Errata for the Compute Express Link Specification Revision 3.0

December 13, 2023

LEGAL NOTICE FOR THIS SPECIFICATION FROM COMPUTE EXPRESS LINK CONSORTIUM, INC.

This CXL Errata for the Compute Express Link Specification (this "CXL Specification" or this "document") is owned by and is proprietary to Compute Express Link Consortium, Inc., a Delaware nonprofit corporation (sometimes referred to as "CXL" or the "CXL Consortium" or the "Company") and/or its successors and assigns.

NOTICE TO USERS WHO ARE MEMBERS OF THE CXL CONSORTIUM:

Members of the CXL Consortium (sometimes referred to as a "CXL Member") must be and remain in compliance with all of the following CXL Consortium documents, policies and/or procedures (collectively, the "CXL Governing Documents") in order for such CXL Member's use and/or implementation of this CXL Specification to receive and enjoy all of the rights, benefits, privileges and protections of CXL Consortium membership: (i) CXL Consortium's Intellectual Property Policy; (ii) CXL Consortium's Bylaws; (iii) any and all other CXL Consortium policies and procedures; and (iv) the CXL Member's Participation Agreement.

NOTICE TO NON-MEMBERS OF THE CXL CONSORTIUM, INC.:

If you are not a CXL Member and you have obtained a copy of this CXL Specification, you only have a right to review this document or make reference to or cite this document. Any references or citations to this document must acknowledge the Compute Express Link Consortium, Inc's sole and exclusive copyright ownership of this CXL Specification. The proper copyright citation or reference is as follows: "© 2023 COMPUTE EXPRESS LINK CONSORTIUM, INC. ALL RIGHTS RESERVED." When making any such citation or reference 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 the Compute Express Link Consortium, Inc.

Nothing contained in this CXL Specification shall be deemed as granting (either expressly or impliedly) to any party that is not a CXL Member: (ii) any kind of license to implement or use this CXL Specification or any portion or content described or contained therein, or any kind of license in or to any other intellectual property owned or controlled by the CXL Consortium, including without limitation any trademarks of the CXL Consortium; or (ii) any of the rights, benefits, privileges or protections given to a CXL Member under any CXL Governing Documents. For clarity, and without limiting the foregoing notice in any way, if you are not a CXL Member but still elect to implement this CXL Specification or any portion described herein, you are hereby given notice that your election to do so does not give you any of the rights, benefits, and/or protections of the CXL Members, including without limitation any of the rights, benefits, privileges or protections given to a CXL Member under the CXL Consortium's Intellectual Property Policy.

LEGAL DISCLAIMERS FOR, AND ADDITIONAL NOTICE TO, ALL PARTIES:

THIS DOCUMENT AND ALL SPECIFICATIONS AND/OR OTHER CONTENT PROVIDED HEREIN ARE PROVIDED ON AN "AS IS" BASIS. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, COMPUTE EXPRESS LINK CONSORTIUM, 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 NON-INFRINGEMENT. In the event this CXL Specification makes any references (including without limitation any incorporation by reference) to another standard's setting organization's or any other party's ("Third Party") content or work, including without limitation any specifications or standards of any such Third Party ("Third Party Specification"), you are hereby notified that your use or implementation of any Third Party Specification: (i) is not governed by any of the CXL Governing Documents; (ii) may require your use of a Third Party's patents, copyrights or other intellectual property rights, which in turn may require you to independently obtain a license or other consent from that Third Party in order to have full rights to implement or use that Third Party Specification; and/or (iii) may be governed by the intellectual property policy or other policies or procedures of the Third Party which owns the Third Party Specification. Any trademarks or service marks of any Third Party which may be referenced in this CXL Specification is owned by the respective owner of such marks. The COMPUTE EXPRESS LINK®, CXL® and CXL LOGO trademarks (the "CXL

Contents

Revision History

Revision	Description	Date
1.0	First Release: G1-G2	August 30, 2022
2.0	Second Release: G3-G23; G2 no longer valid, replaced by G12	April 13, 2023
3.0	Third Release: G24-G30; G17 no longer valid, replaced by G26	December 13, 2023

G1 Section 3.3.7 and Section 4.3, BIRsp PBR message requires SPID field

The specification is missing the SPID in the PBR-format version of the BIRsp message, and this field is required to make FAM device memory isolation secure. For G-FAM, the GFD uses the SPID to associate the BIRsp with its outstanding BISnp. For an LD-FAM MLD connected via a PBR Edge Port, the Edge Port uses the SPID to determine the appropriate LD-ID to use in an HBR-format BIRsp message going to the MLD, which may pass through one or more HBR switches below the PBR switch. The security model will be covered in later specification updates.

3.3.7 M2S Back-Invalidate Response (BIRsp)

The Back-Invalidate Response (BIRsp) message class contains response messages from the Master to the Subordinate as a result of Back-Invalidate Snoops. This message class is not supported in 68B Flit mode.

Table 3-37 M2S BIRsp Fields

Field	Width (Bits)			Description
	68B Flit	256B Flit	PBR Flit	
Valid	N/A	1		The valid signal indicates that this is a valid response
Opcode		4		Response type with encodings in Table 3-38
BI-ID		12	0	BI-ID of the device that is the destination of the message. See Section 9.14 for details on how this field is assigned to devices. Not applicable in PBR messages where DPID infers this field.
BITag		12		Tracking ID from the device
LowAddr		2		The lower 2 bits of Cacheline address (Address[7:6]). This is needed to differentiate snoop responses when a Block Snoop is sent and receives snoop response for each cacheline. For block response (opcode names *Blk), this field is Reserved.
DPID		0	12	Destination Port ID
SPID		0	12	Source Port ID
RSVD		9		
Total		**40**	**52~~40~~**	

4.3 CXL.cachemem Link Layer 256B Flit Mode

<Below are various tables and slot formats that need to need to reflect the larger PBR message size>

Table 4-14. 256B G-Slot Formats

Format	SlotFmt Encoding	HBR				PBR	
		Messages	Downstream	Upstream	Length in Bits (Max 124)	Messages	Length in Bits (Max 124)
G0	0000b	H2D REQ + H2D RSP	X		112	H2D Req	92
G1	0001b	3 H2D RSP	X		120	2 H2D RSP	96
G2	0010b	D2H Req + 2 D2H RSP		X	124	D2H REQ	96
G3	0011b	4 D2H RSP		X	96	3 D2H RSP	108
G4	0100b	M2S REQ	X		100	M2S Req	120
G5	0101b	3 M2S BIRsp	X		120	23 M2S BIRsp	10420
G6	0110b	S2M BISnp + S2M NDR		X	124	S2M BISnp	96
G7	0111b	3 S2M NDR		X	120	2 S2M NDR	96
G8	1000b	RSVD				RSVD	
G9	1001b						
G10	1010b						
G11	1011b						
G12	1100b	4 H2D DH	X		112	3 H2D DH	108
G13	1101b	4 D2H DH		X	96	3 D2H DH	108
G14	1110b	M2S RwD	X		104	M2S RwD	124
G15	1111b	3 S2M DRS		X	120	2 S2M DRS	96

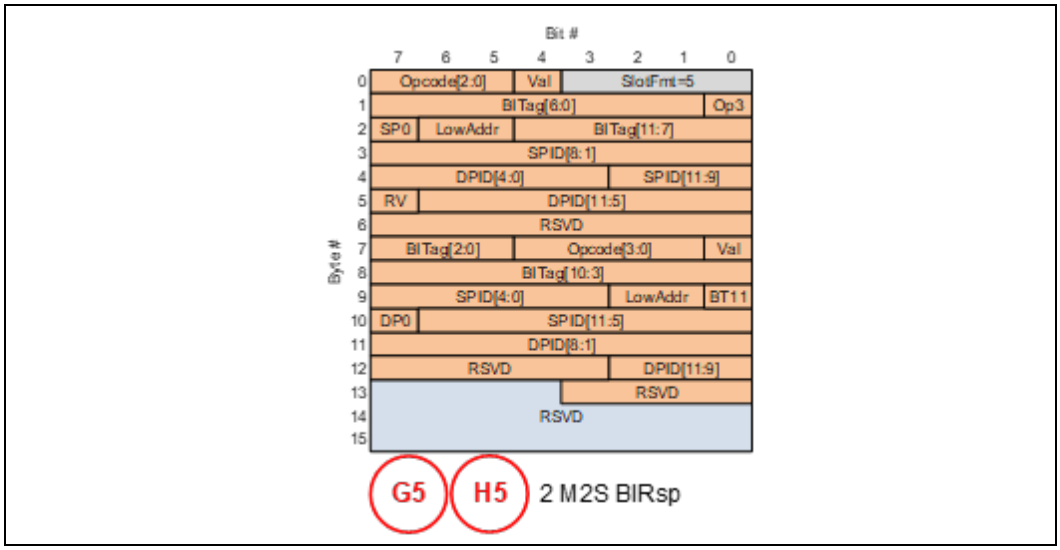
Table 4-15. 256B H-Slot Formats

Format	SlotFmt Encoding	HBR				PBR	
		Messages	Downstream	Upstream	Length in Bits (Max 108)	Messages	Length in Bits (Max 108)
H0	0000b	H2D REQ	X		72	H2D Req	92
H1	0001b	2 H2D RSP	X		80	2 H2D RSP	96
H2	0010b	D2H Req + 1 D2H RSP		X	100	D2H REQ	96
H3	0011b	4 D2H RSP		X	96	3 D2H RSP	108

H4	0100b	M2S REQ	X		100	M2S Req (Zero Extended)	108 (120)
H5	0101b	2 M2S BIRsp	X		80	2 M2S BIRsp	~~80~~104
H6	0110b	S2M BISnp		X	84	S2M BISnp	96
H7	0111b	2 S2M NDR		X	80	2 S2M NDR	96
H8	1000b	LLCTRL				LLCTRL	
H9	1001b	RSVD				RSVD	
H10	1010b						
H11	1011b						
H12	1100b	3 H2D DH	X		84	3 H2D DH	108
H13	1101b	4 D2H DH		X	96	3 D2H DH	108
H14	1110b	M2S RwD	X		104	M2S RwD (Zero Extended)	108 (124)
H15	1111b	2 S2M DRS		X	80	2 S2M DRS	96

<Figure below replaces the CXL3 figure>

Figure 4-57. 256B Packing: G5/H5 PBR Messages



G2 Latency-Optimized Empty Flits Allocate to Tx Retry Buffer

This errata is no longer valid and has been replaced with errata G12.

Currently, CXL.cachemem Empty flits are not allocated to the Tx Retry Buffer. This errata makes a change to allocate Latency-Optimized Empty Flits to the Tx Retry Buffer. Note, Standard Empty Flits are not impacted by this errata.

In Section 6.2.3.11, make the following update:

The 2 bytes of Flit Header as defined in Table 6-5 are transmitted as the first two bytes of the flit. The 2-bit Flit Type field indicates whether the flit carries CXL.io traffic, CXL.cachemem traffic, ALMPs, IDLE flits, Empty flits, or NOP flits. Please refer to Section 6.2.3.1.1.1 for more details. The Prior Flit Type definition is as defined in PCIe Base Specification; it enables the receiver to know that the prior flit was an non-retryable Empty flit, NOP flit, or IDLE flit, and thus does not require replay (i.e., can be discarded) if it has a CRC error. The Type of DLLP Payload definition is as defined in PCIe Base Specification for CXL.io flits; otherwise, this bit is reserved. The Replay Command[1:0] and Flit Sequence Number[9:0] definitions are as defined in PCIe Base Specification.

