LEGAL NOTICE:

© Copyright 2007 - 2021 NVM Express, Inc. ALL RIGHTS RESERVED.
This erratum to the NVMe over Fabrics revision 1.1 specification 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 erratum to the NVMe over Fabrics revision 1.1 specification 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: **"© 2007 - 2021 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

Technical input submitted to the NVM Express™ Workgroup is subject to the terms of the NVM Express™ Participant's agreement. Copyright © 2014-2021 NVMe™ Corporation.

info@nvmexpress.org

NVM Express™ Technical Errata

Errata ID	**002**
Revision Date	**04/23/2021**
Affected Spec Ver.	**NVMe over Fabrics 1.1**
Corrected Spec Ver.	

Errata Author(s)

Name	Company
David Peterson	Broadcom
Constantine Sapuntzakis	Pure Storage
Claudio DeSanti	DellEMC
David Black	DellEMC
Fred Knight	NetApp

Errata Overview

This errata updates:

Section 2.1 Figure 7: Fabrics Command Capsule PDU - Submission Queue Entry Format

Section 2.3.1 Data and SGL Locations within a Command Capsule

Section 7.4.5 text

Section 7.4.5.2 text

Section 7.4.6 text

Section 7.4.9 text

Section 7.4.9.1 text

Section 7.4.10.3 text

Section 7.4.10.6 Figure 77: Command Capsule PDU (CapsuleCmd)

Section 7.4.10.8 Figure 79: Host to Controller Data Transfer PDU (H2CData)

Section 7.4.10.9 Figure 80: Controller to Host Data Transfer PDU (C2HData)

Technical input submitted to the NVM Express™ Workgroup is subject to the terms of the NVM Express™ Participant's agreement. Copyright © 2014-2021 NVMe™ Corporation.

Revision History

Revision Date	Change Description
09/10/2020	Creation
12/15/2020	Added additional ECN material for review.
02/09/2021	Updates from work group review.
04/22/2021	Integrated into the NVMe over Fabrics Specification, revision 1.1.
04/23/2021	Removed the strikeout of an H.

Incompatible Changes

None

Description of Specification Changes

Modify section 2.1 as shown below:

Figure 7: Fabrics Command Capsule PDU - Submission Queue Entry Format

Bytes	Description
00	**Opcode (OPC):** Set to 7fh to indicate a Fabrics command.
01	~~Reserved~~**PSDT:** This field is described in the NVMe Base Specification. Bits 7:6 of this field should be set to 01b, and may be set to 00b.
03:02	**Command Identifier (CID):** This field specifies a unique identifier for the command. The identifier shall be unique among all outstanding commands associated with a particular queue.
04	**Fabrics Command Type (FCTYPE):** The field specifies the Fabrics command transferred in the capsule. The Fabrics command types are defined in Figure 14. If this field is set to a reserved value, the command should be aborted with a status code of Invalid Field in Command.
39:05	Reserved
63:40	**Fabrics Command Type Specific:** This field is Fabrics command type specific.

Modify section 2.3.1 as shown below:

2.3.1 Data and SGL Locations within a Command Capsule

The Submission Queue Entry within the command capsule includes one SGL entry. If there are additional SGL entries to be transferred in the command capsule, then those entries shall be contiguous and located immediately after the Submission Queue Entry.

An NVMe Transport binding specification defines the support for data as part of the command capsule. The controller indicates the starting location of data within a command capsule via the In Capsule Data Offset (ICDOFF) field in the Identify Controller data structure.

Technical input submitted to the NVM Express™ Workgroup is subject to the terms of the NVM Express™ Participant's agreement. Copyright © 2014-2021 NVMe™ Corporation.

There are restrictions for SGLs that the host should follow:

- If the ICDOFF field is a non-zero value, then all ~~the~~ SGL descriptors following the Submission Queue Entry shall not have a total size greater than (ICDOFF * 16); ~~and~~
- if the SGL descriptors following the Submission Queue Entry have a total size greater than (ICDOFF * 16), then the controller shall abort the command with the status code set to Invalid Number of SGL Descriptors;
- the host shall not place more SGL Data Block or Keyed SGL Data Block descriptors within a capsule than the maximum indicated in the Identify Controller data structure~~.~~; and
- if the host places more SGL Data Block of Keyed SGL Data Block descriptors in a capsule than the maximum indicated in the Maximum SGL Data Block Descriptors field in the Identify Controller data structure, then the controller shall abort the command with the status code set to Invalid Number of SGL Descriptors.

