LEGAL NOTICE:

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

NVM Express Technical Proposal for New Feature

Technical Proposal ID	**TP 4098 Multiple Atomicity**
Change Date	**2022-01-19**
Builds on Specification	**NVM Express NVM Command Set Specification 1.0a**
References Specification	

Technical Proposal Author(s)

Name	**Company**
Kapil Karkra, Jonathan Hughes, Jackson Ellis	Intel
David Black	Dell EMC

This proposal intends to:

- Define the new multiple atomic operation in terms of existing atomicity parameters.

Revision History

Revision Date	Change Description
2021-05-06	Initial rough version
2021-06-07	• Incorporated the following feedback from the 05/06/2021 Technical WG meeting ○ Make the scope of multiple atomicity namespace instead of controller ○ Remove unchanged text and use ellipses as shorthand. ○ Modify the picture to illustrate multiple boundaries clearly ○ Add an editorial sentence that multiple atomicity doesn't apply to Compare and Write fused pair.
2021-06-10	• Incorporated the following feedback from the 06/10/2021 Technical WG meeting ○ Rephrased the definition of MAM to clarify namespace scope and fixed some formatting errors. ○ Achieved phase-2 exit
2021-06-28	• Phase-3 draft version ○ Made the language consistent with the original atomicity section, including the Atomicity Parameters table. ○ Modified the Multiple Atomicity picture to show NABO as a specific LBA offset where the atomic granularities begin. Then the next boundary starts at Offset + y * Size for y = 1, 2, 3, and so on. ○ Added the qualification that only a single subrange is allowed in each atomic granularity. ○ Restored previous MAM definition to explicitly state that atomicity guarantees are provided by the controller for the namespace.
2021-07-01	• Incorporated feedback from 07/01/2021 Technical WG meeting ○ Moved the MAM bit to the NSFEAT field of the identify namespace data structure (CNS 00h) ○ Rephrased the MAM definition yet again to avoid using the term controller. Workgroup did acknowledge that while the bit's scope is namespace, controller is the one meeting the atomicity requirements, but it makes more sense to simply avoid the term controller. ○ Added "in multiple atomicity mode" to make sure the new normative rules are applicable to Multiple Atomicity Mode.
2021-07-13	• Editing pass: ○ More precision on atomic *write* operations ○ Use already-defined Namespace Atomic Boundary term to spec MAM ○ Exclude MAM from existing Atomic Boundaries text. ○ Rewrite MAM mode text, cribbing from existing Atomic Boundaries text ○ Spell out MAM parameter restrictions, recommend (should) that MAM normal and power fail atomicity be the same. ○ Rework example description to better contrast the two modes.
2021-07-20	• More editing. ○ Use atomic LBA subrange to indicate range of LBAs between atomic boundaries to which MAM provides an atomicity guarantee. ○ Move Single Atomicity Mode text into its own subsection and provide a short atomicity mode overview. ○ Reformat NSFEAT bits into a table, as suggested by Mike Allison.
2021-07-22	• Incorporated feedback from 07/22/20201 Technical WG meeting ○ Modified the 2.4.1 opening introductory text ○ Incorporated Mike Allison's 07/21 email feedback (typographical and capitalization errors). ○ Change description of new figure for Multiple Atomicity Mode to say that it shows differences from Single Atomicity Mode. ○ Reworked text that describes example. ○ Bring Feature Enhancement bullets up to date with changes to TP.

2021-08-03	• More editorial cleanup.
	  ○ State that any command in Multiple Atomicity Mode that does not cross any atomic boundary is processed as if in Single Atomicity Mode.
	  ○ Remove changes to fused operation (Compare & Write) atomicity to leave existing requirements intact.
	  ○ Lots of additional edits.

2021-08-05	• Minor updates from meeting, reset change tracking • Editorial corrections from Mike Allison • Additional minor editorial cleanups
2021-08-12	• Meeting decision: Multiple Atomicity Mode atomic write behavior shall be the same for normal and power fail conditions. • Editorial corrections from Ed Hsieh, including fixing off-by-one figure numbers and getting section numbers right. Resulted in a lot of cross-reference updates. • Add Multiple Atomicity mode modifications to AWUN and AWUPF field descriptions • Additional minor changes from meeting discussion