In Table 6-5, make the following update:

Table 6-5. 256B Flit Header

Flit Header Field	Flit Header Bit Location	Description
Flit Type[1:0]	[7:6]	• 00b = Physical Layer IDLE flit or Physical Layer NOP flit or CXL.io NOP flit • 01b = CXL.io Payload flit • 10b = CXL.cachemem Payload flit or CXL.cachemem generated Empty flit • 11b = ALMP Please refer to Table 6-6 for more details.
Prior Flit Type	[5]	• 0 = Prior flit was an non-retryable Empty, NOP, or IDLE flit (not allocated into Replay buffer) • 1 = Prior flit was a Payload flit or retryable Empty flit (allocated into Replay buffer)
Type of DLLP Payload	[4]	• If (Flit Type = (CXL.io Payload or CXL.io NOP): Use as defined in PCIe Base Specification • If (Flit Type != (CXL.io Payload or CXL.io NOP)): Reserved
Replay Command[1:0]	[3:2]	Same as defined in PCIe Base Specification.
Flit Sequence Number[9:0]	{[1:0], [15:8]}	10-bit Sequence Number as defined in PCIe Base Specification

In section 6.2.3.1.1.1, make the following update:

A Flit Type encoding of 10b indicates either a flit with valid CXL.cachemem Payload flit or a CXL.cachemem Empty flit; this enables CXL.cachemem to minimize idle to valid traffic transitions by arbitrating for use of the ARB/MUX transmit data path even while it does not have valid traffic to send so that it can potentially fill later slots in the flit with late arriving traffic, instead of requiring CXL.cachemem to wait until the next 256-byte

flit boundary to begin transmitting valid traffic. CXL.cachemem Empty flits associated with standard 256B flits are non-retryable and must not be allocated into the transmit retry buffer or receive retry buffer. On the other hand, CXL.cachemem Empty flits associated with latency-optimized 256B flits are retryable and must be allocated into the transmit retry buffer.

Table 6-5. Flit Type[1:0]

Encoding	Flit Payload	Source	Description	Allocated to Retry Buffer?
00b	Physical Layer NOP	Physical Layer	Physical Layer generated (and sunk) flit with no valid payload; inserted in the data stream when its Tx retry buffer is full and it is backpressuring the upper layer or when no other flits from upper layers are available to transmit.	No
	IDLE		Physical Layer generated (and consumed) all 0s payload flit used to facilitate LTSSM transitions as described in PCIe Base Specification	No
	CXL.io NOP	CXL.io Link Layer	Valid CXL.io DLLP payload (no TLP payload); periodically inserted by the CXL.io link layer to satisfy the PCIe Base Specification requirement for a credit update interval if no other CXL.io flits are available to transmit.	No
01b	CXL.io Payload		Valid CXL.io TLP and valid DLLP payload	Yes
10b	CXL.cachemem Payload	CXL.cachemem Link Layer	Valid CXL.cachemem slot and/or CXL.cachemem credit payload	Yes
	CXL.cachemem Empty		No valid CXL.cachemem payload; generated when CXL.cachemem link layer speculatively arbitrates to transfer a flit to reduce idle to valid transition time but no valid CXL.cachemem payload arrives in time to use any slots in the flit.	If non-retryable: No If retryable: Allocate to Tx Retry Buffer only
11b	ALMP	ARB/MUX	ARB/MUX Link Management Packet	Yes

G3 Latency-Optimized Flit Processing When Even CRC Fails on Replayed Flit

This errata corrects Table 6-7 to specify that if the even half of the flit fails CRC checking on the retransmitted flit that an FEC decode and correct operation must be performed. Currently, the specification incorrectly states that if the even half had been previously consumed, that the odd half is permitted to be consumed if it passes CRC. The sequence number resides in the even half and must

be checked in the retransmitted flit, thus the even half of the flit must pass CRC as a prerequisite for consuming any part of the flit.

In Table 6-7, make the following updates:

Original Flit			Post-FEC Corrected Flit			Retransmitted Flit		
Even CRC	Odd CRC	Action	Even CRC	Odd CRC	Subsequent Action	Even CRC	Odd CRC	Subsequent Action
Pass	Pass	Consume Flit	N/A	N/A	N/A	N/A	N/A	N/A
Pass	Fail	Permitted to consume even flit half; perform FEC decode and correct	Pass	Pass	Consume even flit half if not previously consumed (must drop even flit half if previously consumed); Consume odd flit half	N/A	N/A	N/A
			Pass	Fail	Permitted to consume even flit half if not previously consumed; Request Retry	Pass	Pass	Consume even flit half if not previously consumed (must drop even flit half if previously consumed); Consume odd flit half
						Pass	Fail	Permitted to consume even flit half if not previously consumed (must drop even flit half if previously consumed); perform FEC decode and correct
						Fail	Pass /Fail	Perform FEC decode and correct and evaluate next steps
			Fail	Pass /Fail	Request Retry; Log error for even flit half if	Pass	Pass	Consume even flit half if not previously consumed (must drop even flit half if previously consumed); Consume odd flit half
						Pass	Fail	Permitted to consume even flit half if not previously consumed (must drop even flit half if previously consumed); perform FEC decode and correct

					previously consumed[1]	Fail	Pass	~~Consume odd flit half if even flit half was previously consumed; otherwise, p~~Perform FEC decode and correct and evaluate next steps
						Fail	Fail	Perform FEC decode and correct and evaluate next steps
Fail	Pass	Perform FEC decode and correct	Pass	Pass	Consume flit	N/A	N/A	N/A
			Pass	Fail	Permitted to consume even flit half; Request Retry	Pass	Pass	Consume even flit half if not previously consumed (must drop even flit half if previously consumed); Consume odd flit half
						Pass	Fail	Permitted to consume even flit half if not previously consumed; Perform FEC decode and correct and evaluate next steps
						Fail	Pass	~~Consume odd flit half if even flit half was previously consumed; otherwise, p~~Perform FEC decode and correct and evaluate next steps
						Fail	Fail	Perform FEC decode and correct and evaluate next steps
			Fail	Pass /Fail	Request Retry	Pass	Pass	Consume flit
						Pass	Fail	Permitted to consume even flit half; Perform FEC decode and correct and evaluate next steps

[1]The receiver must not consume the FEC-corrected odd flit half that passes CRC because the FEC correction operation is potentially suspect in this particular scenario.

						Fail	Pass /Fail	Perform FEC decode and correct and evaluate next steps
Fail	Fail	Perform FEC decode and correct	Pass	Pass	Consume flit	N/A	N/A	N/A
			Pass	Fail	Permitted to consume even flit half; Request Retry	Pass	Pass	Consume even flit half if not previously consumed (must drop even flit half if previously consumed); Consume odd flit half
						Pass	Fail	Permitted to consume even flit half if not previously consumed; Perform FEC decode and correct and evaluate next steps
						Fail	Pass	~~Consume odd flit half if even half was previously consumed; otherwise, p~~Perform FEC decode and correct and evaluate next steps
						Fail	Fail	Perform FEC decode and correct and evaluate next steps
			Fail	Pass /Fail	Request Retry	Pass	Pass	Consume flit
						Pass	Fail	Permitted to consume even flit half; Perform FEC decode and correct and evaluate next steps
						Fail	Pass /Fail	Perform FEC decode and correct and evaluate next steps

G4 Table 4-19 IDE.TMAC and IDE.MAC messages

Table 4-19 indicates incorrectly that IDE.MAC message requires remaining slots to be Reserved, but the intent is that these message should allow for protocol slots to be sent after in the same flit. These messages are injected during normal high bandwidth traffic and are expected to occur in a flit mixed with other protocol traffic. The description in the Security chapter reflects this expectation.

Table 4-19 also indicates incorrectly that IDE.TMAC messages can have protocol slots after the control messages. This is not allowed as the IDE.TMAC is used to truncate a epoch when there is nothing left to send as described in the Security Chapter.

For clarification the last column will be changed to say "Remaining Slots and CRD field are Reserved?". This aligns with how things were done in 68B control flit/messages and was the intent that was not explicitly stated.

<Including only the portion of Table 4-19 showing IDE message with the fix required. The changes are only the final column in the table inverting the Yes/No in the two rows.>

Table 4-19. 256B Flit Mode Control Message Details

Flit Type	LLCTRL	SubType	SubType Description	Payload	Payload Description	Remaining Slots and CRD field are Reserved?[2]
IDE[3]	0010b	0000b	IDE.Idle	95:0	Payload RSVD Message sent as part of IDE flows to pad sequences with idle flits. Refer to Chapter 11.0 for details on the use of this message.	Yes
		0001b	IDE.Start	95:0	Payload RSVD Message sent to begin flit encryption.	
		0010b	IDE.TMAC	95:0	MAC Field uses all 96 bits of payload. Truncated MAC Message sent to complete a MAC epoch early. Used only when no protocol messages exist to send.	~~No~~Yes
		0011b	IDE.MAC	95:0	MAC Field uses all 96 bits of payload. This encoding is the standard MAC used at the natural end of the MAC epoch and is sent with other protocol slots encoded within the flit.	No

[2]If yes, all the slots in the current flit after this message are reserved, If no, the slots after this may carry protocol messages (header or data).
[3]Supported only in H-slot.

		~~0011b~~	~~IDE.MAC~~	~~95:0~~	~~MAC Field uses all 96 bits of payload.~~ ~~This encoding is the standard MAC used at the natural end of the MAC epoch and is sent with other protocol slots encoded within the flit.~~	Yes
		0100b	IDE.Stop	95:0	Payload RSVD. Message used to disable IDE. Refer to Chapter 11.0 for details on the use of this message.	
		Others	RSVD	95:0	RSVD	

G5 Figure 4-70 and Figure 4-71 Late Viral injection in 256B Flits (Standard and LatOpt)

Both Figures 4-70 and Figures 4-71 should show two back-to-back flits as being corrupted. This is required to ensure that a full retry occurs at the receiver which is described in the text. The diagrams should be corrected to reflect that two flits are corrupted.

Also, Figure 4-71 is incorrectly showing "FlitA-0" and "FlitA-1" re-injected in after Viral, but it should show "FlitB" to match the description.*

<Original Diagrams below>

Figure 4-70. Viral Error Message Injection Standard 256B Flit

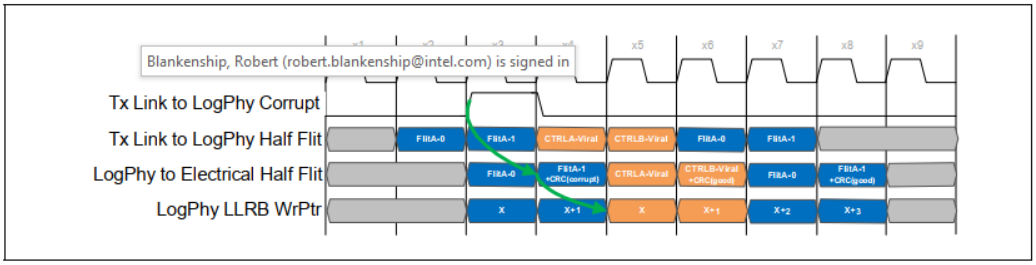
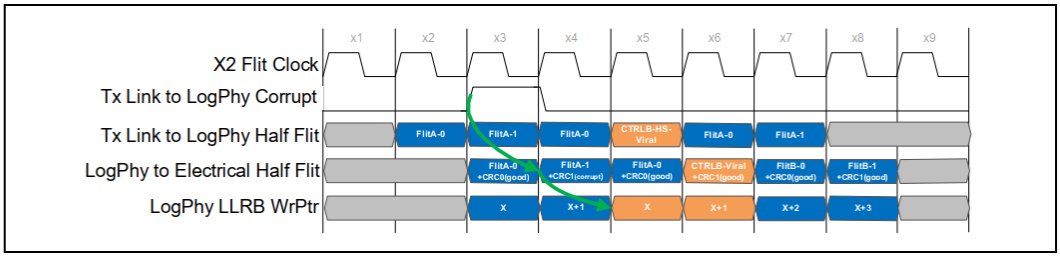


