Errata for the Compute Express Link™ (CXL™) Specification Revision 1.1

February 2020

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

This CXL **Specification Revision 1.1** (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:

If you are a Member of the CXL Consortium (sometimes referred to as a "**CXL Member**"), and even if you have received this publicly-available version of this CXL Specification after agreeing to CXL Consortium's Evaluation Copy Agreement (a copy of which is available https://www.computeexpresslink.org/download-the-specification, each such CXL Member must also be 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:

If you are **not** a CXL Member and have received this publicly-available version of this CXL Specification, your use of this document is subject to your compliance with, and is limited by, all of the terms and conditions of the CXL Consortium's Evaluation Copy Agreement (a copy of which is available at https://www.computeexpresslink.org/download-the-specification**).**

In addition to the restrictions set forth in the CXL Consortium's Evaluation Copy Agreement, 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: **"© 2019-2020 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.

Except for the limited rights explicitly given to a non-CXL Member pursuant to the explicit provisions of the CXL Consortium's Evaluation Copy Agreement which governs the publicly-available version of this CXL Specification, 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 benefits and/or rights as a CXL Member under any CXL Governing Documents.

LEGAL DISCLAIMERS FOR ALL PARTIES:

THIS DOCUMENT AND ALL SPECIFICATIONS AND/OR OTHER CONTENT PROVIDED HEREIN IS 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.

NOTICE TO ALL PARTIES REGARDING THE PCI-SIG UNIQUE VALUE PROVIDED IN THIS CXL SPECIFICATION:

NOTICE TO USERS: THE UNIQUE VALUE THAT IS PROVIDED IN THIS CXL SPECIFICATION IS FOR USE IN VENDOR DEFINED MESSAGE FIELDS, DESIGNATED VENDOR SPECIFIC EXTENDED CAPABILITIES, AND ALTERNATE PROTOCOL NEGOTIATION ONLY AND MAY NOT BE USED IN ANY OTHER MANNER, AND A USER OF THE UNIQUE VALUE MAY NOT USE THE UNIQUE VALUE IN A MANNER THAT (A) ALTERS, MODIFIES, HARMS OR DAMAGES THE TECHNICAL FUNCTIONING, SAFETY OR SECURITY OF THE PCI-SIG ECOSYSTEM OR ANY PORTION THEREOF, OR (B) COULD OR WOULD REASONABLY BE DETERMINED TO ALTER, MODIFY, HARM OR DAMAGE THE TECHNICAL FUNCTIONING, SAFETY OR SECURITY OF THE PCI-SIG ECOSYSTEM OR ANY PORTION THEREOF (FOR PURPOSES OF THIS NOTICE, "**PCI-SIG ECOSYSTEM**" MEANS THE PCI-SIG SPECIFICATIONS, MEMBERS OF PCI-SIG AND THEIR ASSOCIATED PRODUCTS AND SERVICES THAT INCORPORATE ALL OR A PORTION OF A PCI-SIG SPECIFICATION AND EXTENDS TO THOSE PRODUCTS AND SERVICES INTERFACING WITH PCI-SIG MEMBER PRODUCTS AND SERVICES).

Contents

Revision History

Revision	Description	Date
1.0	First Release: Errata E1-E16.	October 2019
2.0	Second Release: Added E17 and E18.	December 2019
3.0	Third Release: Added E19-E24.	January 2020
4.0	Fourth Release: Added E25-E27. Note E26 replaces E18 due to errors in E18 that incorrectly changed the CRC data mask values.	February 2020

E1 CXL.AL and Flex Bus.AL Typo

In general, Flex Bus.AL should be substituted with Flex Bus.CXL throughout the specification document. Additionally, in section 11.4, make the following changes:

11.4 CXL Viral Handling

. . .

When a CXL~~.AL~~ device goes into Viral, the upstream CXL.io shall perform the following:
- Master Abort Upstream Requests
- Completer Abort Upstream Completions
- Signal Failed Response for Downstream Completions

E2 Control SKP Ordered Set Frequency

In section 6.7.1, make the following changes:

6.7.1 Control SKP Ordered Frequency and L1/Recovery Entry

In Flex Bus.CXL mode, if sync header bypass is enabled, the following rules apply:

- After the SDS, the physical layer must schedule a control SKP Ordered Set or SKP Ordered Set after every 340 data blocks, unless it is exiting the data stream. Note: The control SKP OSs are alternated with regular SKP OSs at 16 GT/s or higher speeds; at 8 GT/s, only SKP OSs are scheduled.

E3 CXL Downstream Port Supported PCIe Capabilities

In section 7.2.1.1, Table 60, make the following changes:

Table 60. CXL Downstream Port Supported PCIe Capabilities and Extended Capabilities

Supported PCIe Capabilities and Extended Capabilities	Exceptions[1]	Notes
PCI Express Capability	Slot Capabilities, Slot Control, Slot Status, Slot Capabilites 2, Slot Control 2, and Slot Status 2 registers are not applicable.	None
PCI Power Management Capability	None	None
MSI Capability	None	None
Advanced Error Reporting Extended Capability	None	Required for CXL despite being optional for PCIe
ACS Extended Capability	None	None
Multicast Extended Capability	None	None

Downstream Port Containment Extended Capability	None	None
Designated Vendor-Specific Extended Capability (DVSEC)	None	Please refer to section ~~Figure~~ 7.2.1.3 for Flex Bus Port DVSEC definition.
Secondary PCI Express Extended Capability	None	None
Data Link Feature Extended Capability	None	None
Physical Layer 16.0 GT/s Extended Capability	None	None
Physical Layer 32.0 GT/s Extended Capability	None	None
Lane Margining at the Receiver Extended Capability	None	None
Alternate Protocol Extended Capability	None	None

E4 CXL Upstream Port Supported PCIe Capabilities

In section 7.2.1.2, Table 61, make the following changes:

Table 61. CXL Upstream Port Supported PCIe Capabilities and Extended Capabilities

Support PCIe Capabilties and Extended Capabilties	Exceptions[1]	Notes
PCI Express Capability	None	~~N/A~~None
Advanced Error Reporting Extended Capability	None	Required for CXL despite being optional for PCIe.
~~Multicast Extended Capability~~	~~None~~	~~N/A~~
Virtual Channel Extended Capability	None	VC0 and VC1

Designated Vendor-Specific Extended Capability (DVSEC)	None	Please refer to section ~~Figure~~ 7.2.1.3 for Flex Bus Port DVSEC definition.
Secondary PCI Express Extended Capability	None	None
Data Link Feature Extended Capability	None	None
Physical Layer 16.0 GT/s Extended Capability	None	None
Physical Layer 32.0 GT/s Extended Capability	None	None
Lane Margining at the Receiver Extended Capability	None	None
Alternate Protocol Extended Capability	None	None

E5 CXL Power Management Messages

In section 3.1.2, Table 3, make the following changes:

Field	Description	Notes
Payload[95:0]	. . . AGENT_INFO: If Param.Index == 0, 7:0 - ~~REVISION_ID~~ CAPABILITY_VECTOR 0 - Always set to indicate support for CXL 1.1 PM messages 7:1 - Reserved . . . 15:8 – PrepType 0x00 => General Prep All others reserved ~~0x01 => Early Prep;~~ ~~0x02 => Reset Entry Start (first checkpoint for CXL device blocks during a Reset event/power state transition);~~	CXL Agent must treat the TARGET_AGENT_ID field as Reserved when returning credits to Host. Only Index 0 is defined for AGENT_INFO, all other Index values are reserved.

	~~0x03 => Link Turnoff (typically the last checkpoint during a Reset event/power state transition);~~ 17:16 - Reserved ~~Phase~~ ~~0x00 => Phase 0~~ ~~0x01 => Phase 1~~ ~~0x02 => Phase 2~~ ~~0x03 => Phase 3~~ ~~All others reserved~~ PMREQ: 31:0 - PCIe LTR format All others reserved	

E6 CXL Configuration Space Registers

In chapter 7, Table 57, make the following changes:

Attribute	Description
RO	Read Only
RO-V	Read-Only-Variant
RW	Read-Write
RWS	Read-Write-Sticky
RWO	Read-Write-On~~c~~e-To_Lock Field becomes RO after writing one to it.
RWL	Read-Write-Lockable
RW1CS	Read-Write-One-To-Clear-Sticky

In section 7.1.1.3, make the following changes:

Bit	Attributes	Description
13:0	N/A	Reserved (RSVD).
14	RW1CS	Viral_Status: When set, indicates that the CXL device has entered Viral self-isolation mode. See Section 11.4, "CXL Viral Handling" on page 198 for more details. Note: Viral condition is cleared by warm reset or cold reset, but Viral_Status bit survives warm reset.
15	N/A	Reserved (RSVD).