2021-08-20	• Rebased to NVM Command Set Specification 1.0a – Ratified 2021.07.26
	  ○ Format change applied to Figure 4 ((un-bolded some first column text)
	  ○ Removed sections 2.1.2, 2.1.3.1, and 4.1.3.5 from the TP, and instead added green text note to editor to fix references throughout the document.
	  ○ Removed 1.0a spec's sections 2.1.4.1, 2.1.4.1.1, 2.1.4.2, 2.1.4.2.1, and 2.1.4.2.2. All these sections now have a different section number but are not a concern of this TP. Added a green text note for the editor.
	  ○ Highlighted the links to be inserted in the new text
	  ○ Fixed section numbering and ordered changes with ascending section numbers

2021-08-25	• Minor changes to metatext
	  ○ Builds on NVM Express NVM Command Set Specification 1.0a
	  ○ Highlighted the note to editor about section number changes

2021-08-27	• Incorporated feedback from 8/26/2021 TWG
	  ○ Accepted all previously tracked changes
	  ○ Deleted all comments
	  ○ Changed the sections numbers of two new sections as 2.1.4.TBD1 and 2.1.4.TBD2 and kept the rest of the sections same as the spec 1.0a.
	  ○ Adjusted all references based on the above change
	  ○ Made minor edits based on the TWG feedback

2021-09-03	• Clean up atomicity parameter interaction with atomic boundaries – controller-scope atomicity parameters (e.g., AWUN) apply to atomic boundary functionality if namespace-scope atomicity parameters (e.g., NAWUN) are not reported. This approach was referred to as Option B in working group discussion.
2021-09-09	• Clean version with final edits for member review. Primary change is conversion of the "i.e." list of atomic boundary size requirements to an itemized list.
2021-12-20	• Integrated
2022-01-18	• Added missing comma and updated the value of bit 6 in the NSFEAT field. Formatted to remove pages with limited text.

Description for NVM Command Set Specification Revision 1.0a Changes Document

Feature Enhancement:

- Define a new Multiple Atomicity Mode (MAM) that provides write guarantees between Namespace Atomic Boundaries for commands that cross Namespace Atomic Boundaries.
- Atomicity behavior specified in the NVM Command Set Specification Revision 1.0a and prior NVMe standards is referred to as Single Atomicity Mode with no functional changes.

- Added an MAM bit to the Namespace Features (NSFEAT) field in the I/O Command Set specific Identify Namespace data structure (CSI 00h) to indicate that Multiple Atomicity Mode applies to writes to a namespace.
- Added a new section to specify normative rules for Multiple Atomicity Mode.
- References:
    - Section 2.1.4 Atomic Operation - restructured and extended
    - NVM Command Set Identify Namespace data structure (CNS 00h)

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.
<Green Bracketed>:	Notes to editor

Modify portions of NVM Command Set Specification 1.0a as follows:

< Note to editor:

This TP inserts a new section 2.1.4.TBD1 that changes the section numbers of existing sections immediately after 2.1.4.TBD1. These existing sections are referenced from other parts of this spec. Therefore, this change requires revising all references to 2.1.4.x or 2.1.4.x.x throughout the spec. At the time of writing of this TP, the following sections had references that needed revising:

- Section 2.1.2
- Section 2.1.3.1
- Section 2.1.4.1 (in two places)
- Section 2.1.4.2
- Section 4.1.3.5

>

Modify section 2.1.4 as shown below:

2.1.4 Atomic Operation

A controller is in either Single Atomicity Mode (refer to section 2.1.4.TBD1) or Multiple Atomicity Mode (refer to section 2.1.4.TBD2) for each attached namespace. In Single Atomicity Mode, controller processing of a write command results in at most one atomic write operation, and controller processing of a write command that crosses any Namespace Atomic Boundary (refer to section 2.1.4.3) may or may not result in an atomic write operation. In Multiple Atomicity Mode, controller processing of a write command that crosses any Namespace Atomic Boundary results in multiple atomic write operations.

Multiple Atomicity Mode is a superset of Single Atomicity Mode. A host that is not aware of Multiple Atomicity Mode (i.e., is operating under the assumption that write commands are processed in Single Atomicity Mode) does not encounter unexpected or incompatible behavior.

2.1.4.TBD1 Atomic Operation in Single Atomicity Mode

In Single Atomicity Mode, controller processing of a write command results in at most one atomic write operation. Figure 4 is an overview of the parameters that define the controller's support for Single Atomicity Mode atomic write operation~~s~~. These parameters may affect command behavior and execution order based on write size (on a per controller or a per namespace basis).