Figure 4-71. Viral Error Message Injection LOpt 256B Flit



<Corrected diagram below>

Figure 4-70. Viral Error Message Injection Standard 256B Flit

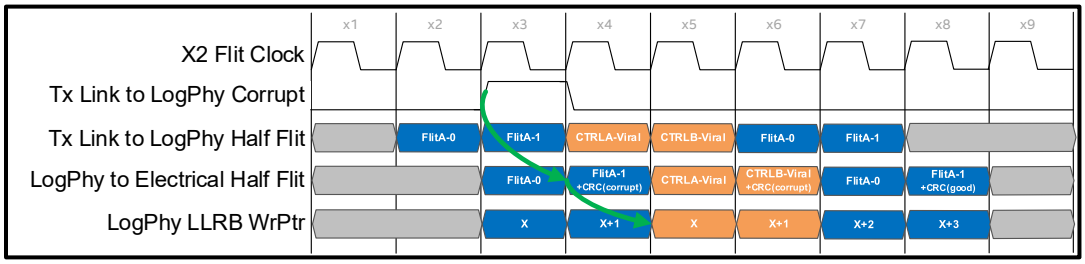
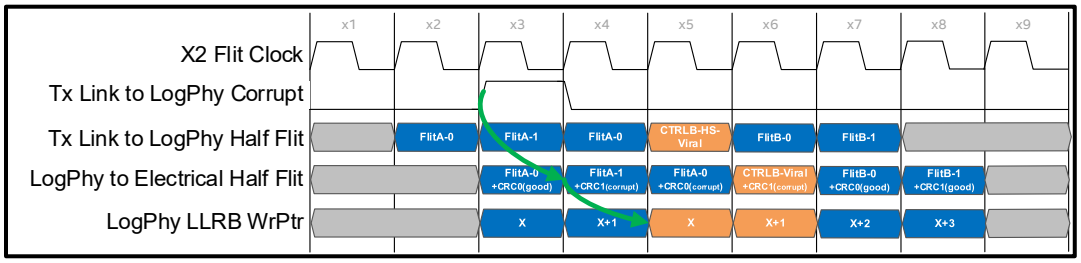


Figure 4-71. Viral Error Message Injection LOpt 256B Flit



G6 CXL Link Capability Version

In Table 8-22 in section 8.2.4, make the following change

Capability	ID	Highest Version	Mandatory	Not Permitted	Optional
..	..	..	..	..	..
CXL Link Capability ..	4	~~2~~3	..	..	..

In section 8.2.4.4, make the following change

Bit Location	Attributes	Description
15:0	..	..
19:16	RO	CXL_Capability_Version: This defines the version number of the CXL_Capability structure present. Version ~~2~~3h represents the structure as defined in this specification.
31:20	..	..

G7 Deprecate the Trust_Level field in the Cache ID Decoder

In section 8.2.4.28.2 CXL Cache ID Decoder Control (Offset 04h)*, make the following changes:*

19:16	RW	**Local Cache ID:** If Assign Cache ID Enable=1, the Port assigns this Cache ID to the directly connected CXL.cache device regardless of whether it is using HDM-D flows or HDM-DB flows. The reset default is 0h.
~~23:20~~	~~RsvdP~~	~~Reserved~~
~~25:24~~	~~RW~~	~~Trust Level: Trust Level assigned to the directly connected Device when Assign Cache ID=1.~~ ~~00b = See Table 8-26.~~ ~~01b = Reserved~~ ~~10b = See Table 8-26.~~ ~~11b = Reserved~~ ~~The reset default is 10b.~~
31:20~~26~~	RsvdP	Reserved

In section 9.15.2, *make the following changes to Table 9-16. Downstream Port Handling of D2H Request Messages:*

Assign Cache ID Value	Forward Cache ID Value	Behavior
0	0	Discard
0	1	Forward upstream. If the message was received over a link operating in 68B Flit Mode, the request is processed as if CacheID field is 0.
1	0	~~If Trust Policy=2, discard the request.~~ ~~If Trust Policy=0, s~~Set CacheID=Local Cache ID and forward upstream. ~~Note that Trust Policy=1 is an invalid setting for a Downstream Port.~~ The link between the device and the Downstream Port may be operating in 68B Flit mode, in which case the D2H request message received by the Downstream Port does not contain the CacheID field.
1	1	Discard (Invalid setting)

In the Implementation Note on page 630, make the following updates

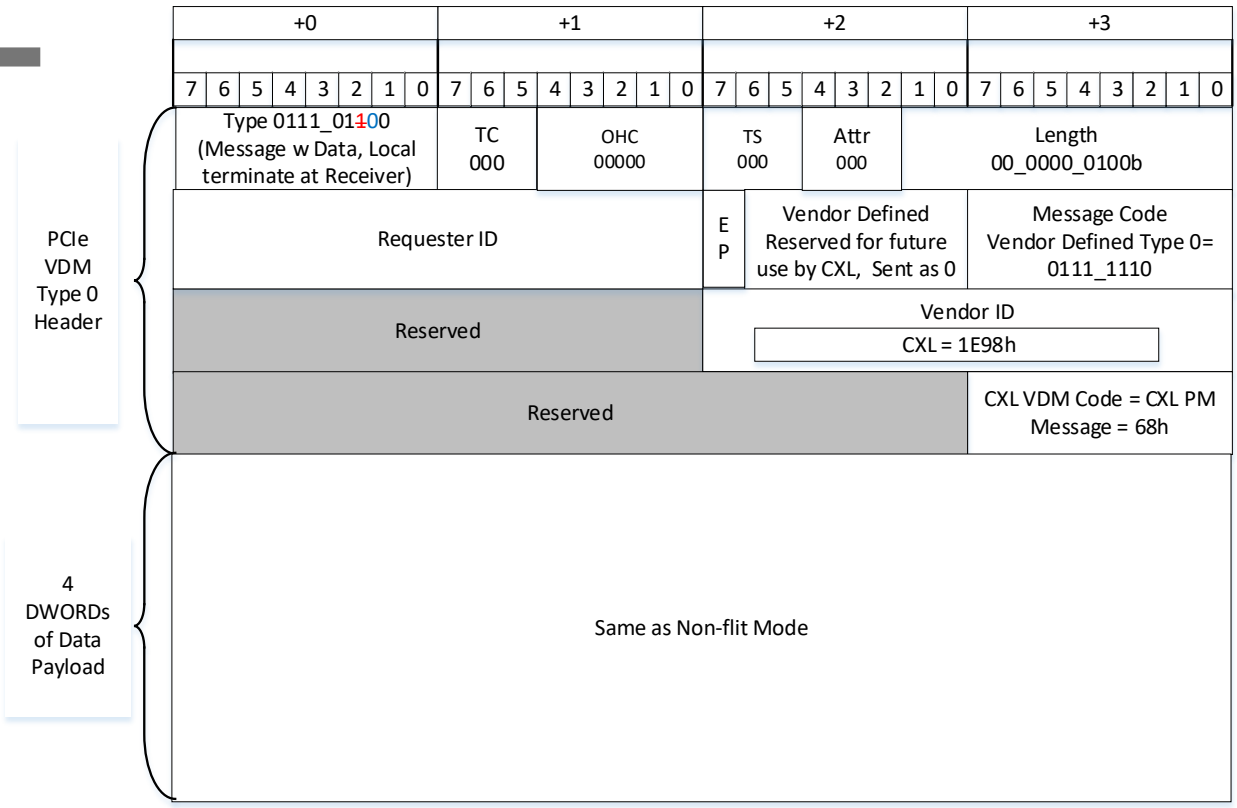
..

3. Configure the DSP of Switch S that is directly connected to Device D to assign Cache ID=c to Device D:
   a. Forward Cache ID=0.
   b. Local Cache ID=c.
   ~~c. Trust Level=0.~~
   ~~d~~c. Assign Cache ID=1.

4. ..

G8 Correct the TLP Type field in PM VDM Header - Flit Mode

In Figure 3-3. CXL Power Management Messages Packet Format - Flit Mode, replace the Type field (Byte 0 in the header) encoding to 0111_0100b from 0111_0110b as shown below



G9 Disambiguate between Message Tag fields

In Section 8.2.9.2.8 Event Notification (Opcode 0106h), make the following changes

..

The FM acknowledges a notification by returning a response. The component shall retransmit the notification every 1 ms using the same Message Tag value in the CCI Message (Figure 7-19) ~~transport header~~ until the FM has returned a response with the 'Success' return code, up to a maximum of 10 retries. No additional Event Notifications shall be sent until the component has received a response from the FM.

G10 Correct Offsets in Identify Output Payload data structure

In Section 8.2.9.1.1 Identify (Opcode 0001h), make the following changes to Table 8-37

Byte Offset	Length in Bytes	Description
..	..	..
08h	8	Device Serial Number: Unique identifier for this device, as defined in the Device Serial Number Extended Capability in PCIe Base Specification
10~~6~~h	1	Maximum Supported Message Size: The maximum supported size of the full message body (as defined in Figure 7-19) in bytes for any requests sent to this component, expressed as 2^n. The minimum supported size is 256 bytes (n=8) and the maximum supported size is 1 MB (n=20). This field is used by the caller to limit the Message Payload size such that the size of the Message Body does not exceed the capabilities of the component. The component shall discard any received messages that exceed the maximum size advertised in this field in a manner that prevents any internal receiver hardware errors. The component shall return a response message with the 'Invalid Payload Length' return code for all received request messages that exceed the maximum size advertised in this field. The CXL specification guarantees that the size of the Identify Output Payload shall never exceed 244 Bytes (256 – 12 Bytes, the combined size of the fields preceding Message Payload).
11~~7~~h	1	Component Type: Indicates the type of component. 00h – Switch. 03h – Type 3 Device. All other encodings are reserved.

G11 Sync Header Bypass Enable Not Applicable at 64 GT/s and Ordered Set Insertion Interval

The Sync Header Bypass optimization is applicable at 8 GT/s, 16 GT/s, and 32 GT/s. The current specification incorrectly states in some places that it is applicable only for 68B Flit mode or not applicable to 256B Flit mode; this errata corrects those inconsistent statements. Additionally, the Ordered Set insertion interval is correctly specified in Table 6-15; however the text in the body of the

spec was inconsistent with the table, so this errata corrects that text. It also updates the compliance chapter to be consistent with this change.

In section 6.4.1 make the following updates:

Upon exit from LTSSM Detect, a Flex Bus link begins training and completes link width negotiation and speed negotiation according to the PCIe LTSSM rules. During link training, the Downstream Port initiates Flex Bus mode negotiation via the PCIe alternate protocol negotiation mechanism. Flex Bus mode negotiation is completed before entering L0 at 2.5 GT/s. If Sync Header bypass is negotiated (applicable only to 8 GT/s, 16 GT/s and 32 GT/s link speeds~~68B Flit mode~~), Sync Headers are bypassed as soon as the link has

In table 6-11, make the following updates:

Bit[10]: **Sync Header Bypass Capable/Enable**	The Downstream Port, Upstream Port, and any retimers advertise their capability in Phase 1; the Downstream Port and Upstream Port advertisethe value as set in the DVSEC Flex Bus Port Control register2. The Downstream Port communicates the results of the negotiation in Phase 2. **Note:** The Retimer must pass this bit unmodified from its Upstream Pseudo Port to its Downstream Pseudo Port. The retimer clears this bit if the retimer does not support this feature when passing from its Downstream Pseudo Port to its Upstream Pseudo Port, but it must never set this bit (only an Upstream Port can set this bit in that direction). If the retimer(s) do not advertise that they are CXL aware, the Downstream Port assumes that this feature is not supported by the Retimer(s) regardless of how this bit is set. **Note:** ~~This bit is not applicable when 256B Flit mode is negotiated and must therefore be ignored in that case.~~ This bit is only applicable at 8 GT/s, 16 GT/s, and 32 GT/s link speeds.