The host shall start data (if present) in command capsules at byte offset (ICDOFF * 16) from the end of the Submission Queue Entry.

Modify section 7.4.5 as shown below:

The TCP transport is a message-based transport that uses capsules for data transfer as defined in the Capsules and Data Transfer section of the NVMe Base Specification. All NVMe/TCP implementations shall support data transfers using ~~c~~Command ~~d~~Data ~~b~~Buffers (described in section 7.4.5.1) and may optionally support in-capsule data.

Host and ~~C~~controller PDU Data is optionally aligned. PDU ~~d~~Data alignment is designed to allow the host or controller to guarantee that the data (and data digest) starting offset ~~to~~ be aligned to some value (usually a cache line). The alignment of data in a PDU is specified by the host and the controller when a connection is established. The ~~Controller to~~ Host PDU Data Alignment (HPDA) field in the ICReq PDU specifies the required alignment of PDU Data (DATA) from the start of the PDU for PDUs that are transferred from the controller to the host. The ~~Host to~~ Controller PDU Data Alignment (CPDA) field in the ICResp PDU specifies the required alignment of PDU Data (DATA) from the start of the PDU for PDUs that are transferred from the host to the controller. An appropriate number of padding bytes shall be inserted by the controller or host in the PAD field to achieve the required alignment. The number of PAD bytes is a function of the required alignment and the size of the PDU ~~h~~Header. PDU PAD bytes are considered as reserved bytes and are not protected by HDGST nor by DDGST. Neither ~~T~~the Host ~~and Controller~~ PDU Data Alignment field (HPDA~~, CPDA~~) nor the Controller PDU Data Alignment field (CPDA) shall ~~not~~ exceed 128 bytes.

Figure 64 shows an example of an H2CData PDU where the CPDA field (refer to section 7.4.10.3) was set to 0Fh by the ~~host~~controller in the ICResp when the connection was established. The H2CData PDU ~~h~~Header size 24 bytes and header digest is disabled. An alignment of 64 bytes is required, thus the host inserts 40 bytes of padding in the PAD field.

Technical input submitted to the NVM Express™ Workgroup is subject to the terms of the NVM Express™ Participant's agreement. Copyright © 2014-2021 NVMe™ Corporation.

[No change to figure 64]

Figure 65 shows an example of a C2HData PDU where the HPDA field (refer to section 7.4.10.2) was set to 03h by the host in the ICReq when the connection was established. The C2HData PDU ~~h~~Header size 24 bytes and header digest is disabled. An alignment of 16 bytes is required, thus the controller inserts 8 bytes of padding in the PAD field.

Figure 65: Example of 64B PDU DATA Alignment in C2HData~~H2CData~~ PDU

[No change to figure 65]

Modify section 7.4.5.2 as shown below:

7.4.5.2 ~~Controller to~~ Host to Controller Command Data Buffer Transfers

Command ~~d~~Data ~~b~~Buffer transfers from a host to a controller parallel the behavior of ~~c~~Command ~~d~~Data ~~b~~Buffer transfers from a controller to a host described in section 7.4.5.2. They are performed from a host to a controller using one or more H2CData PDUs. The data transferred by the H2CData PDUs starts at the beginning of the ~~c~~Command ~~d~~Data ~~b~~Buffer and continues sequentially to the end of the ~~c~~Command ~~d~~Data ~~b~~Buffer (i.e., the DATAO field in the first H2CData PDU is cleared to 0h and the DATAO field in a subsequent H2CData PDU is equal to the DATAO field plus the DATAL field of the previous H2CData PDU).

Reception of a non-contiguous H2CData PDU is treated as a fatal transport error with the Fatal Error Status field set to “PDU Sequence Error”.