E7 CXLCM Flit Packing Rules

In Section 4.2.5, add following Flit Packing Rules that clarify Bit Ordering requirements.

4.2.5 Flit Packing Rules

. . .

- For a given slot, lower bit positions are defined as bit positions that appear starting from lower order Byte #. That is, bits are ordered starting from (Byte# 0, Bit# 0) through (Byte #15, Bit# 7).
- For multi-bit message fields like Address[MSB:LSB], lesser significant bits will appear in lower order bit positions.
- Message ordering within a flit is based on flit bit numbering, i.e. the earliest messages are placed at the lowest flit bit positions and progressively later messages are placed at progressively higher bit positions. Examples: An M2S Req 0 packed in Slot 0 precedes an M2S Req 1 packed in Slot 1. Similarly, a Snoop packed in Slot 1 follows a GO packed in Slot 0, and this ordering must be maintained. Finally, for Header Slot Format H1, an H2D Response packed starting from Byte# 7 precedes an H2D Response packed starting from Byte# 11.

E8 CXLCM Slot Format Diagram Updates

In Section 4.2.3, following diagrams for Header Slot Formats need to be updated to show Slot 3 bit encodings in correct bit positions. Also fixed Figure 50, which showed Addr[51:6] in CXL1.1, but should have show Addr[51:5] to match address bits required for M2S Req.

4.2.3.2 H2D and M2S Formats

Figure 45. H0 - H2D Req + H2D Resp

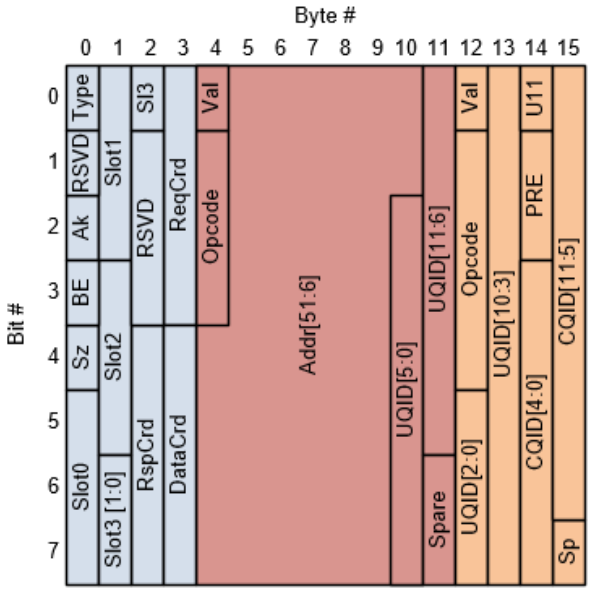


Figure 46. H1 - H2D Data Header + H2D Resp + H2D Resp

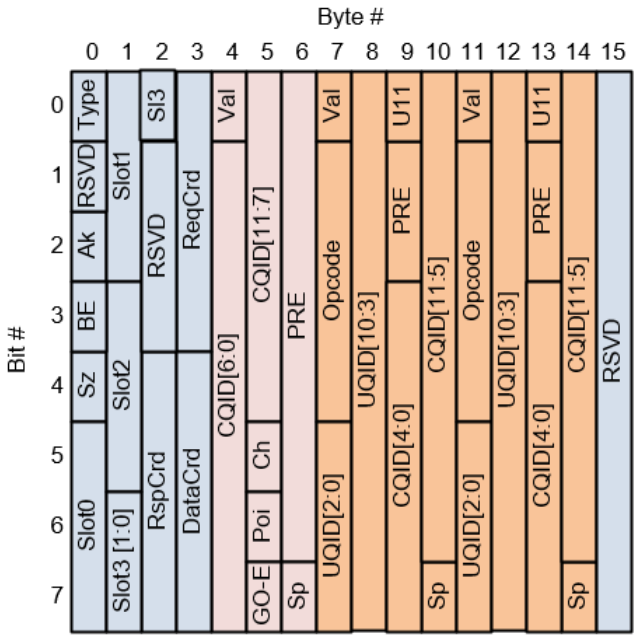


Figure 47. H2 - H2D Req + H2D Data Header

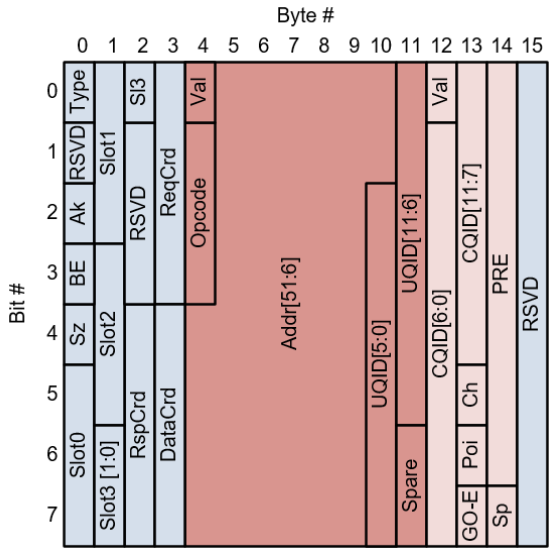


Figure 48. H3 - 4 H2D Data Header

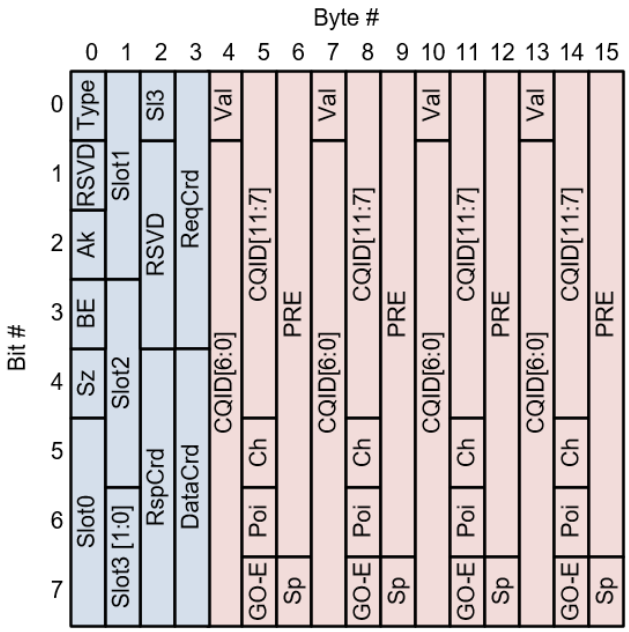


Figure 49. H4 - M2S RwD Header

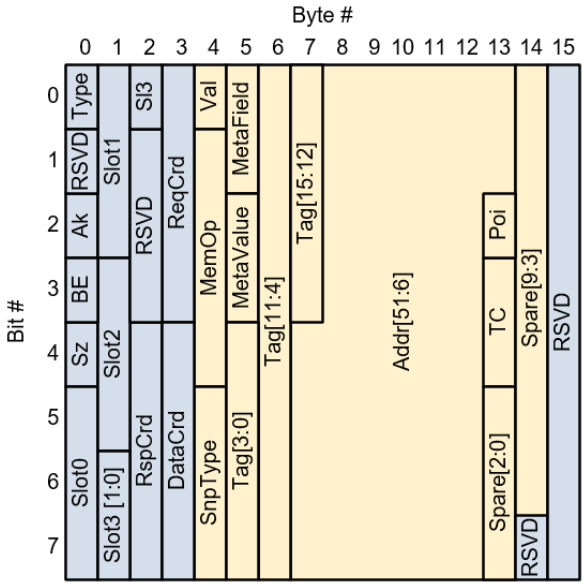
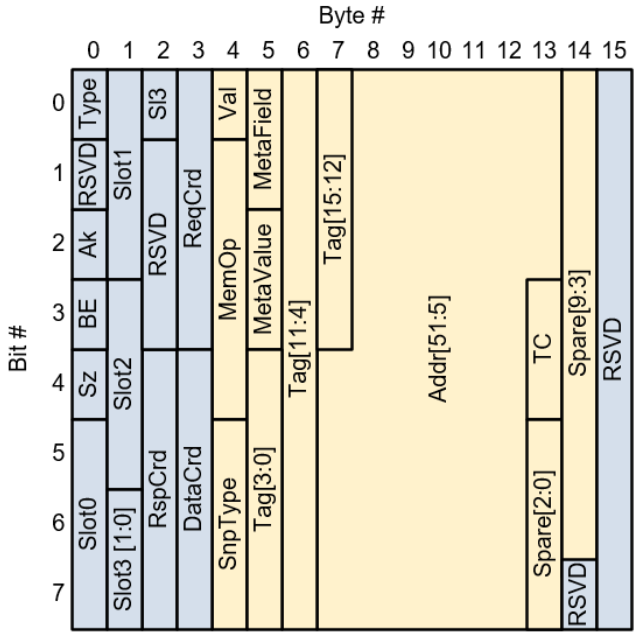