In Section 6.8, make the following updates:

The Sync Header Bypass optimization applies only at 8 GT/s, 16 GT/s, and 32 GT/s link speeds. At 64 GT/s link speeds, 1b/1b encoding is used as specified in PCIe Base Specification; thus, the Sync Header Bypass optimization is not applicable. If PCIe Flit Mode is not enabled and the Sync Header Bypass optimization is enabled, then the CXL specification dictates insertion of Ordered Sets at a fixed interval. If PCIe Flit Mode is enabled or Sync Header Bypass is not enabled, the Ordered Set insertion rate follows the PCIe Base Specification.

In Section 8.2.1.3.2, make the following update to the Flex Bus Port Control register:

3	HwInit	**CXL_Sync_Hdr_Bypass_Enable:** When set, enables bypass of the 2-bit sync header by the Flex Bus physical layer when operating in Flex Bus.CXL mode. This is a performance optimization. ~~This bit is reserved for 256B Flit mode.~~

In Section 8.2.1.3.3, make the following update to the Flex Bus Port Status register:

3	RO	**CXL_Sync_Hdr_Bypass_Enabled:** When set, indicates that bypass of the 2-bit sync header by the Flex Bus physical layer has been enabled when operating in Flex Bus.CXL mode as a result of PCIe alternate protocol negotiation for Flex Bus. ~~This bit is reserved for 256B Flit mode.~~

In Chapter 14 "CXL Compliance Testing", move 14.6.1.9 to 14.6.13.

G12 Empty Flits Allocate to Tx Retry Buffer

This errata replaces Errata G2. Errata G2 introduced changes to allocate CXL.cachemem Latency-Optimized Empty Flits to the Tx Retry Buffer. This errata allocates all CXL.cachemem Empty Flits to the Tx Retry Buffer.

In Section 6.2.3.1. 1, make the following update:

The 2 bytes of Flit Header as defined in Table 6-5 are transmitted as the first two bytes of the flit. The 2-bit Flit Type field indicates whether the flit carries CXL.io traffic, CXL.cachemem traffic, ALMPs, IDLE flits, Empty flits, or NOP flits. Please refer to Section 6.2.3.1.1.1 for more details. The Prior Flit Type definition is as defined in PCIe Base Specification; it enables the receiver to know that the prior flit was ~~an Empty flit,~~ a NOP flit~~,~~ or IDLE flit, and thus does not require replay (i.e., can be discarded) if it has a CRC error. The Type of DLLP Payload definition is as defined in PCIe Base Specification for CXL.io flits; otherwise, this bit is reserved. The Replay Command[1:0] and Flit Sequence Number[9:0] definitions are as defined in PCIe Base Specification.

In Table 6-5, make the following update:

Table 6-5. 256B Flit Header

Flit Header Field	Flit Header Bit Location	Description
Flit Type[1:0]	[7:6]	• 00b = Physical Layer IDLE flit or Physical Layer NOP flit or CXL.io NOP flit • 01b = CXL.io Payload flit • 10b = CXL.cachemem Payload flit or CXL.cachemem generated Empty flit • 11b = ALMP Please refer to Table 6-6 for more details.
Prior Flit Type	[5]	• 0 = Prior flit was a~~n Empty,~~ NOP~~,~~ or IDLE flit (not allocated into Replay buffer) • 1 = Prior flit was a Payload flit or Empty flit (allocated into Replay buffer)
Type of DLLP Payload	[4]	• If (Flit Type = (CXL.io Payload or CXL.io NOP): Use as defined in PCIe Base Specification • If (Flit Type != (CXL.io Payload or CXL.io NOP)): Reserved

Replay Command[1:0]	[3:2]	Same as defined in PCIe Base Specification.
Flit Sequence Number[9:0]	{[1:0], [15:8]}	10-bit Sequence Number as defined in PCIe Base Specification

In section 6.2.3.1.1.1, make the following update:

A Flit Type encoding of 10b indicates either a flit with valid CXL.cachemem Payload flit or a CXL.cachemem Empty flit; this enables CXL.cachemem to minimize idle to valid traffic transitions by arbitrating for use of the ARB/MUX transmit data path even while it does not have valid traffic to send so that it can potentially fill later slots in the flit with late arriving traffic, instead of requiring CXL.cachemem to wait until the next 256-byte flit boundary to begin transmitting valid traffic. ~~CXL.cachemem Empty flits are non-retryable and must not be allocated into the transmit retry buffer or receive retry buffer.~~ CXL.cachemem Empty flits are retryable and must be allocated into the transmit retry buffer.

Table 6-5. Flit Type[1:0]

Encoding	Flit Payload	Source	Description	Allocated to Retry Buffer?
00b	Physical Layer NOP	Physical Layer	Physical Layer generated (and sunk) flit with no valid payload; inserted in the data stream when its Tx retry buffer is full and it is backpressuring the upper layer or when no other flits from upper layers are available to transmit.	No
	IDLE		Physical Layer generated (and consumed) all 0s payload flit used to facilitate LTSSM transitions as described in PCIe Base Specification	No
	CXL.io NOP	CXL.io Link Layer	Valid CXL.io DLLP payload (no TLP payload); periodically inserted by the CXL.io link layer to satisfy the PCIe Base Specification requirement for a credit update interval if no other CXL.io flits are available to transmit.	No
01b	CXL.io Payload		Valid CXL.io TLP and valid DLLP payload	Yes
10b	CXL.cachemem Payload	CXL.cachemem Link Layer	Valid CXL.cachemem slot and/or CXL.cachemem credit payload	Yes
	CXL.cachemem Empty		No valid CXL.cachemem payload; generated when CXL.cachemem link layer speculatively arbitrates to transfer a flit to reduce idle to valid transition time but no valid CXL.cachemem payload arrives in time to use any slots in the flit.	~~No~~ Yes, allocate to Tx Retry Buffer only
11b	ALMP	ARB/MUX	ARB/MUX Link Management Packet	Yes

G13 CXL.cachemem Retry in 68B Flit mode corrections

In Section 4.2.8.5.1, make the following updates:

4.2.8.5.1 Local Retry State Machine (LRSM)

This state machine is activated at the entity that detects an error on a received flit. The possible states for this state machine are:

- RETRY_LOCAL_NORMAL: This is the initial or default state indicating normal operation (no CRC error has been detected).
- RETRY_LLRREQ: This state indicates that the receiver has detected an error on a received flit and a RETRY.Req sequence must be sent to the remote entity.
- RETRY_LOCAL_IDLE: This state indicates that the receiver is waiting for a RETRY.Ack sequence from the remote entity in response to its RETRY.Req sequence. The implementation may require substates of RETRY_LOCAL_IDLE to capture, for example, the case where the last flit received is a Frame flit and the next flit expected is a RETRY.Ack.
- RETRY_PHY_REINIT: The state machine remains in this state for the duration of ~~a~~ the virtual Link State Machine (vLSM) being in ~~physical layer~~ retrain.
- RETRY_ABORT: This state indicates that the retry attempt has failed and the link cannot recover. Error logging and reporting in this case is device specific. This is a terminal state.

The local retry state machine also has the three counters described below. The counters and thresholds described below are implementation specific.

- **TIMEOUT**: This counter is enabled whenever a RETRY.Req request is sent from an entity and the LRSM state becomes RETRY_LOCAL_IDLE. The TIMEOUT counter is disabled and the counting stops when the LRSM state changes to some state other than RETRY_LOCAL_IDLE. The TIMEOUT counter is reset to 0 at link layer initialization and whenever the LRSM state changes from RETRY_LOCAL_IDLE to RETRY_LOCAL_NORMAL or RETRY_LLRREQ. The TIMEOUT counter is also reset when the vLSM transitions from Retrain to Active ~~Physical layer returns from re-initialization~~ (the LRSM transition through RETRY_PHY_REINIT to RETRY_LLRREQ). If the counter has reached its threshold without receiving a RETRY.Ack sequence, then the RETRY.Req request is sent again to retry the same flit. See Section 4.2.8.5.2 for a description of when TIMEOUT increments.

Note: It is suggested that the value of TIMEOUT should be no less than 4096 transfers.

- **NUM_RETRY**: This counter is used to count the number of RETRY.Req requests sent to retry the same flit. The counter remains enabled during the whole retry sequence (state is not RETRY_LOCAL_NORMAL). It is reset to 0 at initialization. It is also reset to 0 when a RETRY.Ack sequence is received with the Empty bit set or whenever the LRSM state is RETRY_LOCAL_NORMAL and an error-free retryable flit is received. The counter is incremented whenever the LRSM state changes from RETRY_LLRREQ to RETRY_LOCAL_IDLE. If the counter reaches a threshold (called MAX_NUM_RETRY), then the local retry state machine transitions to the RETRY_PHY_REINIT. The NUM_RETRY counter is also reset when the vLSM transitions from Retrain to Active ~~Physical layer exits from LTSSM recovery state~~ (the LRSM transition through RETRY_PHY_REINIT to RETRY_LLRREQ).

Note: It is suggested that the value of MAX_NUM_RETRY should be no less than Ah.

- **NUM_PHY_REINIT**: This counter is used to count the number of transitions to RETRY_PHY_REINIT~~physical layer re-initializations~~ generated during an LLR sequence due to the number of retries exceeding MAX_NUM_RETRY. The counter remains enabled during the whole retry sequence (state is not RETRY_LOCAL_NORMAL). It is reset to 0 at initialization and after successful completion of the retry sequence. The counter is incremented whenever the LRSM changes from RETRY_LLRREQ to RETRY_PHY_REINIT due to the number of retries exceeding MAX_NUM_RETRY. If the counter reaches a threshold (called MAX_NUM_PHY_REINIT) instead of transitioning from RETRY_LLRREQ to RETRY_PHY_REINIT, the LRSM will transition to RETRY_ABORT. The NUM_PHY_REINIT counter is also reset whenever a RETRY.Ack sequence is received with the Empty bit set.

Note: It is suggested that the value of MAX_NUM_PHY_REINIT should be no less than Ah.

Note that the condition of TIMEOUT reaching its threshold is not mutually exclusive with other conditions that cause the LRSM state transitions. RETRY.Ack sequences can be assumed to never arrive at the time that the retry requesting device times out and sends a new RETRY.Req sequence (by appropriately setting the value of TIMEOUT – see <Link>Section 0.0.0.0.1). If this case occurs, no guarantees are made regarding the behavior of the device (behavior is "undefined" from a Spec perspective and is not validated from an implementation perspective). Consequently, the LLR Timeout value should not be reduced unless it can be certain this case will not occur. If an error is detected at the same time as TIMEOUT reaches its threshold, then the error on the received flit is ignored, TIMEOUT is taken, and a repeat RETRY.Req sequence is sent to the remote entity.

Table 4-12 Local Retry State Transitions

Current Local Retry State	Condition	Next Local Retry State	Actions
RETRY_LOCAL_NORMAL	An error free retryable flit is received.	RETRY_LOCAL_NORMAL	Increment NumFreeBuf using the amount specified in the ACK or Full_Ack fields. Increment NumAck by 1. Increment Eseq by 1. NUM_RETRY is reset to 0. NUM_PHY_REINIT is reset to 0. Received flit is processed normally by the link layer.
RETRY_LOCAL_NORMAL	Error free non-retryable flit (other than RETRY.Req sequence) is received.	RETRY_LOCAL_NORMAL	Received flit is processed.
RETRY_LOCAL_NORMAL	Error free RETRY.Req sequence is received.	RETRY_LOCAL_NORMAL	RRSM is updated.
RETRY_LOCAL_NORMAL	Error is detected on a received flit.	RETRY_LLRREQ	Received flit is discarded.
RETRY_LOCAL_NORMAL	PHY_RESET[1] / PHY_REINIT[2] is detected.	RETRY_PHY_REINIT	None.
RETRY_LLRREQ	NUM_RETRY == MAX_NUM_RETRY and NUM_PHY_REINIT == MAX_NUM_PHY_REINIT	RETRY_ABORT	Indicate link failure.