The controller governs the rate of the data transfer from the host to the controller using Ready to Transfer (R2T) PDUs. When a connection is established, the host specifies the maximum number of outstanding R2T PDUs (MAXR2T) that the host supports at any point in time for a command. The first R2T PDU of a command shall start with an offset of zero and subsequent R2T PDUs for the command shall solicit data transfers sequentially to the end of the ~~c~~Command ~~d~~Data ~~b~~Buffer (i.e., the R2TO field in the first R2T PDU is cleared to 0h and the R2TO field in subsequent R2T PDUs shall be equal to the R2TO field plus R2TL field of the previous R2T PDU).

H2CData PDUs are sent in response to R2T PDUs and are never sent without receiving a corresponding R2T PDU. The R2T PDU specifies the ~~c~~Command ~~i~~Identifier (refer to the Submission Queue Entry section in the NVMe Base Specification) with which it is associated, a transfer tag (TTAG), a data offset (R2TO) from the beginning of the ~~c~~Command ~~d~~Data ~~b~~Buffer, and the transfer data length (R2TL). When an R2T PDU is received by a host, ~~then~~ the host transfers the data requested by the controller ~~in~~as indicated by the R2T PDU. This data transfer may be performed using one or more H2CData PDUs. H2CData PDUs ~~must~~ start at the offset specified in the R2T PDU (R2TO) and transfer data contiguously until the data transfer length specified by the R2T PDU (R2TL) is reached (i.e., the DATAO field in the first H2CData PDU is equal to R2TO that specified in the R2T PDU and DATAO field in subsequent PDUs is equal the DATAO field plus DATAL field of the previous H2CData PDU). The H2CData PDU length does not exceed the maximum length communicated by the controller during the connection establishment (MAXH2CDATA).

Technical input submitted to the NVM Express™ Workgroup is subject to the terms of the NVM Express™ Participant’s agreement. Copyright © 2014-2021 NVMe™ Corporation.

Reception of a non-contiguous R2T PDU is treated as a fatal transport error with the Fatal Error Status field in the C2HTermReq PDU set to “PDU Sequence Error”.

Reception of a H2CData PDU with a data length which exceeds MAXH2CDATA is treated as a fatal transport error with the Fatal Error Status field in the C2HTermReq PDU set to “Data Transfer Limit Exceeded”.

Reception of a H2CData PDU that is outside the range starting from R2TO to R2TO + R2TL is treated as a fatal transport error with the Fatal Error Status field in the C2HTermReq PDU set to “Data Transfer Out of Range”.

The LAST_PDU flag in H2CData PDUs is cleared to ‘0’ in all but the last H2CData PDU that is associated with a single R2T PDU. The LAST_PDU flag is set to ‘1’ in the last H2CData PDU that satisfies a R2T request. All H2CData PDUs used to satisfy data requested by a controller shall have their Transfer Tag (TTAG) field set to the transfer tag specified in the R2T PDU.

Reception of a H2CData PDU with an unknown ~~transfer tag (~~TTAG~~)~~ value is treated as a fatal transport error with the Fatal Error Status field set to “Invalid PDU ~~h~~Header ~~f~~Field” and the Additional Error Information field containing the TTAG field byte offset.

Reception of an unexpected H2CData PDU is treated as a fatal transport error with the Fatal Error Status field in the C2HTermReq PDU set to “PDU Sequence Error”.

Examples of an unexpected H2CData PDUs are:
• Reception of a H2CData PDU with the LAST_PDU flag cleared to ‘0’ after reception of a H2CData PDU with the LAST_PDU flag set to ‘1’ and is associated with the same R2T PDU.

R2T PDUs associated with a specific command shall be serviced in the order received by the host. R2T PDUs associated with different commands received by a host may be serviced in any order. A controller may send an R2T PDU without waiting for a previous R2T data transfer to complete, but shall not exceed the maximum number of outstanding R2T PDUs per command supported by the host (MAXR2T).

Reception of an R2T PDU that exceeds MAXR2T is treated as a fatal transport error with the Fatal Error Status field in the H2CTermReq PDU set to “R2T Limit Exceeded”.