Figure 4: Atomicity Parameters for Single Atomicity Mode

	Parameter Name	Value[1]
Controller Atomic Parameters (refer to Figure 99)	Atomic Write Unit Normal (AWUN)	
	Atomic Write Unit Power Fail (AWUPF)	≤ AWUN
	Atomic Compare and Write Unit (ACWU)	
sNamespace Atomic Parameters (refer to the Identify Namespace data structure in Figure 97)	Namespace Atomic Write Unit Normal (NAWUN)	≥ AWUN
	Namespace Atomic Write Unit Power Fail (NAWUPF)	≥ AWUPF and ≤ NAWUN
	Namespace Atomic Compare and Write Unit (NACWU)	≥ ACWU
Namespace Atomic Boundary Parameters (refer to the Identify Namespace data structure in Figure 97)	Namespace Atomic Boundary Size Normal (NABSN)	≥ NAWUN and ≥ AWUN
	Namespace Atomic Boundary Offset (NABO)	≤ NABSN and ≤ NABSPF
	Namespace Atomic Boundary Size Power Fail (NABSPF)	≥ NAWUPF and ≥ AWUPF

NOTES:
1. When the parameter is supported, the value shall meet the listed condition(s).

The NVM subsystem reports in the Identify Controller data structure the size in logical blocks of the write operation guaranteed to be written atomically under various conditions, including normal operation, power fail, and in a Compare & Write fused operation (i.e., the maximum size of an atomic write operation under each condition). The values reported in the Identify Controller data structure are valid across all namespaces with any supported namespace format, forming a baseline value that is guaranteed not to change.

An NVM subsystem may report per namespace values for these ~~fields~~ atomicity parameters that are specific to the namespace ~~format~~ and are indicated in the Identify Namespace data structure (refer to Figure 97). If an NVM subsystem reports a per namespace value, then that value shall be greater than or equal to the corresponding baseline value indicated in the Identify Controller data structure (refer to Figure 99).

The values are reported in the fields (Namespace) Atomic Write Unit Normal, (Namespace) Atomic Write Unit Power Fail, and (Namespace) Atomic Compare & Write Unit in the Identify Controller data structure or the Identify Namespace data structure depending on whether the values are the baseline or namespace specific.

A controller may support Atomic Boundaries that shall not be crossed by an atomic write operation. The Namespace Atomic Boundary Parameters (NABSN, NABO, and NABSPF) define these boundaries for a namespace. A namespace supports Atomic Boundaries if NABSN or NABSPF is set to a non-zero value. For a ~~A~~ namespace that does not support Atomic Boundaries, the controller shall clear the NABSN and NABSPF fields to 0h. Namespace Atomicity Parameter and Namespace Atomic Boundary Parameter values may be format specific and may change if the namespace format is modified.

In the case of a shared namespace, operations performed by an individual controller are atomic to the shared namespace at the write atomicity level reported in the corresponding Identify Controller or Identify Namespace data structures of the controller to which the command was submitted.

…

2.1.4.3 Atomic Boundaries

Atomic Boundaries control how the atomicity guarantees defined in section 2.1.4 are enforced by the controller, with the added constraint of the alignment of the LBA range specified in the command. Atomic Boundaries are defined on a per namespace basis only. The namespace supports Atomic Boundaries if NABSN or NABSPF are set to non-zero values.

To ensure backwards compatibility, the values reported for AWUN, AWUPF, and ACWU shall be set such that they are supported even if a write crosses an atomic boundary. If a controller does not guarantee atomicity across atomic boundaries, the controller shall set AWUN, AWUPF, and ACWU to 0h (1 LBA).

The boundary sizes shall be greater than or equal to the corresponding atomic write sizes: ~~(i.e.,~~

- NABSN~~/NABSPF~~ shall be greater than or equal to AWUN;
- NABSN shall be greater than or equal to NAWUN if NAWUN is reported;
- NABSPF shall be greater than or equal to AWUPF~~/NAWUPF~~; and
- NABSPF shall be greater than or equal to NAWUPF if NAWUPF is reported ~~respectively)~~.

In addition, NABO shall be less than or equal to NABSN and NABSPF.

For Boundary Offset (NABO) and Boundary Size (NABSN or NABSPF), the LBA range in a command is within a Namespace Atomic Boundary if none of the logical block addresses in the range cross: Boundary Offset + (y * Boundary Size); for any integer y ≥ 0.

If a write command crosses the atomic boundary specified by the NABO and NABSN values, then for Single Atomicity Mode, the atomicity based on the NAWUN ~~parameters~~ value is not guaranteed. If a write command crosses the atomic boundary specified by the NABO and NABSPF values, then for Single Atomicity Mode, the atomicity based on the NAWUPF ~~parameters~~ value is not guaranteed. Atomicity guarantees for Multiple Atomicity Mode are specified in Section 2.1.4.TBD2.