RETRY_LLRREQ	NUM_RETRY == MAX_NUM_RETRY and NUM_PHY_REINIT < MAX_NUM_PHY_REINIT	RETRY_PHY_REINIT	If an error-free RETRY.Req or RETRY.Ack sequence is received, process the flit. Any other flit is discarded. RetrainRequest is sent to physical layer. Increment NUM_PHY_REINIT.
RETRY_LLRREQ	NUM_RETRY < MAX_NUM_RETRY and a RETRY.Req sequence has not been sent.	RETRY_LLRREQ	If an error-free RETRY.Req or RETRY.Ack sequence is received, process the flit. Any other flit is discarded.
RETRY_LLRREQ	NUM_RETRY < MAX_NUM_RETRY and a RETRY.Req sequence has been sent.	RETRY_LOCAL_IDLE	If an error free RETRY.Req or RETRY.Ack sequence is received, process the flit. Any other flit is discarded. Increment NUM_RETRY.
RETRY_LLRREQ	PHY_RESET[1] / PHY_REINIT[2] is detected.	RETRY_PHY_REINIT	None.
RETRY_LLRREQ	Error is detected on a received flit	RETRY_LLRREQ	Received flit is discarded.
RETRY_PHY_REINIT	Physical layer is still in reinit.	RETRY_PHY_REINIT	None.
RETRY_PHY_REINIT	Physical layer returns from Reinit.	RETRY_LLRREQ	Received flit is discarded. NUM_RETRY is reset to 0.
RETRY_LOCAL_IDLE	RETRY.Ack sequence is received and NUM_RETRY from RETRY.Ack matches the value of the last RETRY.Req sent by the local entity.	RETRY_LOCAL_NORMAL	TIMEOUT is reset to 0. If RETRY.Ack sequence is received with Empty bit set, NUM_RETRY is reset to 0 and NUM_PHY_REINIT is reset to 0.
RETRY_LOCAL_IDLE	RETRY.Ack sequence is received and NUM_RETRY from RETRY.Ack does NOT match the value of the last RETRY.Req sent by the local entity.	RETRY_LOCAL_IDLE	Any received retryable flit is discarded.
RETRY_LOCAL_IDLE	TIMEOUT has reached its threshold.	RETRY_LLRREQ	TIMEOUT is reset to 0.
RETRY_LOCAL_IDLE	Error is detected on a received flit.	RETRY_LOCAL_IDLE	Any received retryable flit is discarded.
RETRY_LOCAL_IDLE	A flit other than RETRY.Ack/RETRY.Req sequence is received.	RETRY_LOCAL_IDLE	Any received retryable flit is discarded.
RETRY_LOCAL_IDLE	A RETRY.Req sequence is received.	RETRY_LOCAL_IDLE	RRSM is updated.
RETRY_LOCAL_IDLE	PHY_RESET[1] / PHY_REINIT[2] is detected.	RETRY_PHY_REINIT	None.
RETRY_ABORT	A flit is received.	RETRY_ABORT	All received flits are discarded.

1. PHY_RESET is the condition of vLSM ~~Physical Layer~~ telling the Link Layer it needs to initiate a Link Layer Retry due to exit from Retrain ~~LTSSM Recovery~~ state.
2. PHY_REINIT is the condition of the Link Layer instructing the Phy to retrain

In Section 4.2.8.5.3:

4.2.8.5.3 Remote Retry State Machine (RRSM)

The remote retry state machine is activated at an entity if a flit sent from that entity is received in error by the local receiver, resulting in a link layer retry request (RETRY.Req sequence) from the remote entity. The possible states for this state machine are:

- RETRY_REMOTE_NORMAL: This is the initial or default state indicating normal operation.
- RETRY_LLRACK: This state indicates that a link layer retry request (RETRY.Req sequence) has been received from the remote entity and a RETRY.Ack sequence followed by flits from the retry queue must be (re)sent.

The remote retry state machine transitions are described in the table below.

Table 4-13 Remote Retry State Transition

Current Remote Retry State	Condition	Next Remote Retry State
RETRY_REMOTE_NORMAL	Any flit, other than error free RETRY.Req sequence, is received.	RETRY_REMOTE_NORMAL
RETRY_REMOTE_NORMAL	Error free RETRY.Req sequence is received.	RETRY_LLRACK
RETRY_LLRACK	RETRY.Ack sequence is not sent.	RETRY_LLRACK
RETRY_LLRACK	RETRY.Ack sequence is sent.	RETRY_REMOTE_NORMAL
RETRY_LLRACK	vLSM in Retrain state~~Physical Layer Reinitialization~~.	RETRY_REMOTE_NORMAL

In Section 4.2.8.6:

4.2.8.6 Interaction with vLSM Retrain State~~Physical Layer Reinitialization~~

On ~~detection of a physical layer LTSSM Recovery~~ detection by the Link Layer of the vLSM transition from Active to Retrain state, the receiver side of the link layer must force a link layer retry on the next flit. Forcing an error will either initiate LLR or cause a current LLR to follow the correct error path. The LLR will ensure that no retryable flits are dropped during the physical layer reinit. Without initiating an LLR it is possible that packets/flits in flight on the physical wires could be lost or the sequence numbers could get mismatched.

Upon detection of a vLSM transition to Retrain~~physical layer LTSSM Recovery~~, the LLR RRSM needs to be reset to its initial state and any instance of RETRY.Ack sequence needs to be cleared in the link layer and physical layer. The device needs to ensure that it receives a RETRY.Req sequence before it transmits a RETRY.Ack sequence.

G14 Clarifications from PCIe L0p errata

PCIe introduced errata for L0p section adding some rules for DLLP handling and "abandoning" the request. This errata covers clarifications to avoid ambiguity between which rules are applicable and which are not in CXL L0p negotiation.

In Section 5.1.2.5:

5.1.2.5 L0p Support

256B Flit mode supports L0p as defined in PCIe Base Specification; however, instead of using Link Management DLLPs, the ARB/MUX ALMPs are used to negotiate the L0p width with the Link partner. PCIe rules related to DLLP transmission, corruption and consequent abandonment of L0p handshakes do not apply to CXL. This section defines the additional rules that are required when ALMPs are used for negotiation of L0p width.

When L0p is enabled, the ARB/MUX must aggregate the requested link width indications from the CXL.io and CXL.cachemem Link Layers to determine the L0p width for the physical link. The Link Layers must also indicate to the ARB/MUX whether the L0p request is a priority request (e.g., such as in the case of thermal throttling). The aggregated width must be greater than or equal to the larger link width that is requested by the Link Layers if it is not a priority request. The aggregated width can be greater if the ARB/MUX decides that the two protocol layers combined require a larger width than the width requested by each protocol layer. For example, if CXL.io is requesting a width of x2, and CXL.cachemem is requesting a width of x2, the ARB/MUX is permitted to request and negotiate x4 with the remote Link partner. The specific algorithm for aggregation is implementation specific.

In the case of a priority request from either Link Layer, the aggregated width is the lowest link width that is priority requested by the Link Layers. The ARB/MUX uses L0p ALMP handshakes to negotiate the L0p link width changes with its Link partner.

The following sequence is followed for L0p width changes:

1. Each Link Layer indicates its minimum required link width to the ARB/MUX. It also indicates whether the request is a priority request.
2. If the ARB/MUX determines that the aggregated L0p width is different from the current width of the physical link, the ARB/MUX must initiate an L0p width change request to the remote ARB/MUX using the L0p request ALMP. It also indicates whether the request is a priority request in the ALMP.
3. The ARB/MUX must ensure that there is only one outstanding L0p request at a time to the remote Link partner.
4. The ARB/MUX must respond with an L0p ACK or an L0p NAK to any outstanding L0p request ALMP within 1 microsecond. (The time is counted only during the L0 state of the physical LTSSM. Time is measured from the receipt of the request ALMP from the Physical Layer to the scheduling of the response ALMP from the ARB/MUX to the Physical Layer. The time does not include the time spent by the ALMPs in the RX or TX Retry buffers.)
5. Whether to send an L0p ACK or an L0p NAK response must be determined using the L0p resolution rules from PCIe Base Specification.
6. If PMTimeout (see Section 8.2.5.1) is enabled and a response is not received for an L0p Request ALMP within the programmed time window, the ARB/MUX must treat this as an uncorrectable internal error and escalate accordingly.
7. Once the L0p ALMP handshake is complete, the ARB/MUX must direct the Physical Layer to take the necessary steps for downsizing or upsizing the link, as follows:

a. Downsizing: If the ARB/MUX receives an L0p ACK in response to its L0p request to downsize, the ARB/MUX notifies the Physical Layer to start the flow for transitioning to the corresponding L0p width at the earliest opportunity. If the ARB/MUX sends an L0p ACK in response to an L0p

request, the ARB/MUX notifies the Physical Layer to participate in the flow for transitioning to the corresponding L0p width once it has been initiated by the remote partner. After a successful L0p width change, the corresponding width must be reflected back to the Link Layers.

b. Upsizing: If the ARB/MUX receives an L0p ACK in response to its L0p request to upsize, the ARB/MUX notifies the Physical Layer to immediately begin the upsizing process. If the ARB/MUX sends an L0p ACK in response to an L0p request, the ARB/MUX notifies the Physical Layer of the new width and an indication to wait for upsizing process from the remote Link partner. After a successful L0p width change, the corresponding width must be reflected back to the Link Layers.

If the Link has not reached the negotiated L0p width 24ms after the L0p ACK was sent or received, the ARB/MUX must trigger the Physical Layer to transition the LTSSM to Recovery.

The L0p ALMP handshakes can happen concurrently with vLSM ALMP handshakes. L0p width changes do not affect vLSM states.

In 256B Flit mode, the PCIe-defined PM and Link Management DLLPs are not applicable for CXL.io and must not be used.

Similar to PCIe, the Physical Layer's entry to Recovery or link down conditions restores the link to its maximum configured width and any Physical Layer states related to L0p are reset as if no width change request was made. The ARB/MUX must finish any outstanding L0p handshakes before requesting the Physical Layer to enter a PM state. If the ARB/MUX is waiting for an L0p ACK or NAK from the remote ARB/MUX when the link enters Recovery, after exit from Recovery, the ARB/MUX must continue to wait for the L0p response, discard that response, and then, if desired, reinitiate the L0p handshake.

G15 CXL Viral Handling

Update section 12.4 to remove timing relationship between reporting an error through AER and generating a Viral indication.

12.4 CXL Viral Handling

CXL links and CXL devices are expected to be Viral compliant. Viral is an errorcontainment mechanism. A platform must choose to enable Viral at boot. The Host implementation of Viral allows the platform to enable the Viral feature by writing into a register. Similarly, a BIOS-accessible control register on the device is written to enable Viral behavior (both receiving and sending) on the device. Viral support capability and control for enabling are reflected in the DVSEC.

When enabled, a Viral indication is generated whenever an Uncorrected_Fatal error is detected. Viral is not a replacement for existing error-reporting mechanisms. Instead, its purpose is an additional error-containment mechanism. The detector of the error is responsible for reporting the error through AER and ~~then~~ generating a Viral indication. Any entity that is capable of reporting Uncorrected_Fatal errors must also be capable of generating a Viral indication.