Figure 50. H5 - M2S Req



4.2.3.3 D2H and S2M Formats

Figure 58. H0 - D2H Data Header + 2 D2H Resp + S2M NDR

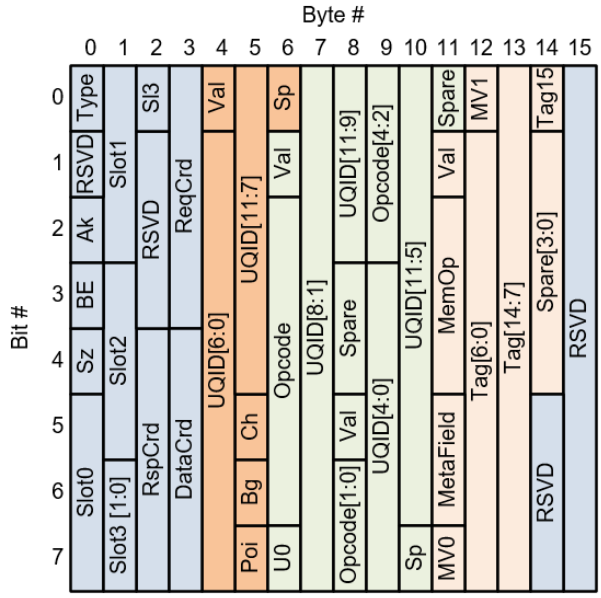


Figure 59. H1 - D2H Req + D2H Data Header

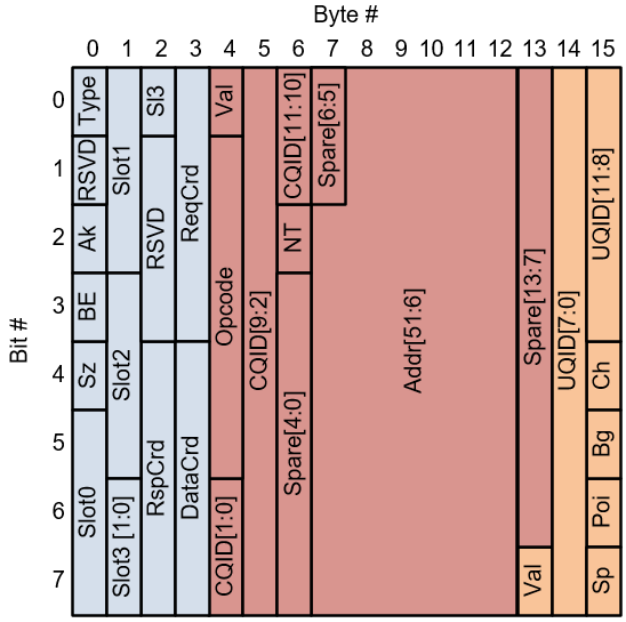


Figure 60. H2 - 4 D2H Data Header + D2H Resp

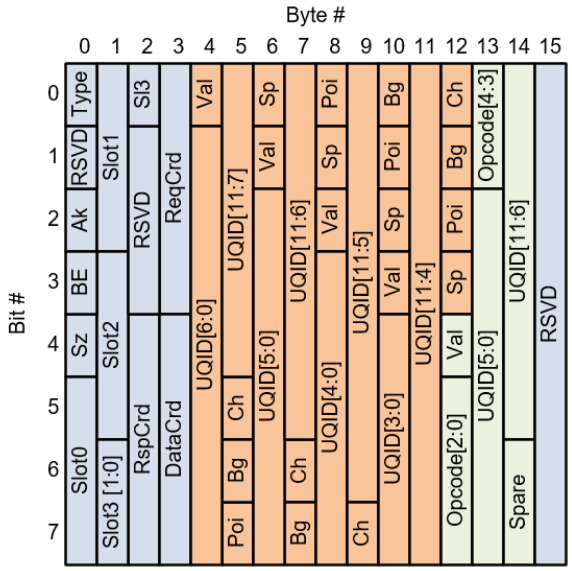


Figure 61. H3 - S2M DRS Header + S2M NDR

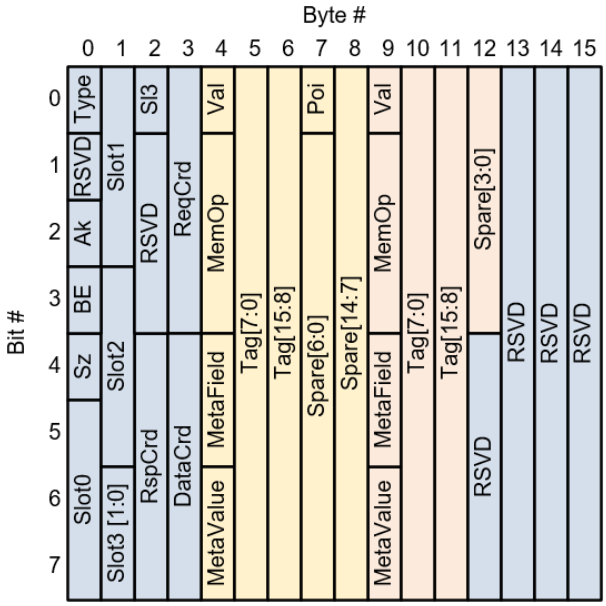


Figure 62. H4 - 2 S2M NDR

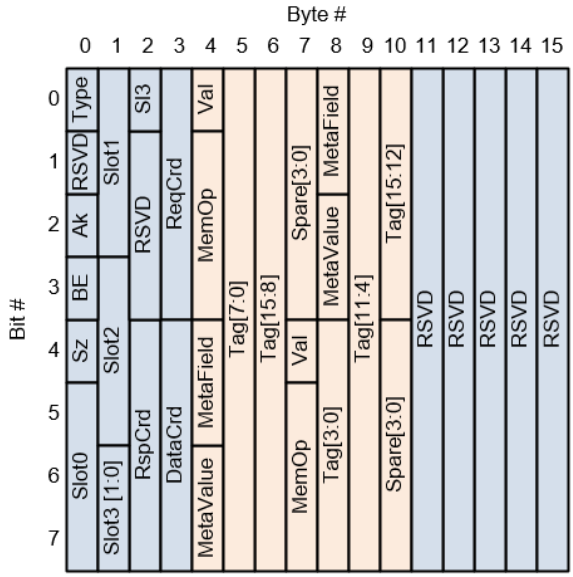
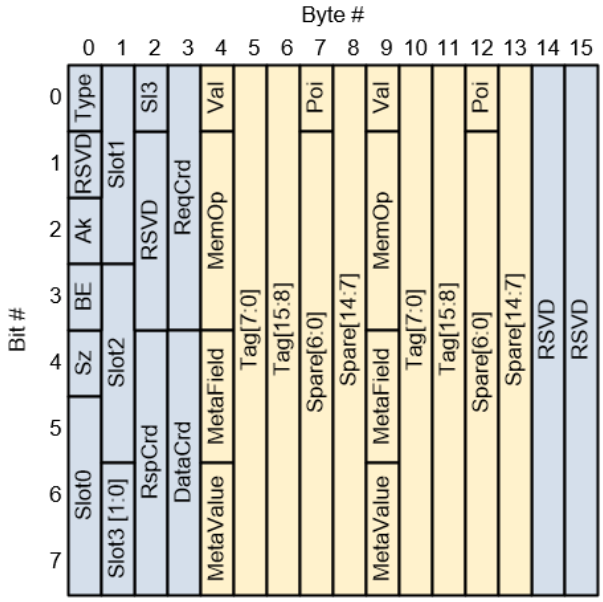


Figure 63. H5 - 2 S2M DRS



E9 CLFlush Definition and Clarification for Biasing

In section 3.2.4.1.13, make the following updates:

3.2.4.1.13 ClFlush

This is a request to the Host to invalidate the cache-line specified in the address field. The typical response is GO-I that will be sent from the Host upon completion in memory.

However, the Host may keep tracking the cache line in Shared state if the Core has issued a Monitor to an address belonging in the cache line. Thus, the Device must not rely on CLFlush/GO-I as an only and sufficient condition to flip a cache line from Host to Device bias mode. Instead, the Device must initiate RdOwnNoData and receive an H2D Response of GO-E before it updates its Bias Table and may subsequently access the cache-line without notifying the Host.

Under error conditions, a ClFlush request may receive the line in the Error (GO-Err) state. The device is responsible for handling the error appropriately.