Figure 8 shows an example of the behavior of Atomic Boundaries. Writes to an individual blue or yellow section do not cross an atomic boundary.

Figure 8: Atomic Boundaries Example
[no change to diagram, diagram omitted from TP doc because copy/paste corrupted colors]

2.1.4.TBD2 Atomic Operation in Multiple Atomicity Mode

In Multiple Atomicity Mode, controller processing of a write command that crosses any Namespace Atomic Boundary results in multiple atomic write operations.

Figure TBD-1 is an overview of the Multiple Atomicity Mode controller parameters that differ from Single Atomicity Mode. These parameters may affect command behavior and execution order based on write size (on a per controller or a per namespace basis).

Figure TBD-1: Atomicity Parameter Differences for Multiple Atomicity Mode

	Parameter Name	Value (Multiple Atomicity Mode)	Value (Single Atomicity Mode)
Namespace Atomic Boundary Parameters (refer to the Identify Namespace data structure in Figure 97)	Namespace Atomic Boundary Size Normal (NABSN)	= NAWUN, = AWUN if NAWUN is not reported, and = NABSPF	≥ NAWUN and ≥ AWUN
	Namespace Atomic Boundary Size Power Fail (NABSPF)	= NAWUPF, = AWUPF if NAWUPF is not reported, and = NABSN	≥ NAWUPF and ≥ AWUPF
	Namespace Atomic Boundary Offset (NABO)	≤ NABSN and ≤ NABSPF (same requirement for both modes)	

In Multiple Atomicity Mode, atomicity guarantees for a write command that does not cross a Namespace Atomic Boundary are the same as Single Atomicity Mode (refer to section 2.1.4.3).

In Multiple Atomicity Mode, controller processing of a write command that crosses any Namespace Atomic Boundary divides the command-specified range of LBAs into LBA subranges at Namespace Atomic Boundaries and performs an atomic write operation on each resulting LBA subrange, called an atomic LBA subrange. The atomicity guarantees apply separately to each atomic LBA subrange. The atomicity value requirements in Figure TBD-1 ensure that each LBA in the command-specified LBA range is included in an atomic write operation (i.e., is part of an atomic LBA subrange).

For Multiple Atomicity Mode, atomicity parameter requirements depend on whether the NVM Subsystem reports per namespace values for the atomicity parameters:

a. if per namespace values are reported, then the controller shall set the NABSN field, the NAWUN field, the NABSPF field, and the NAWUPF field to the same value; and
b. if per namespace values are not reported, then the controller shall set the NABSN field, the AWUN field, the NABSPF field, and the AWUPF field to the same value.

In Multiple Atomicity Mode, the write operation on each atomic LBA subrange is atomic to the NVM with respect to other read or write commands. This applies to both normal operating conditions and operation if a power fail or other error condition interrupts a write operation causing a torn write.

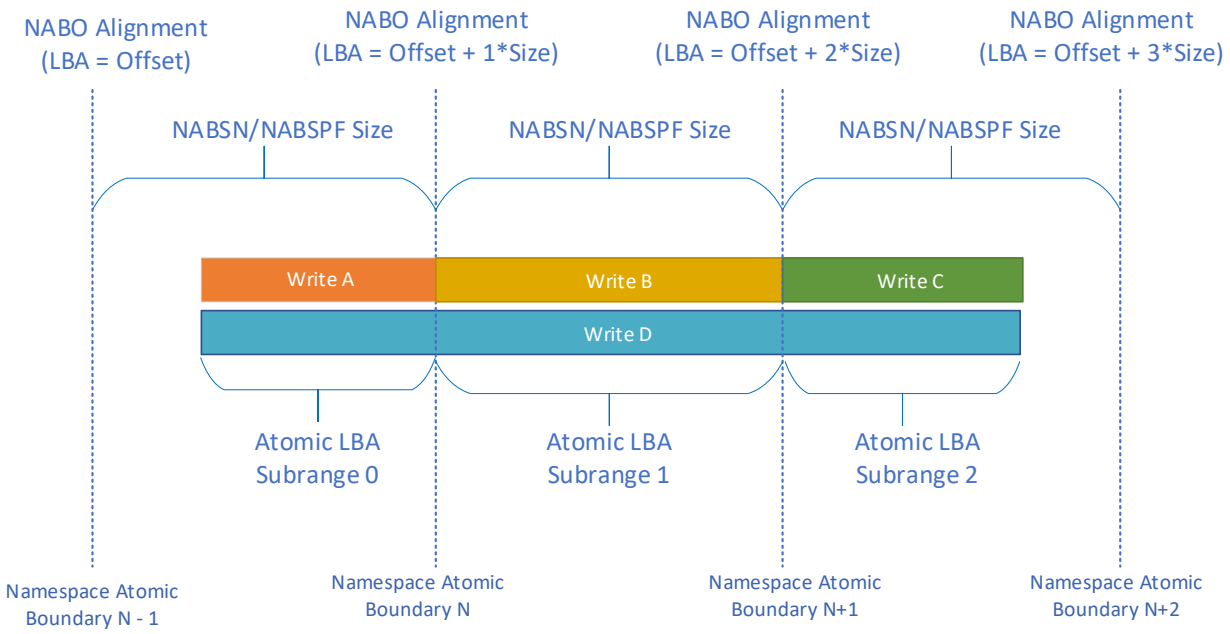
Multiple Atomicity Mode does not affect fused operations (refer to section 2.1.3). All write operations performed by a command that is part of a fused operation shall be performed in Single Atomicity Mode (refer to section 2.1.4.TBD1).