G16 H2D Req Targeting Local Memory of Type 2 Devices

Section 3 is missing the description of the case where a Type 2 device receives a CXL.cache H2D Req message on an address which belongs to its local memory, and the associated requirements for proper

behavior when this situation happens. This issue affects both HDM-D and HDM-DB Type 2 devices, i.e. also affects CXL1.1 and CXL2.

Adding a new subsection 3.2.5.16 dedicated to this situation, with a brief description and associated requirements.

3.2.5.16 H2D Req targeting Device-attached Memory

H2D Req messages are sent by a host to a device because the host believes that the device may own a cacheline that the device previously got through this same host. The very principle of a Type 2 device is to provide direct access to Device-attached Memory, i.e. without going through its host. Host coherence for this region is managed using M2S Req channel. These statements combined could lead a Type 2 device to assume that H2D Req messages can never target addresses belonging to the Device-attached memory by design.

However, a host may decide to snoop more cache peers than strictly required, without any other consideration than the cache peer being visible to the host. This type of behavior is allowed by the CXL protocol and can happen for multiple reasons, including coarse tracking and proprietary RAS features. In that context, a host may generate H2D Req to a Type 2 device on addresses that belong to the Device-attached Memory. H2D Req from the host targeting Device-attached memory can cause coherency issues if the device were to respond with data and, more generally speaking, protocol corner cases.

To avoid these issues, both HDM-D and HDM-DB Type 2 devices are required to :

- Detect H2D Req targeting Device-attached Memory.
- When detected, respond with RspIHitI unconditionally, disregarding all internal states and without changing any internal state (e.g. don't touch the cache).

G17 Buried State on Memory Protocol (replaced by G24)

This errata is no longer valid as it is replaced by G26. The original text of the errata is preserved below but highlighted in dark grey.

The buried cache rules added in CXL3.0 were found to be overly restrictive and need to be relaxed to allow for hosts to resolve conflicts and without creating a very sub-optimial caching in a host for HDM-D/DB. The updated rules in section 3.3.11.1 are captured in this errata. Example flows will be added into future specifications to show problematic cases that drove the changes.

3.3.11.1 Buried Cache State Rules for HDM-D/HDM-DB

Buried Cache state for CXL.mem protocol refers to the state of the cacheline registered by the host's Home Agent logic (HA) for a cacheline address when a new Req or RwD message is being sent. This cache state could be a cache that is controlled by the host, but does not cover the cache in the device that is the owner of the HDM-D/HDM-DB memory. These rules are applicable to only HDM-D/HDM-DB memory where the device is managing coherence.

Buried Cache state rules for host-issued CXL.mem Req/RwD messages:

- Must not issue a MemRd (MetaValue=I)~~/MemRdData~~ if the cacheline is buried in Modified, Exclusive, or Shared state.
- May not issue a MemRd (MetaValue=S) or MemRdData if the cacheline is buried in Modified or Exclusive, but is allowed to issue when the host has Shared or Invalid.

- May issue a MemRd (MetaValue = A) from any state.
- May issue a MemRd (MetaField = NoOp) from any state. Note that the final host cache state may result in a downgraded state such as Invalid when initial buried state exists and conflicting BISnp result in the buried state being downgraded.
- Must not issue MemInv/MemInvNT if the cacheline is buried in Modified or Exclusive state. The Device may request ownership in Exclusive state as an upgrade request from Shared state.
- May issue MemClnEvct from Shared or Exclusive state.
- May issue MemWr with SnpType=SnpInv only from I-state. This use of this encoding is not allow for HDM-DB memory regions where coherence extends to multiple hosts (e.g. Coherent Shared FAM as described in Section 2.4.4).
- MemWr with SnpType=NoOp may only be issued from Modified state.

~~**Error! Reference source not found.**~~Table 3-47 summarizes the Req message and RwD message allowance for Buried Cache state. MemRdFwd/MemWrFwd/BIConflict are excluded from this table because they are response messages.

G18 Clarify Uncorrectable Error Severity Control

Update section 8.2.4.16.3 Uncorrectable Error Severity Register as follows

The Uncorrectable Error Severity register controls whether an individual error is ~~reported~~considered ~~as~~ a Non-fatal or Fatal error. An error is ~~reported~~considered ~~as~~ fatal uncorrectable when the corresponding error bit in the severity register is Set. If an error is considered fatal and viral is enabled, a Viral indication shall be generated (see Section 12.4). If the bit is Cleared, the corresponding error is ~~reported~~considered ~~as~~ non-fatal uncorrectable error and shall not trigger Viral indication. This register does not control whether an error is signaled as ERR_FATAL or ERR_NONFATAL over CXL.io.

G19 Clarify HDM Decoder Functionality

Update section *8.2.4.19.1 CXL HDM Decoder Capability Register (Offset 00h) as follows*

Bit Location	Attributes	Description
..	..	..

22:21	HwInit	**Supported Coherency Models**: Indicates the coherency models that are supported by a CXL.mem device. This field is reserved for all other components[2]. 00b - Unknown 01b - Device Coherent. The Target Range Type bit in an HDM decoder must be 0 when the HDM decoder is committed. Otherwise, the device behavior is undefined. 10b - Host-Only. The Target Range Type bit in an HDM decoder must be 1 when the HDM decoder is committed. Otherwise, the device behavior is undefined. 11b - Host-Only or Device Coherent. The Target Range Type bit in an HDM decoder is RW and may be set to either 0 or 1 by software before committing the HDM decoder.
31:231~~1~~	RsvdP	**Reserved**

Update section *8.2.4.19.7 CXL HDM Decoder n Control Register (Offset 20h*n+20h) as follows*

Bit Location	Attributes	Description
..	..	..
9	RWL	**Commit** - Software sets this to 1 to commit Decoder n. The locking behavior is described in Section 8.2.4.20.13. Default value of this bit is 0. A 1 to 0 transition of this bit shall cause the associated Committed bit to transition from 1 to 0.
..	..	..
12	RWL / RO	**Target Range Type**: Formerly known as Target Device Type. This bit is RWL for ~~BI-capable CXL.mem devices,~~ CXL Host Bridges, and Upstream Switch Ports. This bit is permitted to be RO for devices that do not support this reconfigurability~~other than BI-capable Type 3 devices~~ and it may return the value of 0 or 1 to represent the only coherency model they support. 0: Target is a Device Coherent Address range (HDM-D or HDM-DB) 1: Target is a Host-Only Coherent Address range (HDM-H). The locking behavior is described in Section 8.2.4.20.13. Default value of this bit is 0.

13	RWL/ RO	**BI**: This bit is RWL for BI-capable components. This bit is reserved for components that do not support BI. Devices that require BI for managing coherency are permitted to hardwire this bit to 1.[4] 0: Device is not permitted to issue BISnp requests to this range. 1: Device is permitted to issue BISnp requests to this range.
..	..	..

Update section *8.2.4.19.12 Committing Decoder Programming as follows*

Regardless of the setting of the Lock on Commit bit, the decoder logic in a UIO-capable switch or root port shall ensure that the number of decoders configured with UIO=1 does not exceed the number of UIO-capable decoders encoded in the CXL HDM Decoder Capability register (see Section 8.2.4.20.1). If software attempts to violate this restriction, the decode logic shall set ErrorOnCommit=1.

If the device requires BI for managing coherency, software must ensure that BI bit in HDM Decoder Control Register is set before committing the HDM decoder, otherwise the device operation is undefined. Software must ensure the device and any applicable DSPs, USPs and Root Port are configured such that the device is able to issue BISnp request before committing any HDM decoder with BI bit set, otherwise the device operation is undefined.

Decoder logic shall set either *Committed* or *Error Not Committed* flag within 10 ms of a write to the Commit bit.

G20 IDE and LOpt Interactions

Update section *8.2.4.21.1 CXL IDE Capability (Offset 00h) as follows*

Bit Location	Attributes	Description
..	..	..

23	HwInit/ RsvdP	**LOpt IDE Capable:** If set, this component supports IDE when the link is operating in latency-optimized 256B Flit Mode (see Figure 11-13 and Figure 11-14). If 0, this component does not support IDE when the link is operating in latency-optimized 256B Flit Mode. If the link is operating in latency-optimized 256B Flit Mode, the System Firmware or System Software must clear CXL_Latency_Optimized_256B_Flit_Enable bit the DVSEC Flex Bus Port Control register (see Section 8.2.1.3.2) in the Downstream Port and then retrain the link prior to enabling IDE. Once IDE is terminated, the System Firmware or System Software may set CXL_Latency_Optimized_256B_Flit_Enable bit the DVSEC Flex Bus Port Control register (see Section 8.2.1.3.2) in the Downstream Port and then retrain the link so it can transition to latency-optimized 256B flit mode. This bit was introduced as part of Version=2.
31:243	RsvdP	**Reserved**

G21 ARB/MUX Error Mark Register attributes and defaults

Update section *8.2.5.3 ARB/MUX Uncorrectable Error Mask Register (Offset 08h) as follows*

Bit	Attributes	Description
0	RW1CS	**PM Timeout Error Mask**: 0 = PM Timeout Error is logged as an Internal Uncorrected Error in the associated root port, similar to CXL.cachemem errors ~~(default)~~ 1 = PM Timeout Error is not recorded or reported The default value for this bit is 1.
1	RW1CS	**L0p Timeout Error Mask**: 0 = L0p Timeout Error is logged as an Internal Uncorrected Error in the associated root port, similar to CXL.cachemem errors ~~(default)~~ 1 = L0p Timeout Error is not recorded or reported The default value for this bit is 1.

31:2	RsvdZ	**Reserved**

G22 Miscellaneous DCD Clarifications

Update Section 7.6.7.6.5 Initiate Dynamic Capacity Add (Opcode 5604h) as follows

...

The command shall fail with Resources Exhausted when the length of the added capacity plus the current capacity present in all extents associated with the specified region exceeds the decode length for that region.

The command shall fail with Invalid Extent List when the Selection Policy is set to Prescriptive and the Extent Count is invalid.

The command shall fail with Retry Required if its execution would cause the specified LD's Dynamic Capacity Event Log to overflow.

...

Update Table 7-62 Initiate Dynamic Capacity Add Request as follows

Byte Offset	Length in Bytes	Description
..	..	..
03h	1	**Region Number:** Dynamic Capacity region to which the capacity is being added. Valid range is from 0 to 7. This field is reserved when the Selection Policy is set to Prescriptive ~~or Enable Shared Access~~.
..	..	..

Update Section 7.6.7.6.6 Initiate Dynamic Capacity Release (Opcode 5605h) as follows

...

The command shall fail with Invalid Input under the following conditions:

- When the command is sent with an invalid Host ID, or an invalid region number, or an unsupported Removal Policy
- When the command is sent with a Removal Policy of Tag-based and the input Tag does not correspond to any currently allocated capacity

- When Sanitize on Release is set but is not supported by the device
- When the command attempts to release only a portion of tagged sharable capacity

...

The command shall fail with Resources Exhausted when the length of the removed capacity exceeds the total assigned capacity for that region or for the specified tag when the Removal Policy is set to Tag-based.

The command shall fail with Invalid Extent List when the Removal Policy is set to Prescriptive and the Extent Count is invalid or when the Extent List includes blocks that are not currently assigned to the region.

The command shall fail with Retry Required if its execution would cause the specified LD's Dynamic Capacity Event Log to overflow, unless the Forced Removal flag is set, in which case the removal happens regardless of whether an Event is logged.

...

Update Table 8-47 Dynamic Capacity Event Record as follows

Byte Offset	Length in Bytes	Description
..	..	..
35	1	Flags • Bit[0]: More: – 0 = this is the last (or only) record associated with this event – 1 = the next Event Record is also associated with this event, providing an additional extent. Records grouped this way shall be continuous in the Event Log, with no unrelated records between them, and shall contain the same Dynamic Capacity Event Type. • Bit[7:1]: Reserved
365h	23	Reserved
..	..	..