Figure 68 illustrates a command that performs a 12,288 bytes (3000h bytes) host to controller ~~c~~Command ~~d~~Data ~~b~~Buffer transfer using R2T PDUs:

1. The host sends a Command Capsule PDU (CapsuleCmd) to the controller containing an SQE with a Transport SGL Data Block descriptor with a ~~s~~Sub ~~t~~Type value of Ah in SGL1. The Length field in the descriptor has a value of 3000h;
2. The controller processes the command and sends an R2T PDU to the host that requests 3,000 bytes (BB8h byte) of data with a data offset of zero;
3. When the host receives the R2T PDU, that host sends an H2CData PDU that transfers 1,000 bytes (3E8h bytes). The ~~R2T~~ Data Offset (~~R2TO~~DATAO) field is cleared to 0h, the ~~R2T~~ Data Length (~~R2TL~~DATAL) field is set to 3E8h, and the Last (LAST_PDU) flag is cleared to ‘0’ since this is not the last PDU of the data transfer;
4. The host sends a subsequent H2CData PDU that transfers 2,000 bytes (7D0h bytes). The DATAO field is set to 3E8h, the DATAL field is set to 7D0h, and the LAST_PDU

Technical input submitted to the NVM Express™ Workgroup is subject to the terms of the NVM Express™ Participant’s agreement. Copyright © 2014-2021 NVMe™ Corporation.

flag is set to ‘1’ since this is the last PDU of the data transfer requested by the R2T PDU;

5. The controller sends a subsequent R2T PDU that ~~transfers~~requests 8,192 bytes (2000h bytes). The R2TO field is set to BB8h and the R2TL field is set to 2000h;
6. The controller sends a subsequent R2T PDU that ~~transfers~~requests 1,096 bytes (448h bytes) if the host supports MAXR2T greater than one. The R2TO field is set to 2BB8h and the R2TL field is 448h;
7. The host sends a H2CData PDU that transfers 8,192 bytes (2000h bytes). The DATAO field is set to BB8h, the DATAL field is set to 2000h, and the LAST_PDU flag is set to ‘1’ since this is the last PDU of the data transfer;
8. The host send a subsequent H2CData PDU that transfers 1,096 bytes (448h bytes). The DATAO field is set to 2BB8h, the DATAL field is set to 448h, and the LAST_PDU flag is set to ‘1’ since this is the last PDU of the data transfer; and
9. The controller sends a Response Capsule PDU (CapsuleResp) to the host containing a CQE.

[No change to figure 68]

Modify section 7.4.6 as shown below:

NVMe/TCP facilitates an optional PDU Header Digest (HDGST) and Data ~~d~~Digest (DDGST). The presence of each digest is negotiated at the connection establishment.

The host requests the use of a header digest by setting the HDGST_ENABLE flag in the ICReq PDU. The controller may accept (or reject) the use of a header digest by setting (or clearing) the HDGST_ENABLE flag in the ICResp PDU. The PDU Header ~~d~~Digest is enabled if HDGST_ENABLE flag is set in both the ICReq and ICResp PDUs. If the PDU ~~h~~Header ~~d~~Digest is enabled, then all the subsequent PDUs transferred in this connection except H2CTermReq and C2HTermReq PDUs shall contain a HDGST field and have the HDGSTF flag set to ‘1’ in the PDU ~~h~~Header FLAGS field. If the PDU ~~h~~Header ~~d~~Digest is enabled, the header digest is contained within the HDGST field of the PDU and protects the PDU ~~h~~Header. If PDU ~~h~~Header ~~d~~Digest is not enabled, then all subsequent PDUs shall not contain a HDGST field and shall have the HDGSTF flag~~, if defined,~~ cleared to ‘0’ in the PDU ~~h~~Header FLAGS field.

The host requests the use of a data digest by setting the DDGST_ENABLE flag in the ICReq PDU. The controller may accept (or reject) the use of a data digest by setting (or clearing) the DDGST_ENABLE flag in the ICResp PDU. The PDU Data ~~d~~Digest is enabled if the DDGST_ENABLE flag is set in both the ICReq and ICResp PDUs. If the PDU ~~d~~Data ~~d~~Digest is enabled, then ~~all~~ Command Capsule PDUs containing in-capsule data, ~~and all~~ H2CData PDUs, and C2HData PDUs transferred in this connection shall contain a DDGST field and have the DDGSTF flag set to ‘1’ in the PDU ~~h~~Header FLAGS field. If the PDU ~~d~~Data ~~d~~Digest is not enabled, then these PDUs shall not contain a DDGST field and shall have the DDGSTF flag cleared to ‘0’ in the PDU ~~h~~Header FLAGS field. If data digest is enabled, the data digest is contained within the DDGST field of the PDU and protects the PDU ~~d~~Data.