2.1.4.TBD2.1 Namespace Atomic Boundaries in Single and Multiple Atomicity Modes

In Single Atomicity Mode write commands that cross Namespace Atomic Boundaries obtain no atomicity guarantees. For example, as shown in Figure TBD-2, a single write command D that crosses Namespace Atomic Boundaries obtains no atomicity guarantees. Three separate write commands (A, B, and C) are necessary to obtain atomicity guarantees for the LBA subranges between Namespace Atomic Boundaries.

In contrast, in Multiple Atomicity Mode, write command D results in an atomicity guarantee for each LBA subrange obtained by dividing the LBA range at Namespace Atomic Boundaries (i.e., atomicity guarantees apply to LBA subranges 0, 1 and 2). In this Multiple Atomicity Mode example, a single write command (D) in obtains atomicity guarantees that require three write commands (A, B and C) to obtain in Single Atomicity Mode.

Figure TBD-2: Multiple Atomicity Example



Modify section 4.1.5.1 as shown below:

4.1.5.1 NVM Command Set Identify Namespace data structure (CNS 00h)

…

Figure 97: Identify – Identify Namespace Data Structure, NVM Command Set

<table>
<tr><th>Bytes</th><th>O/M[1]</th><th>Description</th></tr>
<tr><td>…</td><td></td><td></td></tr>
<tr><td>24</td><td>M</td><td>
Namespace Features (NSFEAT): This field defines features of the namespace.

~~Bits 7:5 are reserved.~~

~~Bit 4 (OPTPERF) if set to ‘1’ indicates that the fields NPWG, NPWA, NPDG, NPDA, and NOWS are defined for this namespace and should be used by the host for I/O optimization (refer to the NVM Set List section in the NVMe Base Specification). If cleared to ‘0’, then the controller does not support the fields NPWG, NPWA, NPDG, NPDA, and NOWS for this namespace.~~

~~Bit 3 (UIDREUSE) This bit is as defined in the UIDREUSE bit in the I/O Command Set Independent Identify Namespace data structure (refer to the I/O Command Set Independent Identify Namespace data structure section in the NVMe Base Specification).~~

~~Bit 2 (DAE) if set to ‘1’ indicates that the controller supports the Deallocated or Unwritten Logical Block error for this namespace. If cleared to ‘0’, then the controller does not support the Deallocated or Unwritten Logical Block error for this namespace. Refer to section 3.2.3.2.1.~~

~~Bit 1 (NSABP) if set to ‘1’ indicates that the fields NAWUN, NAWUPF, and NACWU are defined for this namespace and should be used by the host for this namespace instead of the AWUN, AWUPF, and ACWU fields in the Identify Controller data structure. If cleared to ‘0’, then the controller does not support the fields NAWUN, NAWUPF, and NACWU for this namespace. In this case, the host should use the AWUN, AWUPF, and ACWU fields defined in the Identify Controller data structure in the NVMe Base Specification. Refer to section 0.~~