Update Table 8-52. Get Event Interrupt Policy Output Payload to add the missing DCD Interrupt Message Number field

Byte Offset	Length in Bytes	Description
..	..	..

04h	1	**Dynamic Capacity Event Log Interrupt Settings**: When enabled, the device shall signal an interrupt when the Dynamic Capacity event log transitions from having no entries to having one or more entries.[1] Bits[1:0]: Interrupt Mode 00b = No interrupts 01b = MSI/MSI-X 10b = Reserved 11b = Reserved Bits[7:~~2~~4]: Reserved Bits[7:4]: Interrupt Message Number - see definition below.

Update Table 8-125 Get Dynamic Capacity Configuration Output Payload as follows

Byte Offset	Length in Bytes	Description
0	1	**Number of Available Regions**: The device shall report the total number of regions of Dynamic Capacity available. Each region may be unconfigured or configured with a different block size and capacity. ~~This is the number of valid region configurations returned in this payload.~~ A DCD shall report between 1 and 8 regions. All other values are reserved.
01h	~~7~~1	~~Reserved~~ Regions Returned: This is the number of region configurations returned in this payload
02h	6	Reserved
..	..	..

Update Section 8.2.9.8.9.3 Add Dynamic Capacity Response (Opcode 4802h) as follows

...

In response to a Add Capacity Event Record, or records grouped via the More flag (see Table 8-47), the host shall respond with exactly one Add Dynamic Capacity Response command, corresponding to the order of the Add Capacity Events received, to update the device with the explicit portions of the added Dynamic Capacity the host is now utilizing. For non-sharable capacity, t~~T~~he host may send the Add Dynamic Capacity Response command with no Extent List, if the host does not utilize any of the added capacity, or an Extent List describing a subset of the original Add Capacity Event Record Extent List. After this command is received, the device is free to reclaim capacity that the host does not utilize. For sharable capacity, the host shall respond with either no Extent List or an Extent List describing the full capacity -- it shall accept all or none of the sharable capacity or the device shall

return Invalid Extent List. When capacity is described by multiple extents as indicated by the More flag (see Table 8-47), the host shall respond with a single response for the entire group.

..

Table 8-129 Add Dynamic Capacity Response Input Payload

Byte Offset	Length in Bytes	Description
..	..	..
04h	41	Flags • Bit[0]: More: – 0 = this is the last (or only) record associated with this event – 1 = the next Add Dynamic Capacity Response Input Payload is also associated with this operation, providing additional extents. Payloads can be grouped this way to overcome limits due to maximum mailbox payload sizes. Payloads grouped this way shall be submitted with no unrelated records between them and shall contain the same mailbox opcode. • Bit[7:1]: Reserved
05h	3	Reserved
..	..	..

8.2.9.8.9.4 Release Dynamic Capacity (Opcode 4803h)

...

The device shall report Invalid Extent List if it detects a malformed Extent List. Examples of a malformed Extent List include:

- Overlapping Starting DPA and Lengths for multiple extents
- Starting DPA not aligned to the Region Block Size
- Length not a multiple of the Region Block Size
- The Extent List covers only a portion of a tagged sharable capacity

...

Table 8-131 Release Dynamic Capacity Input Payload as follows

Byte Offset	Length in Bytes	Description
..	..	..
04h	41	Flags • Bit[0]: More: – 0 = this is the last (or only) record associated with this event – 1 = the next Release Dynamic Capacity Input Payload is also associated with this operation, providing additional extents. Payloads can be grouped this way to overcome limits due to maximum mailbox payload sizes. Payloads grouped this way shall be submitted with no unrelated records between them and shall contain the same mailbox opcode. • Bit[7:1]: Reserved
05h	3	Reserved
..	..	..

Update Section 9.13.3 Dynamic Capacity Device (DCD) as follows

Dynamic Capacity is a feature of a CXL memory device that allows memory capacity to change dynamically without the need for resetting the device. A DCD is a CXL memory device that implements Dynamic Capacity. Unlike a traditional DPA range that a CXL memory device might support, a Dynamic Capacity DPA range is subdivided into 1 to 8 DC regions, each of which is subdivided by the DCD into a number of fixed-size blocks, referred to as DC blocks. The host software is expected to program the maximum potential capacity utilizing one or more HDM decoders to span the entire DPA range of all configured regions. The DCD controls the allocation of these DC blocks to the host and utilizes events to signal the host when changes to the allocation of these DC blocks occurs. The DCD communicates the state of these DC blocks through an Extent List that describes the starting DPA and length of all DC blocks the host can access. The Extent List does not contain extents that are still pending acceptance from the host via the Add Dynamic Capacity Response command (see Section 8.2.9.8.9.3). Similarly, the Extent List does contain extents that are still pending release acceptance from the host via the Release Dynamic Capacity (see Section 8.2.9.8.9.4) Figure 9-22 illustrates a typical Extent List. Figure 9-23 illustrates an Extent List in which the DC blocks are shared by multiple hosts. Adding and releasing capacity utilizes the Extent List to control the host's access to portions of the memory without the need to alter the HDM programming of the total potential Dynamic Capacity.

...

The basic sequence to add Dynamic Capacity to a host:

- The DCD adds a Add Capacity Event Record (see Section 8.2.9.2.1.5) to the device's Dynamic Capacity Event Log containing the extent of the capacity being added, sets the Dynamic Capacity Event Log bit in the Event Status register and, if enabled, generates an interrupt to alert the host to the new event record. The DCD does this for each extent in the Add Capacity operation being performed, using the More flag as necessary (see Table 8-47), avoiding overflow, and allowing the host to consume the events as necessary to complete the

operation. If the Dynamic Capacity Event Log overflows at any point, the host shall utilize Get Dynamic Capacity Extent List to retrieve the current list of host accessible DC blocks.

...

The basic sequence to release Dynamic Capacity from a host:

- The DCD adds a Release Capacity Event Record to the device's Dynamic Capacity Event Log (see Section 8.2.9.2.1.5) containing the extent of the capacity it is requesting to be released, sets the Dynamic Capacity Event Log bit in the Event Status register and, if enabled, generates an interrupt to alert the host to the new event record. The DCD does this for each extent in the Release Capacity operation being performed, using the More flag as necessary (see Table 8-47), avoiding overflow, and allowing the host to consume the events as necessary to complete the operation. If the Dynamic Capacity Event Log overflows at any point, the host shall utilize Get Dynamic Capacity Extent List to retrieve the current list of host accessible DC blocks.

..

Devices may forcefully release Dynamic Capacity from a host:

Host access to the released capacity may be immediately disabled and the DCD behaves as if the capacity is no longer allocated to the host. The DCD adds a Forced Capacity Release Event Record to the device's Dynamic Capacity Event Log containing the extent of the capacity being released, sets the Dynamic Capacity Event Log bit in the Event Status Register and, if enabled, generates an interrupt to alert the host to the new event record. If the Dynamic Capacity Event Log overflows at any point, the forced removal still takes place and the host shall utilize Get Dynamic Capacity Extent List to retrieve a new list of host accessible DC blocks.

..

G23 Scan Media Clarifications

Update Table 1-113 Get Scan Media Results Output Payload as follows

Byte Offset	Length in Bytes	Description
..	..	..
10h	1	Scan Media Flags Bit[0]: More Media Error Records - When set, the device has more Media Error Records to return for the given Scan Media address range. The host should keep issuing the Get Scan Media Results command ~~with the same Scan Media Restart Physical Address & Scan Media Restart Physical Address Length~~ and retrieve records until this indicator is no longer set. ..

G24 CXL.io Throttling Typo in Flit Type

This errata removes the CXL.io NOP Flit Type encoding referenced in the CXL.io Throttling feature description, as the value referenced was stale after the encoding definition was changed for CXL.io NOP.

Update section 6.4.3.1.2 as follows:

6.4.1.3.2 CXL.io Throttling

The Upstream Port must communicate to the Downstream Port during Phase 1 of alternate protocol negotiation if its CXL.io inbound path does not support receiving consecutive CXL.io flits (including CXL.io NOP flits with a DLLP payload) at a link speed of 64 GT/s. For the purpose of this feature, consecutive CXL.io flits are CXL.io Payload flits or CXL.io NOP flits with a DLLP payload ~~two flits with Flit Type encoding of 01b~~ that are not separated by either an intervening flit not associated with CXL.io~~with a different Flit Type encoding~~ or an intervening Ordered Set. Downstream Ports are required to support throttling transmission of CXL.io traffic to meet this requirement if the Upstream Port advertises this bandwidth limitation in the Modified TS1 Ordered Set (see **Table 6-9.**). One possible usage model for this is Type 3 memory devices that need 64 GT/s link bandwidth for CXL.mem traffic but do not have much CXL.io traffic; this feature enables such devices to simplify their hardware to provide potential buffer and power savings.

G25 Unexpected Flit Type Error in 256B Flit Mode

The current specification does not define how a receiver should handle a flit with Unexpected Flit Type in 256B Flit Mode. This errata specifies that an Unexpected Flit Type should be logged in the standard PCIe Flit Logging Extended Capability, in the Flit Error Log 1 Register, as an Unrecognized Flit.

Add section 6.2.3.3 as follows:

6.2.3.2 Framing Errors in 256B Flit Mode

An Unexpected Flit Type error is detected upon receiving a Flit with a Flit Type encoding associated with a Protocol that was not enabled during negotiation. For example, if a CXL.cachemem Flit Type is received while only CXL.io is enabled, this must be handled as an Unexpected Flit Type error. This is logged as an Unrecognized Flit in the PCIe Flit Logging Extended Capability, Flit Error Log 1 Register. Any interrupt signaling as a result of the logged error follows the PCIe specification definition.

Update Table 6-5 as follows to state CXL.cachemem flit type encoding is reserved if CXL.cachemem is not enabled:

Flit Header Field	Flit Header Bit Location	Description
Flit Type[1:0]	[7:6]	• 00b = Physical Layer IDLE flit or Physical Layer NOP flit or CXL.io NOP flit • 01b = CXL.io Payload flit • 10b = If CXL.cachemem is enabled, CXL.cachemem Payload flit or CXL.cachemem-generated Empty flit; reserved if CXL.cachemem is not enabled • 11b = ALMP Please refer to Table 6-6 for more details.

Prior Flit Type	[5]	• 0 = Prior flit was a NOP or IDLE flit (not allocated into Replay buffer) • 1 = Prior flit was a Payload flit or Empty flit (allocated into Replay buffer)
Type of DLLP Payload	[4]	• If (Flit Type = (CXL.io Payload or CXL.io NOP): Use as defined in PCIe Base Specification • If (Flit Type != (CXL.io Payload or CXL.io NOP)): Reserved
Replay Command[1:0]	[3:2]	Same as defined in PCIe Base Specification.
Flit Sequence Number[9:0]	{[1:0], [15:8]}	10-bit Sequence Number as defined in PCIe Base Specification.

G26 Buried State on Memory Protocol

This errata replaces G17. The prior errata was missing the corresponding updates to the Table 3-47 and a functional change was added to make MemRd and MemInv follow a common set of rules for Buried State.

The buried cache rules added in CXL3.0 were found to be overly restrictive and need to be relaxed to allow for hosts to resolve conflicts and without creating a very sub-optimial caching in a host for HDM-D/DB. The updated rules in section 3.3.11.1 are captured in this errata. Example flows will be added into future specifications to show problematic cases that drove the changes.