Technical input submitted to the NVM Express™ Workgroup is subject to the terms of the NVM Express™ Participant’s agreement. Copyright © 2014-2021 NVMe™ Corporation.

If a host requests the use of header digest or data digest in the ICReq PDU, but the use of the digest was not enabled by the controller in the ICResp PDU, then the host may refuse the connection establishment and terminate the NVMe/TCP connection (refer to section 7.4.4).

If a host did not request the use of header digest or data digest in the ICReq PDU but the use of the digest was enabled by the controller in the ICResp PDU, then the host shall treat that ICResp PDU as a fatal transport error (refer to section 7.4.7) with the Fatal Error Status field set to "Invalid PDU ~~h~~Header ~~f~~Field" and the Additional Error Information field containing the ~~DIGEST~~ HDGST or DDGST field byte offset.

The HDGST ~~Header~~ and DDGST ~~Data digests~~ are calculated using the CRC32C algorithm (refer to http://www.rfc-editor.org/rfc/rfc3385.txt).

Modify section 7.4.9 paragraph 2 as shown below:

TLS implementation is optional for NVMe/TCP. TLS 1.2 implementation is discouraged in favor of TLS 1.3 due to implementation concerns described in section 7.4.9.1. TLS 1.3 implementation will be described in a future specification.

Modify section 7.4.9.1 as shown below:

The PSK cipher suite framework is described in RFC 4279. NVMe/TCP uses NQNs to identify hosts and NVM subsystems, specifically, in the TLS handshake for a PSK cipher suite:

12. The psk_identity field in the ClientKeyExchange message shall contain the host NQN and the subsystem NQN separated by a space (' '=U+0020h) character as a UTF-8 string, including the terminating null (00h) character.

It was discovered after initial publication that RFC 4297 section 5.3 allows TLS 1.2 PSK implementations to limit the length of the psk_identity they support to 128 bytes and that, as of the publication of this document, at least one popular library, OpenSSL 1.1.1, is limited to 127 byte PSK identities. Since NQNs can be larger than 128 bytes each, following the technique above to generate the psk_identity can result in a psk_identity larger than that supported by a TLS 1.2 PSK implementation. No workaround to this implementation limitation is specified. As such, TLS 1.2 implementation with this issue is discouraged.

The following is an example of the psk_identity field in the ClientKeyExchange message assuming that both the host and the NVM subsystem are using the UUID-based format NVMe Qualified Names:

Modify section 7.4.10.3 as shown below:

Controller PDU Data Alignment (CPDA): Specifies the data alignment for all PDUs ~~transferred from the host to the controller~~ that ~~contain~~ transfer data in addition to the PDU Header (refer to section 7.4.1). This ~~value~~ is a 0's based value in units of dwords in the range 0 to 31 (e.g., values 0, 1, and 2 correspond to 4 byte, 8 byte, and 12 byte alignment).

Technical input submitted to the NVM Express™ Workgroup is subject to the terms of the NVM Express™ Participant's agreement. Copyright © 2014-2021 NVMe™ Corporation.

Modify section 7.4.10.6 as shown below:

7.4.10.6 Command Capsule PDU (CapsuleCmd)

Figure 77: Command Capsule PDU (CapsuleCmd)