E10 Buried Cache State Rules and Multiple Access Over CXL.cache

Add a Section 3.2.5.14 for Buried Cache State Rules. Update text in sections 3.2.5.6, 3.2.5.7, and 3.2.5.8

3.2.5.14 Buried Cache State Rules

Whenever the Device initiates a new request on CXL.Cache protocol, Buried Cache state refers to the state of the cache line registered in the Device's Coherency engine (DCOH) for which a particular request is being sent.

Buried Cache State Rules:

- The Device must not issue a Read for a cache line if it is buried in Modified, Exclusive, or Shared state.
- The Device must not issue RdOwnNoData if the cache line is buried in Modified or Exclusive state. The Device may request for ownership in Exclusive state as an upgrade request from Shared state.
- The Device must not issue a Read0-Write if the cache line is buried in Modified, Exclusive, or Shared state. The Host typically responds with GO-I (downgraded state) for such requests.
- All *Evict opcodes must adhere to apropos use case. For example, the Device is allowed to issue DirtyEvict for a cache line only when it is buried in Modified state. For performance benefits, it is recommended that the Device should not silently drop a cache line in Exclusive or Shared state and instead use CleanEvict* opcodes towards the Host.
- The CacheFlushed Opcode is not specific to a cache line, it is an indication to the Host that all of the Device's caches are flushed. Thus, the Device must not issue CacheFlushed if there is any cache line buried in Modified, Exclusive, or Shared state.

Table 20E describes which Opcodes in D2H requests are allowed for a given Buried Cache State:

Table 20E: Allowed Opcodes Per Buried Cache State

D2H Requests		Buried Cache State			
Opcodes	Semantic	Modified	Exclusive	Shared	Invalid
RdCurr	Read				✓
RdOwn	Read				✓

RdShared	Read				✓
RdAny	Read				✓
RdOwnNoData	Read0			✓	✓
ItoMWr	Read0-Write				✓
MemWr	Read0-Write				✓
CLFlush	Read0				✓
CleanEvict	Write		✓		
DirtyEvict	Write	✓			
CleanEvictNoData	Write		✓	✓	
WOWrInv	Write				✓
WOWrInvF	Write				✓
WrInv	Write				✓
CacheFlushed	Read0				✓

3.2.5.6 Multiple Reads to the same cache line

Multiple read requests (cacheable or uncacheable) to the same cache line are allowed only in the following specific cases where host tracking state is consistent regardless of the order requests are processed. The Host can freely reorder requests, so the device is responsible for ordering requests when required. For host memory, multiple RdCurr and/or CLFlush are allowed. For these commands the device ends in I-state, so there is no inconsistent state possible for host tracking of a device cache. With Type 2 devices, in addition to RdCurr and/or CLFlush, multiple RdOwnNoData (bias flip request) is allowed for device attached memory. This case is allowed because with device attached memory the host does not track the device's cache so re-ordering in the host will not create ambiguous state between device and host.

3.2.5.7 Multiple Evicts to the same cache line

~~Multiple Evicts to the same cache line are not allowed. The second Evict may only be issued after the first receives both the CXL.cache GO-I response and the WritePull.~~

~~Since Evict guarantees that the evicted cache line is otherwise in the initiating device, it is impossible to send another Evict without an intervening cacheable Read/Read0 request to that address.~~

Multiple Evicts to the same cache line are not allowed. All Evict messages from the Device provide a guarantee to the Host that the evicted cache line will no longer be present in the Device's caches. Thus, it is impossible to send another Evict for the same cache line without an intervening cacheable Read/Read0 request to that address.

3.2.5.8 Multiple Write Requests to the same cache line

~~Multiple WrInv/WOWrInv/ItoMWr/MemWr to the same cache line are allowed to be outstanding on CXL.cache. The Host can freely reorder requests, so the device is responsible for ordering requests when required.~~

Multiple WrInv/WOWrInv/ItoMWr/MemWr to the same cache line are allowed to be outstanding on CXL.cache. The Host can freely reorder requests, and the Device may receive corresponding H2D Responses in reordered fashion. However, it is generally recommended that the Device should issue no more than one outstanding Write request for a given cache line, and order multiple write requests to the same cache line one after another whenever stringent ordering is warranted.

E11 ARB/MUX Virtual LSM Resolution Table Clarifications

In Section 5.1, modifications are applied to Table 47 and the Note below it, as follows:

Table 47. ARB/MUX Multiple Virtual LSM Resolution Table

Resolved Request from ARB/MUX to Flex Bus Physical Layer (Row = current vLSM[0] state; Column = current vLSM[1] state)	**Reset**	**Active**	**L1.1**	**L1.2**	**L1.3**	**L1.4**	**SLEEP_L2**
Reset	RESET	Active	L1.1	L1.2 or lower	L1.3 or lower	L1.4 or lower	SLEEP_L2
Active	Active	Active	Active	Active	Active	Active	Active
L1.1	L1.1	Active	L1.1	L1.1	L1.1	L1.1	L1.1
L1.2	L1.2 or lower	Active	L1.1	L1.2 or lower	L1.2 or lower	L1.2 or lower	L1.2 or lower
L1.3	L1.3 or lower	Active	L1.1	L1.2 or lower	L1.3 or lower	L1.3 or lower	L1.3 or lower
L1.4	L1.4 or lower	Active	L1.1	L1.2 or lower	L1.3 or lower	L1.4 or lower	L1.4 or lower
SLEEP_L2	SLEEP_L2	Active	L1.1	L1.2 or lower	L1.3 or lower	L1.4 or lower	SLEEP_L2

~~Note: Table 47 is presented as a suggestion, not a requirement.~~

E12 Text Updates in CXLCM Link Layer Chapter

In sections 4.2.8.1 and 4.2.7, make the following updates:

4.2.8.1 LLR Variables

- WrPtr: This indexes the entry of the LLRB that will record the next new flit. When an entity sends a flit, it copies that flit into the LLRB entry indicated by the WrPtr and then increments the WrPtr by one (modulo the size of the LLRB). This is implemented using a wrap-around counter that wraps around to 0 after reaching the depth of the LLRB. Certain LLCTRL flits do not affect the WrPtr. WrPtr stops incrementing after receiving an error indication at the remote entity (RETRY.Req message), until normal operation resumes again (all flits from the LLRB have been retransmitted). WrPtr is initialized to 0 and is incremented only when a flit is put into the LLRB.

  Implementation Note: WrPtr may continue to increment after receiving Retry.Req message if there are pre-scheduled All Data Flits that are not yet sent over the link. This implementation willl ensure that All Data Flits not interleaved with other flits are correctly logged into the Link Layer Retry Buffer.

- NumFreeBuf: This indicates the number of free LLRB entries at the entity. NumFreeBuf is decremented by 1 whenever an LLRB entry is used to store a transmitted flit. NumFreeBuf is incremented by the value encoded in the Ack/ Full_Ack field of a received flit. NumFreeBuf is initialized at reset time to the size of the LLRB. The maximum number of retry queues at any entity is limited to 255 (8 bit counter). Also, note that the retry buffer at any entity is never filled to its capacity, therefore NumFreeBuf is never ‘0. If there is only 1 retry buffer entry available, then the sender cannot send an ACK bearing flit which may either be a protocol flit or an LLCRD Control flit. This restriction is required to avoid ambiguity between a full or an empty retry buffer during a retry sequence that may result into incorrect operation. This implies if there are only 2 retry buffer entries left (NumFreeBuf = 2), then the sender can send an Ack bearing flit only if the outgoing flit encodes a value of at least 1 (which may be a Protocol flit with Ak bit set), else a LLCRD control flit is sent with Full_Ack value of at least 1. This is required to avoid deadlock at the link layer due to retry buffer becoming full at both entities on a link and their inability to send ACK through header flits.

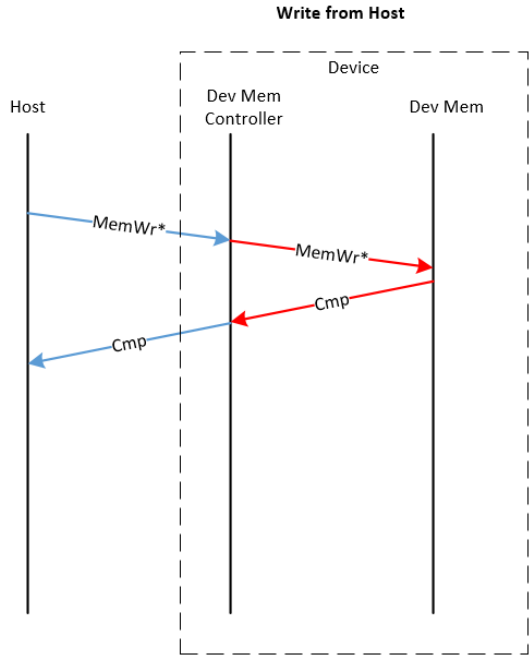
4.2.7 Link Layer Initialization

- The Tx portion of the Link Layer must wait until the Rx portion of the Link Layer has received at least one valid flit that is CRC clean before sending the LLCTRLINIT.Param flit. Before this condition is met, the Link Layer must transmit only LLCTRL-Retry flits, i.e., Retry.Frame/Req/Ack/Idle flits.