3.3.11.1 Buried Cache State Rules for HDM-D/HDM-DB

Buried Cache state for CXL.mem protocol refers to the state of the cacheline registered by the host's Home Agent logic (HA) for a cacheline address when a new Req or RwD message is being sent. This cache state could be a cache that is controlled by the host, but does not cover the cache in the device that is the owner of the HDM-D/HDM-DB memory. These rules are applicable to only HDM-D/HDM-DB memory where the device is managing coherence.

For implementations that allow multiple outstanding requests to the same address, the possible future cache state must be included as part of the buried cache state. To avoid this complexity, it is recommended to limit to one Req/RwD per cacheline address.

Buried Cache state rules for host-issued CXL.mem Req/RwD messages:

- Must not issue a MemRd/MemInv/MemInvNT (MetaValue=I)~~/MemRdData~~ if the cacheline is buried in Modified, Exclusive, or Shared state.
- May not issue a MemRd/MemInv/MemInvNT (MetaValue=S) or MemRdData if the cacheline is buried in Modified or Exclusive, but is allowed to issue when the host has Shared or Invalid.
- May issue a MemRd/MemInv/MemInvNT (MetaValue = A) from any state.
- May issue a MemRd/MemInv/MemInvNT (MetaField = NoOp) from any state. Note that the final host cache state may result in a downgraded state such as Invalid when initial buried state exists and conflicting BISnp result in the buried state being downgraded.
- ~~Must not issue MemInv/MemInvNT if the cacheline is buried in Modified or Exclusive state. The Device may request ownership in Exclusive state as an upgrade request from Shared state.~~
- May issue MemClnEvct from Shared or Exclusive state.
- May issue MemWr with SnpType=SnpInv only from I-state. This use of this encoding is not allow for HDM-DB memory regions where coherence extends to multiple hosts (e.g. Coherent Shared FAM as described in Section 2.4.4).
- MemWr with SnpType=NoOp may only be issued from Modified state.

summarizes the Req message and RwD message allowance for Buried Cache state. MemRdFwd/MemWrFwd/BIConflict are excluded from this table because they are response messages.

Table 3-47 Allowed Opcodes for HDM-D/HDM-DB Req and RwD Messages per Buried Cache State

CXL.mem Req/RwD				Buried Cache State			
Opcodes	**MetaField**	**MetaValue**	**SnpType**	**Modified**	**Exclusive**	**Shared**	**Invalid**
~~MemRd/~~MemRdData	All Legal Combinations		All Legal Combinations			X	X
MemClnEvct					X	X	
MemRd/MemInv/MemInvNT	MS0/EMD	A		X1	X	X ~~(When MV=A)~~	X
		S				X	X
		I					X
	No-Op	N/A		X1	X	X	X
	EMD	Explicit No-Op					
MemWr	All Legal Combinations		~~SnpType=~~No-Op	X			
			~~SnpType=~~SnpInv				X

[1]Requesters that have active reads with buried-M state must expect data return to be stale. It is up to the requester to ensure that possible stale data case is handled in all cases including conflicts with BISnp.

G27 Responses for Requests Targeting NXM

The CXL specification is incomplete regarding CXL.mem requests targeting non-existent memory (NXM). It includes the MemData-NXM opcode for MemRd/MemRdData requests (that decode to non-existent memory) but does not mention how to handle the other request opcodes (e.g., MemInv). The new section, below, fits within Section 3.3 "CXL.mem" between current Section 3.3.10 and 3.3.11 and discusses the need for special handling while providing a table covering all opcodes. This errata also adds cross-references to this new section in existing tables:

3.3.11 Responses for Requests Targeting NXM

Device responses to CXL.mem requests differ between HDM-H regions and HDM-D/HDM-DB regions, which creates an ambiguity when device receives a CXL.mem request it cannot map to a specific memory region. In this situation, devices shall respond according to Table 1-1. CXL.mem Responses for Requests to Non-existent Memory Requesting device must accept and properly handle these responses regardless of its memory region decode results.

The ambiguity mentioned above is for reads and for some MemInv* cases. For reads, the response is DRS only for HDM-H or a DRS+NDR for HDM-D*. For MemInv*, HDM-H returns Cmp opcode and HDM-D/HDM-DB may expect only Cmp-E or Cmp-S as show in Table C-3 "HDM-D/HDM-DB Memory Requests".

The capability to support MemData-NXM is exposed in the "CXL HDM Decoder Capability Register" bit 20 (see Section 8.2.4.19.1).

Table 1-1. CXL.mem Responses for Requests to Non-existent Memory

CXL.mem Request	Device Response when NXM
MemRd, MemRdData	MemData-NXM See Table 8-27 "CXL.mem Read Response – Error Cases" for additional details.
MemInv, MemInvNT, MemClnEvct, MemWr, MemWrPtl	Cmp

End of new section. The following are changes to cells in existing tables.

Table 3-53. S2M DRS Opcodes

Row "MemData-NXM", Column "Description" – Add cross reference to new section 3.13.

Table C-3. HDM-D/HDM-DB Memory Requests

Row "MemRd + MemData-NXM", Column "Description" – Add cross reference to new section 3.13.

Rows "MemInv", Column "Device Response, S2M NDR" – Add footnote to "Cmp-E" and "Cmp-S" cells with footnote indicating NXM case exception and cross reference to new section 3.13.

Row "MemRdData + MemData-NXM", Column "Description" – Add cross reference to new section 3.13.

Table C-6. HDM-H Memory Request

Footnote 2: Add cross reference to new section 3.13.

G28 Reserved Bit field forwarding

The CXL specification does not stated any requirement for Reserved bit forwarding in a switch. The new section below addresses the required handling for reserved bits. This fits within Section 7.3 "CXL.io, CXL.cachemem Decode and Forwarding" and under the sub-set for 7.3.2 CXL.cache and 7.3.3 CXL.mem.

7.3.2.3 CXL.Cache Reserved bit forwarding

A switch shall forward 256B Flit messages reserved bits between the ingress port and the egress port. Both HBR and PBR formats are defined for 256B flit messages where a switch can translate between those formats. When doing the translation between HBR and PBR formats defined for 256B flits the Reserved bits shall be preserved. When a switch with 256B flit capability sends to a port with 68B flit format the reserved bits shall be set to zero. Similarly, messages received as 68B flit formats shall never have reserved bits forwarded to a port with 256B flit messages.

Note: The reason for forwarding of reserved bits is to allow new features to be supported without requiring changes to existing switches. The reason for not forwarding in 68B flit format is that new features are expected to be added only to 256B flit formats so there is no need to support the complexity of translating reserved bits to/from 68B flits.

7.3.3.3 CXL.Mem Reserved bit forwarding

CXL.mem follows the same rules as CXL.cache as defined in Section 7.3.2.3.

G29 S2M Opcodes for 256B Flit only

The CXL.mem protocol has added new features that only apply to 256B flits. For M2S Req/RwD the opcode table notes the opcodes through use of a footnote. This was not done for S2M NDR/DRS messages and this errata adds the footnote to those opcode tables. Table 3-50 is in Section 3.3.9 "S2M No Data Response (NDR)" and Table 3-xx is in Section…

Note that the errata shows the foot note at the bottom of the page with opcode highlight, but when merged into the specification this will be attached to each table without the highlighting.

Table 3-50. S2M NDR Opcodes

Opcode	Description	Encoding
Cmp	Completions for Writebacks, Reads and Invalidates.	000b
Cmp-S	Indication from the DCOH to the Host for Shared state.	001b
Cmp-E	Indication from the DCOH to the Host for Exclusive ownership.	010b
Cmp-M	Indication from the DCOH to the Host for Modified state. This is optionally supported by host implementations and devices must support disabling of this response.	011b
BI-ConflictAck[4]	Completion of the Back-Invalidate conflict handshake.	100b
CmpTEE[1]	Completion for Writes (MemWr*) with TEE intent. Does not apply to any M2S Req.	101b
Reserved	Reserved	<Others>

Table 3-53. S2M DRS Opcodes

Opcode	Description	Encoding
MemData	Memory read data. Sent in response to Reads.	000b

4 Only support in 256B flit mode.

MemData-NXM	Memory Read Data to Non-existent Memory region. This response is only used to indicate that the device or the switch was unable to positively decode the address of the MemRd as either HDM-H or HDM-D*. Must encode the payload with all 1s and set poison if poison is enabled. This special opcode is needed because the host will have expectation of a DRS only for HDM-H or a DRS+NDR for HDM-D*, and this opcode allows devices/switches to send a single response to the host, allowing a deallocation of host tracking structures in an otherwise ambiguous case.	001b
MemDataTEE1	Same as MemData but in response to MemRd* with TEE attribute.	010b
Reserved	Reserved	<Others>

G30 Chapter 7 Errata

In Section 7.2.1.3, make the following update:

In the case where the switch, FM, and host boot at the same time:

1. VCSs are statically defined.
2. DSP vPPBs within each VCS are unbound and presented to the host as Link Down.
3. Switch discovers downstream devices and presents them to the FM.
4. Host enumerates the VH and configures the DVSEC registers.

In Section 7.3.4, make the following update:

All PPBs are FM-owned. A PPB can be connected to a port that is disconnected or~~,~~ linked up ~~as an RCD, CXL SLD, or CXL MLD~~. SLD components can be bound to a host or unbound. Unbound SLD components can be accessed by the FM using CXL.io transactions via the FM API. LDs within an MLD component can be bound to a host or unbound. Unbound LDs are FM-owned and can be accessed through the switch using CXL.io transactions via the FM API.

In Section 7.5, make the following update:

Table 7-13. CXL Switch RAS

~~Host~~Triggering Action	Description	Switch Action for Non-pooled Devices	Switch Action for Pooled Devices
Switch boot	Optional power-on reset pin	Assert PERST# Deassert PERST#	Assert PERST# Deassert PERST#
Upstream PERST# asserted	VCS fundamental reset	Send Hot Reset	Write to MLD DVSEC to trigger LD Hot Reset of the associated LD **Note:** Only the FMLD provides the MLD DVSEC capability.

FM issues port reset command	Reset of an FM-owned DSP	Send Hot Reset	Send Hot Reset
PPB Secondary Bus Reset	Reset of an FM-owned DSP	Send Hot Reset	Write to MLD DVSEC to trigger LD Hot Reset of all LDs
USP receivesd Hot Reset	VCS fundamental reset	Send Hot Reset	Write to MLD DVSEC to trigger LD Hot Reset of the associated LD
USP vPPB Secondary Bus Reset	VCS US SBR	Send Hot Reset	Write to MLD DVSEC to trigger LD Hot Reset of the associated LD
DSP vPPB Secondary Bus Reset	VCS DS SBR	Send Hot Reset	Write to MLD DVSEC to trigger LD Hot Reset of the associated LD
Host writes FLR	Device FLR	No switch involvement	No switch involvement
Switch watchdog timeout	Switch fatal error	Equivalent to power-on reset	Equivalent to power-on reset

In Section 7.6.6.7, make the following update:

When a device is Hot-Added to an unbound port on a switch, the FM receives a notification and is responsible for binding as described in the steps below:

1. The switch notifies the FM by generating **Physical Switch Event Records** as the Presence Detect sideband signal is asserted or when a Link Up is detected if the PPB does not support Presence Detectand the port links up.

In Section 7.6.7.3.1, make the following update:

When sent to an MLD, this provided command is tunneled by the FM-owned LD to the specified LD, as illustrated in the example in Figure 7-22 of a "Set LSA Request" being tunneled to LD 1 in an MLD.

In Section 7.6.7.6.1, make the following update:

Table 7-61. Get DCD Info Response Payload

Byte Offset	Length in Bytes	Description
00h	1	**Number of Hosts**: Total number of hosts that the device supports. This field shall have a minimum value of 1.
01h	1	**Number of Supported DC Regions**: The device shall report the total number of Dynamic Capacity Regions available per hostLD. DCDs shall report between 1 and 8 regions. All other encodings are reserved.
…		