<table>
<tr><th>Bytes</th><th>PDU Section</th><th>Description</th></tr>
<tr><td>00</td><td rowspan="5">CH</td><td>PDU-Type: 04h</td></tr>
<tr><td>01</td><td>FLAGS:
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7:2</td><td>Reserved</td></tr>
<tr><td>1</td><td>DDGSTF: If set to ‘1’, then a valid DDGST ~~Data digest~~ value follows the PDU Data.</td></tr>
<tr><td>0</td><td>HDGSTF: If set to ‘1’, then a valid HDGST ~~Header digest~~ value follows the PDU ~~h~~Header.</td></tr>
</table></td></tr>
<tr><td>02</td><td>HLEN: Fixed length of 72 bytes (48h).</td></tr>
<tr><td>03</td><td>PDO: Data ~~O~~offset within PDU. This value shall be a multiple of the data alignment specified by ~~complies to~~ the CPDA field ~~set by the controller~~ in the ICResp PDU (refer to section 7.4.10.3) that was previously sent by the controller on this TCP connection.</td></tr>
<tr><td>07:04</td><td>PLEN: Total length of PDU (including PDU ~~h~~Header, HDGST, PAD, DATA, and DDGST) in bytes.</td></tr>
<tr><td>71:08</td><td>PSH</td><td>NVMe-oF Command Capsule SQE (CCSQE): ~~NVMe-oF~~ Command Capsule SQE.</td></tr>
<tr><td>75:72</td><td>HDGST</td><td>HDGST: If HDGSTF is set in the FLAGS field, this field is present and contains the ~~H~~header digest (refer to section 7.4.6).</td></tr>
<tr><td>N - 1:76</td><td>PAD</td><td>PAD: If in-capsule data is present, the length of this shall be the necessary number of bytes required to achieve the alignment specified by CPDA (refer to section 7.4.10.3).</td></tr>
<tr><td>M - 1:N</td><td>DATA</td><td>NVMe-oF In-Capsule Data (CCICD): This field contains the in-capsule data, if any, of the NVMe-oF Command Capsule.</td></tr>
<tr><td>M + 3:M</td><td>DDGST</td><td>Data Digest (DDGST): If DDGSTF is set in the FLAGS field, and the CCICD field is present, then this field contains the ~~D~~data ~~D~~digest (refer to section 7.4.6) of the CCICD field (in-capsule data).</td></tr>
</table>

Technical input submitted to the NVM Express™ Workgroup is subject to the terms of the NVM Express™ Participant’s agreement. Copyright © 2014-2021 NVMe™ Corporation.

Modify section 7.4.10.8 as shown below:

7.4.10.8 Host To Controller Data Transfer PDU (H2CData)

Figure 79: Host To Controller Data Transfer PDU (H2CData)

<table>
<tr><th>Bytes</th><th>PDU Section</th><th>Description</th></tr>
<tr><td>00</td><td rowspan="8">CH</td><td>PDU-Type: 06h</td></tr>
<tr><td>01</td><td>FLAGS:
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7:3</td><td>Reserved</td></tr>
<tr><td>2</td><td>LAST_PDU: If set to ‘1’, indicates the PDU is the last in the set of H2CData PDUs that correspond to the same R2T PDU.</td></tr>
<tr><td>1</td><td>DDGSTF: If set to ‘1’, then a valid DDGST ~~Data digest~~ value follows the PDU Data.</td></tr>
<tr><td>0</td><td>HDGSTF: If set to ‘1’, then a valid HDGST ~~Header digest~~ value follows the PDU ~~h~~Header.</td></tr>
</table></td></tr>
<tr><td>02</td><td>HLEN: Fixed length of 24 bytes (18h).</td></tr>
<tr><td>03</td><td>PDO: Data ~~O~~offset within PDU. This value shall be a multiple of the data alignment specified by ~~complies to~~ the CPDA field ~~set by the controller~~ in the ICResp PDU (refer to section ~~7.4.9.3~~7.4.10.3) that was previously sent by the controller on this TCP connection.</td></tr>
<tr><td>07:04</td><td>PLEN: Total length of PDU (including PDU ~~h~~Header, HDGST, PAD, DATA, and DDGST) in bytes.</td></tr>
<tr><td>09:08</td><td rowspan="4">PSH</td><td>Command Capsule CID (CCCID): This field contains the SQE.CID value of the Command Capsule PDU associated with the ~~c~~Command ~~d~~Data ~~b~~Buffer.</td></tr>
<tr><td>11:10</td><td>Transfer Tag (TTAG): This field contains the Transfer Tag of the corresponding R2T received by the ~~controller~~ host.</td></tr>
<tr><td>15:12</td><td>Data Offset (DATAO): Byte offset from start of Command ~~d~~Data ~~b~~Buffer. This value shall be a multiple of dwords.</td></tr>
<tr><td>19:16</td><td>Data Length (DATAL): PDU DATA field length in bytes. This value shall be a multiple of dwords.</td></tr>
<tr><td>23:20</td><td>PSH</td><td>Reserved</td></tr>
<tr><td>27:24</td><td>HDGST</td><td>HDGST: If HDGSTF is set in the FLAGS field, this field is present and contains the ~~H~~header digest (refer to section 7.4.6).</td></tr>
<tr><td>N - 1:N</td><td>PAD</td><td>PAD: The length of this field shall be the necessary number of bytes required to achieve the alignment specified by CPDA (refer to section 7.4.10.3).</td></tr>
<tr><td>M - 1:N</td><td>DATA</td><td>PDU-Data</td></tr>
<tr><td>M + 3:M</td><td>DDGST</td><td>Data Digest (DDGST): If DDGSTF is set in the FLAGS field, this field is present and contains the data digest (refer to section 7.4.6).</td></tr>
</table>