E13 Fixed Typos in Figure depicting Write from Host

Update Figure 39, Red arrows from Device Memory Controller to Device Memory should be MemWr and Cmp*

Figure 39. Write from Host



E14 S2M/D2H G5 Format Update

Updates to S2M/D2H G5 format from 3 NDR to 2 NDR messages. The global flit rules in section 4.2.5 restrict total S2M NDR to 2 total per flit which take precedence, so it is impossible to make use of 3rd NDR message in the S2M/D2H G5 format. Changing the G5 format to only include 2 NDR is done to ensure consistency and avoid confusion.

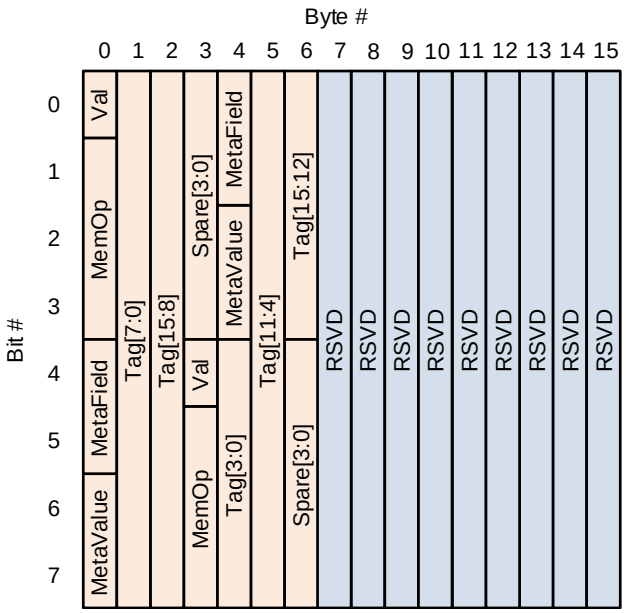
First change is to Table 40. Second change is to change G5 format in Figure 70 to include only the first 2 NDR messages converting the 3rd to RSVD bits.

Table 40. D2H/S2M Slot Formats

Format to Req Type Mapping	D2H/S2M	
	Type	Size
H0	CXL.cache Data Header + 2 CXL.cache Resp + CXL.mem NDR	85
H1	CXL.cache Req + CXL.cache Data Header	96
H2	4 CXL.cache Data Header + CXL.cache Resp	88
H3	CXL.mem DRS Header + CXL.mem NDR	68
H4	2 CXL.mem NDR	56

H5	2 CXL.mem DRS Header	80
G0	CXL.cache/ CXL.mem Data Chunk	128
G1	CXL.cache Req + 2 CXL.cache Resp	119
G2	CXL.cache Req + CXL.cache Data Header + CXL.cache Resp	116
G3	4 CXL.cache Data Header	68
G4	CXL.mem DRS Header + 2 CXL.mem NDR	96
G5	~~3~~ 2 CXL.mem NDR	~~84~~ 56
G6	3 CXL.mem DRS Header	120

Figure 70. G5 - ~~3~~ 2 S2M NDR



E15 PM timeout value.

CXL1.1 had a "TBD" value in the PM handling in section 9.4.2. We are updating this value to be 1ms with pause in timer for recovery states.

9.4.2 Compute Express Link PM Entry Phase 2

. . .

6. The Downstream Component ARB&MUX port must wait for ~~<TBD> amount of time~~ at least 1mS (not including time spent in recovery states) for a response from the Upstream Component. If no response is received from the Upstream component then the Downstream Component is permitted to abort the PM entry or retry entry into PM again.

E16 Figure 65 Title

Figure 65, which is the format for byte enable slot, incorrectly includes "S2M" in the title. CXL.mem S2M messages cannot carry byte enables as defined in table 36. Update to the title removes S2M. Note that M2S message can carry Byte Enables and is captured in Figure 52.

Figure 65. G0 - D2H~~/S2M~~ Byte Enable

E17 Implied EDS and Variable Length NULL Flits

In section 6.2.2, make the below changes. Additionally, at the end of section 6.7.1, add text and Figures 94a and 94b that show examples of NULL with EDS usage. Note, the figure numbering will be updated when this errata is integrated into the specification.

6.2.2 Protocol ID[15:0]

The 16-bit protocol ID field specifies whether the transmitted flit is CXL.io, CXL.cache/CXL.mem, or some other payload. The table below provides a list of valid 16-bit protocol ID encodings. Encodings that include an implied EDS token signify that the next block after the block in which the current flit ends is an ordered set block. Implied EDS tokens can only occur with the last flit transmitted in a data block~~; flits that cross the data block boundary cannot be associated with an implied EDS token~~.

NULL flits are inserted into the data stream by the physical layer when there are no valid flits available from the link layer. A NULL flit transferred with an implied EDS token ends precisely at the data block boundary preceding the Ordered Set block; these are variable length flits, up to 528 bits, intended to facilitate transition to ordered set blocks as quickly as possible. When 128/130b encoding is used, the variable length NULL flit ends on the first block boundary encountered after the 16-bit protocol ID has been transmitted, and the Ordered Set is transmitted in the next block. Because Ordered Set blocks are inserted at fixed block intervals that align to the flit boundary when sync headers are disabled (as described in section 6.7.1), variable length NULL flits will always contain a fixed 528 bit payload when sync headers are disabled. Please see section 6.7.1 for examples of NULL flit with implied EDS usage scenarios. A NULL flit is comprised of all zeros payload.

. . .

Table 51. Flex Bus.CXL Protocol IDs

Protocol ID[15:0]	Description
0000_0000_0000_0000	Reserved
1111_1111_1111_1111	CXL.io
1101_0010_1101_0010	CXL.io with implied EDS token
0101_0101_0101_0101	CXL.cache/CXL.mem
1000_0111_1000_0111	CXL.cache/CXL.mem with implied EDS token
1001_1001_1001_1001	NULL flit (generated by Physical Layer)
0100_1011_0100_1011	NULL flit with implied EDS token: Variable length flit containing NULLs that ends precisely at the data block boundary preceding the Ordered Set block (generated by the Physical Layer)

1100_1100_1100_1100	CXL ARB/MUX Link Management Packets (ALMPs)
0001_1110_0001_1110	CXL ARB/MUX Link Management Packets (ALMPs) with implied EDS token
All Others	Reserved

6.7.1 Control SKP Ordered Set Frequency and L1/Recovery Entry

. . .

Figure 94a illustrates a scenario where a NULL flit with implied EDS token is sent as the last flit before exiting the data stream in the case where sync hdr bypass is enabled. In this example, near the end of the 339th block, the link layer has no flits to send, so the physical layer inserts a NULL flit. Since there is exactly one flit's worth of time before the next Ordered Set must be sent, a NULL flit with implied EDS token is used. In this case, the variable length NULL flit with EDS token crosses a block boundary and contains a 528 bit payload of zeros.

Figure 94b illustrates a scenario where a NULL flit with implied EDS token is sent as the last flit before exiting the data stream in the case where 128/130b encoding is used. In this example, the NULL flit contains only a 16 bit payload of zeros.

