entry due to the controller associated with the LBA Migration Queue being suspended as a result of a Migration Send command (refer to the NVM Express Base Specification);
- the last entry due to logging being stopped as a result of the controller processing a command that is defined to stop logging (e.g., the Track Send command (refer to the NVM Express Base Specification));
- the controller detecting that the LBA Migration Queue has become full; or
- the first entry due to the controller associated with the LBA Migration Queue being resumed while suspended as a result of a Migration Send command.

5.MQ.1 LBA Migration Queue Entries

The LBA Migration Queue Format (LBAMQF) field in the I/O Command Set specific Identify Controller data structure (refer to Figure 102) defines the supported format for entries in an LBA Migration Queue. Figure NVMCS-FIG1 defines the supported format.

A controller may aggregate the results of multiple commands processed by a controller into a single entry. For example:

- If the controller processes three sequential Write commands, then the controller may post a single entry that incorporates all of the logical blocks written.
- If the controller processes the sequential commands that modify the same LBA range, then the controller may post a single entry for the LBA range with the result from the last command of that set of sequential commands.

The controller shall only post an entry into an LBA Migration queue if the processing for the command reported by the entry has taken effect, which may be before the completion for that command is posted. For example, if a Write command is processed by a controller that results in the posting of an entry in the LBA Migration queue that has:

- LBAINR bit cleared to ‘0’;
- the SLBA field set to the SLBA field of that Write command; and
- the NLB field set to the NLB field of that Write command,

then the host may issue a Read command to obtain the logical blocks written even if the completion for that Write command is not posted.

If:

- the controller processes a command that is a request to modify a logical block; and
- that modification results:
  - in the logical block remaining in the same condition (e.g., host deallocates a logical block that is already deallocated); or
  - the logical block has the exact same data (e.g., the controller processes a Write Zeroes command for a logical block that already is written with zero data),

then the controller may or may not report that logical block in an entry in the LBA Migration Queue.

Figure NVMCS-FIG1: LBA Migration Queue Entry Type 0

Bytes	Description
03:00	**Namespace Identifier (NSID):** This field indicates the namespace identifier associated with the logical blocks reported in this entry. If the LBACIR field is set to 10b, then this field shall be cleared to 0h and should be ignored by the host.
07:04	**Number of Logical Blocks (NLB):** This field indicates the number of logical blocks reported by this entry. This is a 0’s based value. If the LBACIR field is set to 10b, then this field shall be cleared to 0h and should be ignored by the host.

Figure NVMCS-FIG1: LBA Migration Queue Entry Type 0

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>15:08</td><td><b>Starting LBA (SLBA):</b> This field indicates the 64-bit address of the first logical block of data reported by this entry.
If the LBACIR field is non -zero, then this field shall be cleared to 0h and should be ignored by the host.</td></tr>
<tr><td>30:16</td><td>Reserved</td></tr>
<tr><td>31</td><td><b>LBA Migration Queue Attributes (LBAMQA):</b> This field indicates attributes associates with the LBE Migration Queue entry.
<table>
<tr><th>Bits</th><th>Definition</th></tr>
<tr><td>07:06</td><td><b>LBA Change Information Attribute (LBACIR):</b> This field indicates attributes associated with the reporting of the LBA range in this entry.
If the ESA field is cleared to 000b, then this field shall be cleared to 00b.
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>00b</td><td>This entry is reporting logical blocks that have changed in the reported namespace. The SLBA field and the NLB field identify the LBA range.</td></tr>
<tr><td>01b</td><td>This entry is reporting that all logical blocks in the reported namespace have changed.</td></tr>
<tr><td>10b</td><td>This value indicates that no LBA range is being reported by this entry.</td></tr>
<tr><td>11b</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>05</td><td><b>Deallocated LBAs (DLBA)</b>: If this bit is set to ‘1’, then the logical blocks reported by this entry have been deallocated.
If this bit is cleared to ‘0’, then the logical blocks reported by this entry have been modified (e.g., been written or deallocated).</td></tr>
<tr><td>04</td><td>Reserved</td></tr>
<tr><td>03:01</td><td><b>Entry Sequence Attribute (ESA):</b> This field specifies the attribute of this entry related to starting and stopping of the posting of entries into the LBA Migration Queue.
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>000b</td><td>This entry is not the:
• first entry placed into the LBA Migration Queue since:
○ logging was started; or
○ the controller associated with the LBA Migration Queue resumed from a suspended state;
and
• last entry placed into the LBA Migration Queue since logging was stopped or the controller associated with the LBA Migration Queue was suspended.</td></tr>
<tr><td>001b</td><td>This is the first entry placed into the LBA Migration Queue since:
• logging was started; or
• the controller associated with the LBA Migration Queue resumed from a suspended state.</td></tr>
<tr><td>010b</td><td>This is the last entry placed into the LBA Migration Queue as a result of logging being stopped due to a request from the host.</td></tr>
<tr><td>011b</td><td>This is the last entry placed into the LBA Migration Queue as a result of the controller associated with the LBA Migration Queue being suspended.</td></tr>
<tr><td>100b to 110b</td><td>Reserved</td></tr>
<tr><td>111b</td><td>This is the last entry placed into the LBA Migration Queue due to the LBA Migration Queue becoming full. This value notifies the host that the controller has stopped logging logical block changes into that LBA Migration Queue.</td></tr>
</table></td></tr>
<tr><td>00</td><td><b>Controller Data Queue Phase Tag (CDQP):</b> This bit identifies whether a LBA Migration Queue entry is new as defined by the Controller Data Queue Phase Tag section in the NVM Express Base Specification.</td></tr>
</table></td></tr>
</table>

…

Description of Specification Changes for the NVM Express Management Interface Specification 1.2d

Modify a portion of section 6 as shown below:

6 NVM Express Admin Command Set

…

Figure 116: List of NVMe Admin Commands Supported using the Out-of-Band Mechanism

Command	Opcode	NVMe Storage Device O/M/P [1]	NVMe Enclosure O/M/P [1]	Reference Specification
…				
Lockdown	24h	O	O	NVM Express Base Specification
Track Send	3Dh	P	P	NVM Express Base Specification
Track Receive	3Eh	P	P	NVM Express Base Specification
Migration Send	41h	P	P	NVM Express Base Specification
Migration Receive	42h	P	P	NVM Express Base Specification
Controller Data Queue	45h	P	P	NVM Express Base Specification
Doorbell Buffer Config	7Ch	P	P	NVM Express Base Specification
…				

…

6.5 Set Features and Get Features

…

Figure 127: Management Endpoint - Feature Support

Feature Name [2]	Feature Identifier	Support Requirements [1]	
		NVMe Storage Device	NVMe Enclosure
…			
Rotational Media Information	16h	O	O
Controller Data Queue	21h	P	P
Vendor Specific	C0h to FFh	O	O
…			

…