Technical input submitted to the NVM Express™ Workgroup is subject to the terms of the NVM Express™ Participant’s agreement. Copyright © 2014-2021 NVMe™ Corporation.

Modify section 7.4.10.9 as shown below:

7.4.10.9 Controller To Host Data Transfer PDU (C2HData)

Figure 80: Controller To Host Data Transfer PDU (C2HData)

<table>
<tr><th>Bytes</th><th>PDU Section</th><th>Description</th></tr>
<tr><td>00</td><td rowspan="6">CH</td><td>PDU-Type: 07h</td></tr>
<tr><td>01</td><td>FLAGS:
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7:4</td><td>Reserved</td></tr>
<tr><td>3</td><td>SUCCESS: If set to ‘1’, indicates that the command referenced by CCCID was completed successfully with no other information and that no ~~r~~Response Capsule PDU is sent by the Controller.</td></tr>
<tr><td>2</td><td>LAST_PDU: If set to ‘1’, indicates the PDU is the last C2HData PDU sent in response to a Command Capsule PDU ~~in the Command Data transfer series~~.</td></tr>
<tr><td>1</td><td>DDGSTF: If set to ‘1’, then a valid DDGST ~~Data digest~~ value follows the PDU Data</td></tr>
<tr><td>0</td><td>HDGSTF: If set to ‘1’, then a valid HDGST ~~Header digest~~ value follows the PDU ~~h~~Header.</td></tr>
</table></td></tr>
<tr><td>02</td><td>HLEN: Fixed length of 24 bytes (18h).</td></tr>
<tr><td>03</td><td>PDO: Data ~~O~~offset within PDU. This value shall be a multiple of the data alignment specified by ~~complies to~~ the HPDA field ~~set by the controller~~ in the ICReq PDU (refer to section ~~7.4.9.2~~7.4.10.2) that was previously sent by the host on this TCP connection.</td></tr>
<tr><td>07:04</td><td>PLEN: Total length of PDU (including PDU ~~h~~Header, HDGST, PAD, DATA, and DDGST) in bytes.</td></tr>
<tr><td>09:08</td><td rowspan="5">PSH</td><td>Command Capsule CID (CCCID): This field contains the SQE.CID value of the Command Capsule PDU associated with the host-resident data.</td></tr>
<tr><td>11:10</td><td>Reserved</td></tr>
<tr><td>15:12</td><td>Data Offset (DATAO): Byte offset from start of host-resident data. This value shall be dword aligned.</td></tr>
<tr><td>19:16</td><td>Data Length (DATAL): PDU DATA field length in bytes. This value shall be dword aligned.</td></tr>
<tr><td>23:20</td><td>Reserved</td></tr>
<tr><td>27:24</td><td>HDGST</td><td>HDGST: If HDGSTF is set in the FLAGS field, this field is present and contains the ~~H~~header digest (refer to section 7.4.6).</td></tr>
<tr><td>N - 1:N</td><td>PAD</td><td>PAD: If HPDA is set to a non-zero value, then this field is padded as specified by HPDA.</td></tr>
<tr><td>M - 1:N</td><td>DATA</td><td>PDU-Data</td></tr>
<tr><td>M + 3:M</td><td>DDGST</td><td>Data Digest (DDGST): Data ~~D~~digest (refer to section 7.4.6) of the PDU-Data field.</td></tr>
</table>

Technical input submitted to the NVM Express™ Workgroup is subject to the terms of the NVM Express™ Participant’s agreement. Copyright © 2014-2021 NVMe™ Corporation.