Figure 94a. NULL Flit w/EDS and Sync Hdr Bypass Optimization

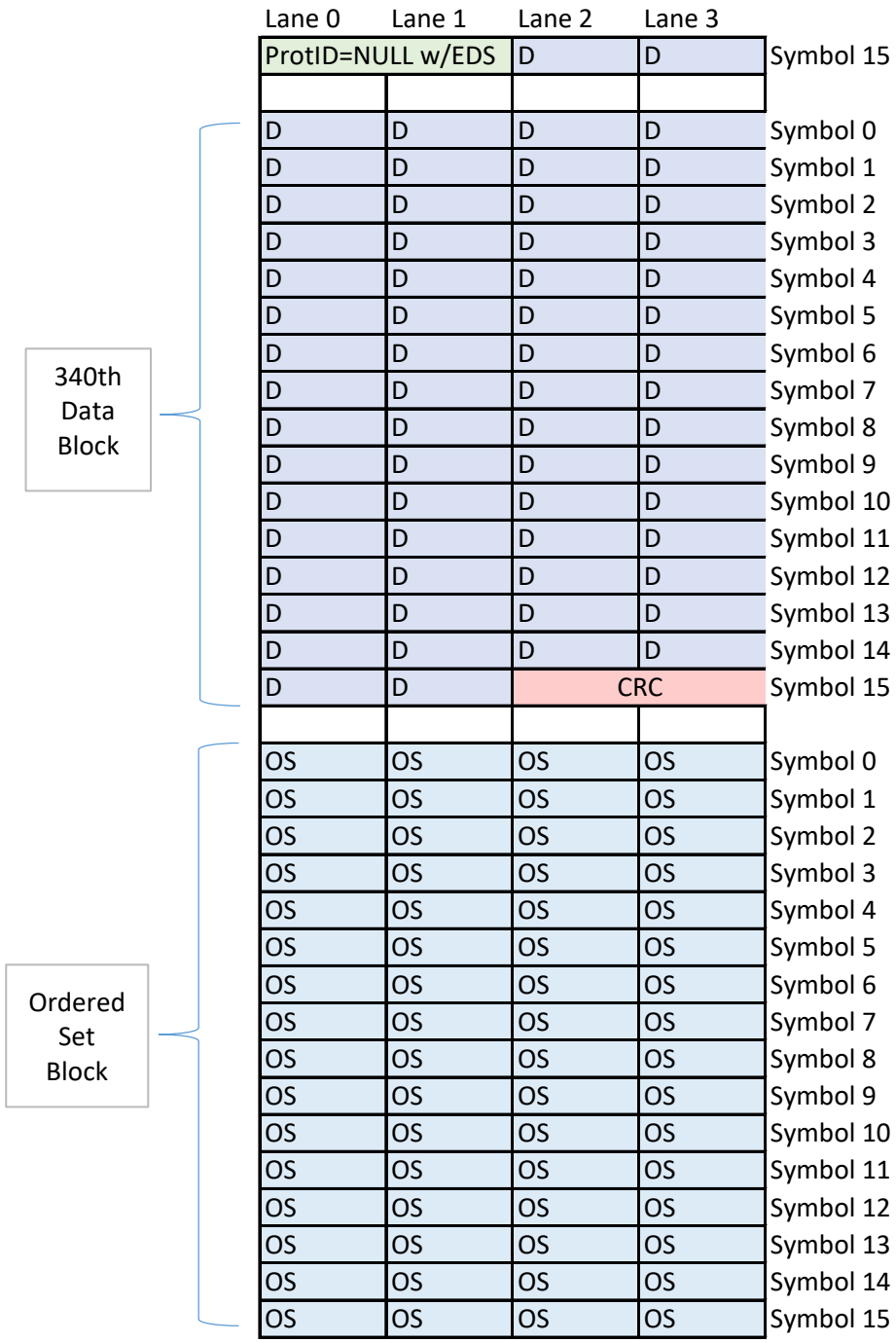
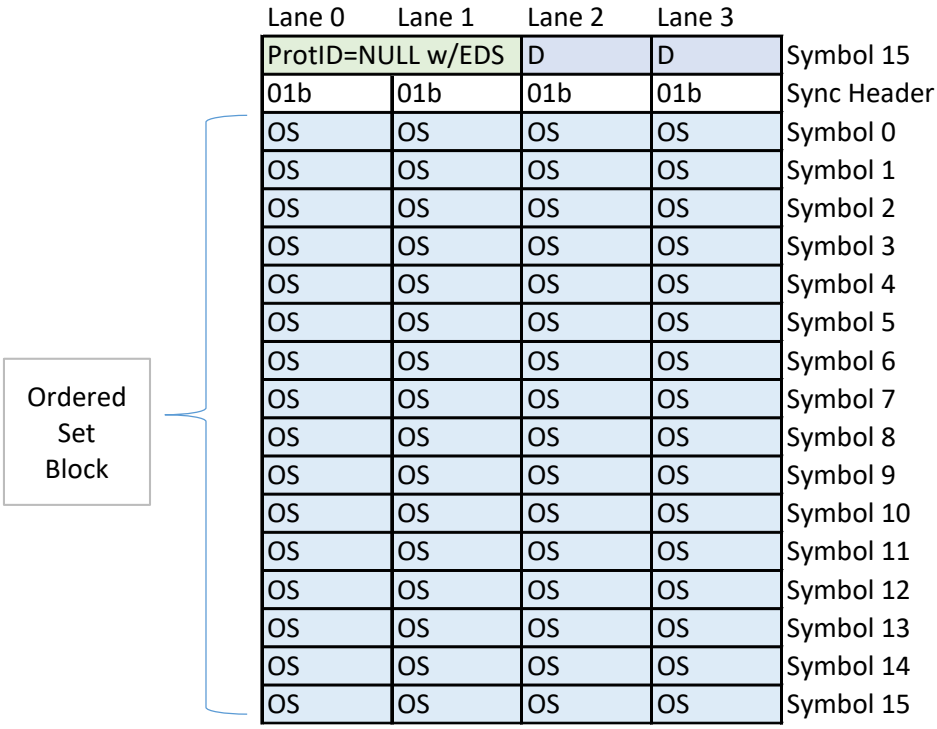


Figure 94b. NULL Flit w/EDS with 128/130b Encoding



E18 CRC bit location

Errors were found in the CRC data mask in the E18 released prior to errata version 4, so this errata item has been removed and replaced with E26.

E19 Text Updates in CXLCM Link Layer Chapter (part 2)

Additional updates to Retry flow text that is inclusive of changes discussed in E12. The additional changes are updates to description of NumFreeBuff=2 case below.

4.2.8.1 LLR Variables

- WrPtr: This indexes the entry of the LLRB that will record the next new flit. When an entity sends a flit, it copies that flit into the LLRB entry indicated by the WrPtr and then increments the WrPtr by one (modulo the size of the LLRB). This is implemented using a wrap-around counter that wraps around to 0 after reaching the depth of the LLRB. Certain LLCTRL flits do not affect the WrPtr. WrPtr stops incrementing after receiving an error indication at the remote entity (RETRY.Req message), until normal operation resumes again (all flits from the LLRB have been retransmitted). WrPtr is initialized to 0 and is incremented only when a flit is put into the LLRB.

Implementation Note: WrPtr may continue to increment after receiving Retry.Req message if there are pre-scheduled All Data Flits that are not yet sent over the link. This implementation willl ensure that All Data Flits not interleaved with other flits are correctly logged into the Link Layer Retry Buffer.

- NumFreeBuf: This indicates the number of free LLRB entries at the entity. NumFreeBuf is decremented by 1 whenever an LLRB entry is used to store a transmitted flit. NumFreeBuf is incremented by the value encoded in the Ack/Full_Ack (Ack is the protocol flit bit AK, Full_Ack defined as part of LLCRD message) field of a received flit. NumFreeBuf is initialized at reset time to the size of the LLRB. The maximum number of retry queues at any entity is limited to 255 (8 bit counter). Also, note that the retry buffer at any entity is never filled to its capacity, therefore NumFreeBuf is never `0. If there is only 1 retry buffer entry available, then the sender cannot send a~~n ACK bearing flit~~ Retryable flit. This restriction is required to avoid ambiguity between a full or an empty retry buffer during a retry sequence that may result into incorrect operation. This implies if there are only 2 retry buffer entries left (NumFreeBuf = 2), then the sender can send an Ack bearing flit only if the outgoing flit encodes a value of at least 1 (which may be a Protocol flit AK bit set), else a LLCRD control flit is sent with Full_Ack value of at least 1. This is required to avoid deadlock at the link layer due to retry buffer becoming full at both entities on a link and their inability to send ACK through header flits. This rule also creates an implicit expectation that you cannot start a sequence of "All Data Flits" that cannot be completed before NumFreeBuf=2 because you must be able to inject the Ack bearing flit when NumFreeBuf=2 is reached.

4.2.7 Link Layer Initialization

- The Tx portion of the Link Layer must wait until the Rx portion of the Link Layer has received at least one valid flit that is CRC clean before sending the LLCTRLINIT.Param flit. Before this condition is met, the Link Layer must transmit only LLCTRL-Retry flits, i.e., Retry.Frame/Req/Ack/Idle flits.

E20 Appendix A.2 fix for Bias Flip

The Appendix section A.2 was inconsistent with the rest of the spec that mandates Bias Flip to use RdOwnNoData not CLFLUSH.

- Cache flush executed using ~~CLFLUSH~~ RdOwnNoData on CXL CXL.cache protocol.

E21 Description of RSP_PRE

Fix wording on RSP_PRE which has stale wording In Table 11.

Table 11. CXL.cache - H2D Response Fields

H2D Response	Width	Description
Valid	1	The Valid field indicates that this is a valid response to the device.
Opcode	4	The Opcode field indicates the type of the response being sent. Details in Table 20
RspData	12	The response Opcode determines how the RspData field is interpreted as shown in Table 20. Thus, depending on Opcode, it can either contain the UQID or the MESI information in bits [3:0] as shown in Table 13.