~~Bit 0 (THINP) if set to ‘1’ indicates that the namespace supports thin provisioning. If cleared to ‘0’ indicates that thin provisioning is not supported Refer to section 2.1.1 for details on the usage of this bit.~~
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7:~~5~~</td><td>Reserved</td></tr>
<tr><td>6</td><td>Multiple Atomicity Mode (MAM): if set to ‘1’, then Multiple Atomicity Mode (refer to section 2.1.4.TBD1) applies to write operations to this namespace. If cleared to ‘0’, then Single Atomicity Mode (refer to section 2.1.4.TBD1) applies to write operations to this namespace.</td></tr>
<tr><td>4</td><td>OPTPERF: if set to ‘1’ indicates that the fields NPWG, NPWA, NPDG, NPDA, and NOWS are defined for this namespace and should be used by the host for I/O optimization (refer to the NVM Set List section in the NVMe Base Specification). If cleared to ‘0’, then the controller does not support the fields NPWG, NPWA, NPDG, NPDA, and NOWS for this namespace.</td></tr>
<tr><td>3</td><td>UIDREUSE: This bit is as defined in the UIDREUSE bit in the I/O Command Set Independent Identify Namespace data structure (refer to the I/O Command Set Independent Identify Namespace data structure section in the NVMe Base Specification).</td></tr>
<tr><td>2</td><td>DAE: if set to ‘1’ indicates that the controller supports the Deallocated or Unwritten Logical Block error for this namespace. If cleared to ‘0’, then the controller does not support the Deallocated or Unwritten Logical Block error for this namespace. Refer to section 3.2.3.2.1.</td></tr>
<tr><td>1</td><td>NSABP: if set to ‘1’ indicates that the fields NAWUN, NAWUPF, and NACWU are defined for this namespace and should be used by the host for this namespace instead of the AWUN, AWUPF, and ACWU fields in the Identify Controller data structure. If cleared to ‘0’, then the controller does not support the fields NAWUN, NAWUPF, and NACWU for this namespace. In this case, the host should use the AWUN, AWUPF, and ACWU fields defined in the Identify Controller data structure in the NVMe Base Specification. Refer to section 2.1.4.</td></tr>
<tr><td>0</td><td>THINP: if set to ‘1’ indicates that the namespace supports thin provisioning. If cleared to ‘0’ indicates that thin provisioning is not supported Refer to section 2.1.1 for details on the usage of this bit.</td></tr>
</table>
</td></tr>
<tr><td>…</td><td></td><td></td></tr>
</table>

Modify section 4.1.5.2 as shown below:

…

Figure 99: Identify – Identify Controller data structure, NVM Command Set Specific Fields

Bytes	O/M[1]	Description
527:526	M	**Atomic Write Unit Normal (AWUN):** This field indicates the size of the write operation guaranteed to be written atomically to the NVM across all namespaces with any supported namespace format during normal operation. This field is specified in logical blocks and is a 0's based value. If a specific namespace guarantees a larger size than is reported in this field, then this namespace specific size is reported in the NAWUN field in the Identify Namespace data structure. Refer to section 2.1.4. If a write command is submitted with size less than or equal to the AWUN value, the host is guaranteed that the write command is atomic to the NVM with respect to other read or write commands. If a write command is submitted with size greater than the AWUN value, then there is no guarantee of command atomicity, but atomicity is guaranteed for portions of the command if the command is processed in Multiple Atomicity Mode (refer to section 2.1.4.5). AWUN does not have any applicability to write errors caused by power failure (refer to Atomic Write Unit Power Fail). A value of FFFFh indicates all commands are atomic as this is the largest command size. It is recommended that implementations support a minimum of 128 KiB (appropriately scaled based on LBA size).
529:528	M	**Atomic Write Unit Power Fail (AWUPF):** This field indicates the size of the write operation guaranteed to be written atomically to the NVM across all namespaces with any supported namespace format during a power fail or error condition. If a specific namespace guarantees a larger size than is reported in this field, then this namespace specific size is reported in the NAWUPF field in the Identify Namespace data structure. Refer to section 2.1.4. This field is specified in logical blocks and is a 0's based value. The AWUPF value shall be less than or equal to the AWUN value. If a write command is submitted with size less than or equal to the AWUPF value, the host is guaranteed that the write is atomic to the NVM with respect to other read or write commands. If a write command is submitted that is greater than this size, there is no guarantee of command atomicity, but atomicity is guaranteed for portions of the command if the command is processed in Multiple Atomicity Mode (refer to section 2.1.4.5). If the write size is less than or equal to the AWUPF value and the write command fails, then subsequent read commands for the associated logical blocks shall return data from the previous successful write command. If a write command is submitted with size greater than the AWUPF value, then there is no guarantee of data returned on subsequent reads of the associated logical blocks.

…