RSP_PRE	2	RSP_PRE carries performance monitoring information ~~for requests that do not receive data~~. Details in Table 12
CQID	12	Command Queue ID: This is a reflection of the CQID sent with the D2H Request and indicates which device entry is the target of the response.
RSVD	1	
Total	32	

E22 CXL Downstream Port Supported PCIe Capabilities (part 2)

Additional updates to Table 60 that is inclusive of changes discussed in E3.

Table 60. CXL Downstream Port Supported PCIe Capabilities and Extended Capabilities

~~Supported~~ PCIe Capabilities and Extended Capabilities	Exceptions[1]	Notes
PCI Express Capability	Slot Capabilities, Slot Control, Slot Status, Slot Capabilites 2, Slot Control 2, and Slot Status 2 registers are not applicable.	None
PCI Power Management Capability	~~None~~ Not Applicable. Software should ignore.	None
MSI Capability	~~None~~ Not Applicable. Software should ignore.	None
Advanced Error Reporting Extended Capability	~~None~~ Not Applicable. Software should ignore.	Required for CXL devices despite being optional for PCIe. Downstream Port is required to forward ERR_ messages.

ACS Extended Capability	None	None
Multicast Extended Capability	~~None~~ Not Applicable. Software should ignore.	None
Downstream Port Containment Extended Capability	~~None~~ Use with care. DPC trigger will bring down physical link, reset device state, disrupt .cache and .mem traffic.	None
Designated Vendor-Specific Extended Capability (DVSEC)	None	Please refer to ~~section Figure~~ 7.2.1.3 for Flex Bus Port DVSEC definition.
Secondary PCI Express Extended Capability	None	None
Data Link Feature Extended Capability	None	None
Physical Layer 16.0 GT/s Extended Capability	None	None
Physical Layer 32.0 GT/s Extended Capability	None	None
Lane Margining at the Receiver Extended Capability	None	None
Alternate Protocol Extended Capability	None	None

E23 CXL Upstream Port Supported PCIe Capabilities (part 2)

Additional updates to Table 61 that is inclusive of changes discussed in E4.

Table 61. CXL Upstream Port Supported PCIe Capabilities and Extended Capabilities

~~Support~~ PCIe Capabilities and Extended Capabilities	Exceptions[1]	Notes
PCI Express Capability	None	~~N/A~~None
Advanced Error Reporting Extended Capability	~~None~~ Not Applicable. Software should ignore.	Required for CXL devices despite being optional for PCIe. Link/Protocol errors detected by Upstream Port are logged/reported via RCiEP.

~~Multicast Extended Capability~~	~~None~~	~~N/A~~
Virtual Channel Extended Capability	None	VC0 and VC1
Designated Vendor-Specific Extended Capability (DVSEC)	None	Please refer to ~~section~~ ~~Figure~~ 7.2.1.3 for Flex Bus Port DVSEC definition.
Secondary PCI Express Extended Capability	None	None
Data Link Feature Extended Capability	None	None
Physical Layer 16.0 GT/s Extended Capability	None	None
Physical Layer 32.0 GT/s Extended Capability	None	None
Lane Margining at the Receiver Extended Capability	None	None
Alternate Protocol Extended Capability	None	None

E24 Clarify Logging of Errors Detected by Upstream Port

Section 11.2.2.2 CXL Device Error Handling Flow incorrectly implied that Upstream Port is required to implement AER capability structure and log link errors there.

Errors that are not related to any specific Function within the device (Non-Function errors) are reported to the Host via PCIe error messages where they can be escalated to the platform. ~~Non-Function errors are logged in the Upstream Port RCRB in the PCIe AER Registers. In addition, t~~ The UP reports non-function errors to all RCiEPs where they are logged. Each RCiEP reports the non-function specific errors to the host via error messages. Software should be aware that even though an RCiEP does not have a software-visible link, it may still log link-related errors. At most one error message of a given severity is generated for a multifunction device. The error message must include the Requester ID of a function that is enabled to send the error message. Error messages with the same Requester ID may be merged for different errors with the same severity. No error message is sent if no function is enabled to do so. If different functions are enabled to send error messages of different severity, at most one error of each severity level is sent. ~~If a Root Complex Error Collector is implemented, errors may optionally be sent to the corresponding RCEC.~~ This error will be sent to the corresponding RCEC. Each RCiEP must be associated with no more than one RCEC.

E25 Clarification to Force LLCRD definition

Section 4.2.8.2 ACK Forcing requires expanding to also cover Crd value forcing to avoid starvation risk. Also including general clarification on how LLCRD forcing should be implemented.

Section 7.2.2.1.21 defines the control register used in section 4.2.8.2. Updating the register description to describe CRD requirements and including recommended default and minimum values.

In Section 1.2, adding "Link Layer Clock" definition to Table 1 in the Introduction because it is used by this section but not defined.

4.2.8.2 ~~ACK~~LLCRD Forcing

Recall that the LLR protocol requires space available in the LLRB to transmit a new flit, and that the sender must receive explicit acknowledgment from the receiver before freeing space in the LLRB. In scenarios where the traffic flow is very asymmetric, this requirement could result in traffic throttling and possibly even starvation.

Suppose that the A→B direction has very heavy traffic, but there is no traffic at all in the B→A direction. In this case A could exhaust its LLRB size, while B never has any return traffic in which to embed Acks. In CXL we want to minimize injected traffic to reserve bandwidth for the other traffic stream(s) sharing the link.

To avoid starvation, CXL must still permit ~~Ack~~LLCRD Control message forcing (injection of a non-traffic flit to carry an Acknowledge and Credit return), but this function is more ~~heavily~~ constrained ~~so as not to waste~~ to avoid wasting bandwidth. In CXL, when B has accumulated ~~at least 16~~ a programmable minimum number of Acks to return, B's CXL.cache/mem link layer will inject a LLCRD flit ~~for~~ to return an Acknowledge ~~return~~. The threshold of pending Acknowledges before forcing the LLCRD can be adjusted using the "Ack Force Threshold" field in the "CXL Link Layer Ack Timer Control Register".

There is also a timer-controlled mechanism to force LLCRD when the timer reaches a threshold. The timer will clear whenever an ACK/CRD carrying message is sent. It will increment every link layer clock an ACK/CRD carrying message is not sent and any Credit value to return is greater than 0 or Acknowledge to return is greater than 1. The reason the Acknowledge threshold value is specified as "greater than 1", as opposed to "greater than 0", is to avoid repeated forcing of LLCRD when no other retryable flits are being sent. If the timer incremented when the pending Acknowledge count is "greater than 0", there would be a continuous exchange of LLCRD messages carrying Acknowledges on an otherwise idle link; this is because the LLCRD is itself retryable and results in a returning Acknowledge in the other direction. The result is that the link layer would never be truly idle when the transaction layer traffic is idle. The timer threshold to force LLCRD is configurable using the "Ack or CRD Flush Retimer" field in the "CXL Link Layer Ack Timer Control Register".

It should also be noted that the ~~The~~ CXL.cache link layer must accumulate a minimum of 8 Acks to set the ACK bit in a CXL.cache and CXL.mem flit header. If ~~Ack~~ LLCRD forcing occurred after the accumulation of 8 Acks, it could result in a negative beat pattern where real traffic always arrives soon after a forced Ack, but not long enough after for enough Acks to re-accumulate to set the ACK bit. In the worst case this could double the bandwidth consumption of the CXL.cache side. By waiting for at least 16 Acks to accumulate, the CXL.cache/mem link layer ensures that it can still opportunistically return Acks in any real traffic that arrives after a forced Ack return. To avoid this issue, it is recommended that the Ack Force Threshold value be set to 16 or greater in the "CXL Link Layer Ack Timer Control Register".

It is recommended that link layer prioritize other link layer flits before LLCRD forcing.

Pseudocode for forcing function below:

```
IF (SENDING ACK CRD MESSAGE==FALSE AND (ACK TO RETURN >1 OR
CRD TO RETURN>0))

   TimerValue++

ELSE
```

```
    TimerValue=0

IF (TimerValue >=Ack Flush Retimer OR ACK TO RETURN >= Ack
Force Threshold)

    Force LLCRD = TRUE

ELSE

    Force LLCRD=FALSE
```

Note: Ack_Flush_Retimer and Ack_Force_Threshold are values that come from "CXL Link Layer Ack Timer Control Register".

7.2.2.1.21 CXL Link Layer Ack Timer Control Register (Offset 0x28)

Bit Location	Attributes	Description
7:0	RWS	**Ack Force Threshold:** This specifies how many Flit Acks the Link Layer should accumulate before injecting a LLCRD. The recommended default/minimum value is 0x10 (16 decimal). See Section 4.2.8.2 for additional details.
17:8	RWS	**Ack** or CRD **Flush Retimer:** This specifies how many link layer clockcycles the entity should wait in case of idle, before flushing accumulated Acks or CRD using a LLCRD. This applies for any case where accumulated Acks is greater than 1 or accumulated CRD for any channel is greater than 0. The recommended default/minimum value is 0x20. See Section 4.2.8.2 for additional details.

Table 1 is a list of terminology in Section 1.2. We add the following table row to that table. This term is used in multiple places.

Table 1. Terminology / Acronyms

Term / Acronym	*Definition*
Link Layer Clock	Link Layer Clock is the FLIT datapath clock of the CXL.cache/mem link layer where max frequency in this generation is 1 GHz (32GT/s * 16 lanes = 1 flit).

E26 CRC bit location - Version 2

CRC bit location described in 4.2.8.7.2 incorrectly shows CRC bits at lower bit location and they should be upper bits in the flit. Also including clarification on how Data Mask is used. Removing Figure 77 and replacing with Data Mask as plain text. This change was originally documented in E18, but errors were discovered in the data mask and the wrong original table. This erratum updates the data mask values

and updated the original table data to the correct values. Note: the original definition in CXL1.1 had the correct mask values.

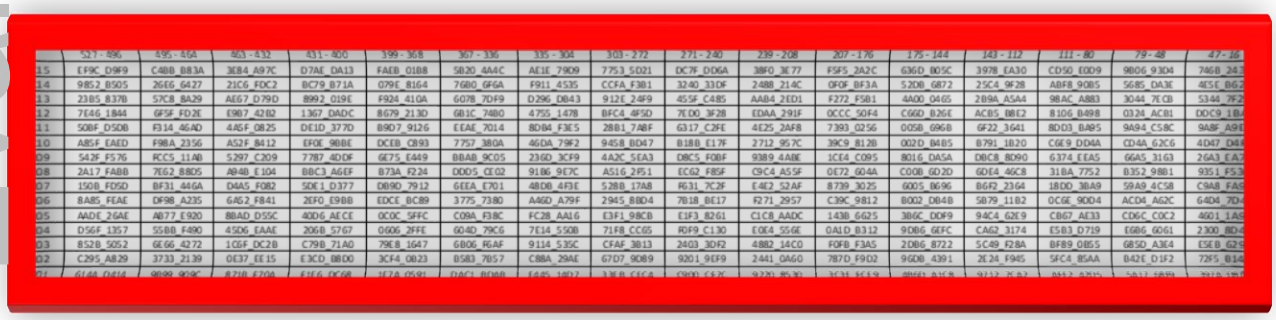
4.2.8.7.2 CRC-16 Computation

Below are the ~~384~~ 512 bit data masks for use with an XOR tree to produce the 16 CRC bits. ~~The mask bit order is CRC[N]=DM[527:016].~~ Data Mask bits [~~527:016~~511:0] for each CRC bit are applied to the Flit bits [~~527:016~~511:0] and XOR is performed.~~,~~ The resulting CRC bits are included as flit bits [~~015:000~~527:512] ~~are~~and defined to be CRC[15:00]. Pseudocode example for calculating CRC bit 15 is CRC[15] = XOR (DM[15][511:0] AND Flit[511:0]).

The Flit Data Masks for the 16 CRC bits ~~is~~are located ~~in the table~~ below~~.~~:

```
DM[15][511:0] =
512'hEF9C_D9F9_C4BB_B83A_3E84_A97C_D7AE_DA13_FAEB_01B8_5B20_4A4C_AE1E_79D9_7753_5D21_DC7F_DD6A_38F0_3E77_F5F5_2A2C_636D_B05C_3978_EA30_CD50_E0D9_9B06_93D4_746B_2431

DM[14][511:0] =
512'h9852_B505_26E6_6427_21C6_FDC2_BC79_B71A_079E_8164_76B0_6F6A_F911_4535_CCFA_F3B1_3240_33DF_2488_214C_0F0F_BF3A_52DB_6872_25C4_9F28_ABF8_90B5_5685_DA3E_4E5E_B629

DM[13][511:0] =
512'h23B5_837B_57C8_8A29_AE67_D79D_8992_019E_F924_410A_6078_7DF9_D296_DB43_912E_24F9_455F_C485_AAB4_2ED1_F272_F5B1_4A00_0465_2B9A_A5A4_98AC_A883_3044_7ECB_5344_7F25

DM[12][511:0] =
512'h7E46_1844_6F5F_FD2E_E9B7_42B2_1367_DADC_8679_213D_6B1C_74B0_4755_1478_BFC4_4F5D_7ED0_3F28_EDAA_291F_0CCC_50F4_C66D_B26E_ACB5_B8E2_8106_B498_0324_ACB1_DDC9_1BA3

DM[11][511:0] =
512'h50BF_D5DB_F314_46AD_4A5F_0825_DE1D_377D_B9D7_9126_EEAE_7014_8DB4_F3E5_28B1_7A8F_6317_C2FE_4E25_2AF8_7393_0256_005B_696B_6F22_3641_8DD3_BA95_9A94_C58C_9A8F_A9E0

DM[10][511:0] =
512'hA85F_EAED_F98A_2356_A52F_8412_EF0E_9BBE_DCEB_C893_7757_380A_46DA_79F2_9458_BD47_B18B_E17F_2712_957C_39C9_812B_002D_B4B5_B791_1B20_C6E9_DD4A_CD4A_62C6_4D47_D4F0

DM[09][511:0] =
512'h542F_F576_FCC5_11AB_5297_C209_7787_4DDF_6E75_E449_BBAB_9C05_236D_3CF9_4A2C_5EA3_D8C5_F0BF_9389_4ABE_1CE4_C095_8016_DA5A_DBC8_8D90_6374_EEA5_66A5_3163_26A3_EA78

DM[08][511:0] =
512'h2A17_FABB_7E62_88D5_A94B_E104_BBC3_A6EF_B73A_F224_DDD5_CE02_91B6_9E7C_A516_2F51_EC62_F85F_C9C4_A55F_0E72_604A_C00B_6D2D_6DE4_46C8_31BA_7752_B352_98B1_9351_F53C

DM[07][511:0] =
512'h150B_FD5D_BF31_446A_D4A5_F082_5DE1_D377_DB9D_7912_6EEA_E701_48DB_4F3E_528B_17A8_F631_7C2F_E4E2_52AF_8739_3025_6005_B696_B6F2_2364_18DD_3BA9_59A9_4C58_C9A8_FA9E

DM[06][511:0] =
512'h8A85_FEAE_DF98_A235_6A52_F841_2EF0_E9BB_EDCE_BC89_3775_7380_A46D_A79F_2945_8BD4_7B18_BE17_F271_2957_C39C_9812_B002_DB4B_5B79_11B2_0C6E_9DD4_ACD4_A62C_64D4_7D4F

DM[05][511:0] =
512'hAADE_26AE_AB77_E920_8BAD_D55C_40D6_AECE_0C0C_5FFC_C09A_F38C_FC28_AA16_E3F1_98CB_E1F3_8261_C1C8_AADC_143B_6625_3B6C_DDF9_94C4_62E9_CB67_AE33_CD6C_C0C2_4601_1A96

DM[04][511:0] =
512'hD56F_1357_55BB_F490_45D6_EAAE_206B_5767_0606_2FFE_604D_79C6_7E14_550B_71F8_CC65_F0F9_C130_E0E4_556E_0A1D_B312_9DB6_6EFC_CA62_3174_E5B3_D719_E6B6_6061_2300_8D4B

DM[03][511:0] =
512'h852B_5052_6E66_4272_1C6F_DC2B_C79B_71A0_79E8_1647_6B06_F6AF_9114_535C_CFAF_3B13_2403_3DF2_4882_14C0_F0FB_F3A5_2DB6_8722_5C49_F28A_BF89_0B55_685D_A3E4_E5EB_6294

DM[02][511:0] =
512'hC295_A829_3733_2139_0E37_EE15_E3CD_B8D0_3CF4_0B23_B583_7B57_C88A_29AE_67D7_9D89_9201_9EF9_2441_0A60_787D_F9D2_96DB_4391_2E24_F945_5FC4_85AA_B42E_D1F2_72F5_B14A

DM[01][511:0] =
512'h614A_D414_9B99_909C_871B_F70A_F1E6_DC68_1E7A_0591_DAC1_BDAB_E445_14D7_33EB_CEC4_C900_CF7C_9220_8530_3C3E_FCE9_4B6D_A1C8_9712_7CA2_AFE2_42D5_5A17_68F9_397A_D8A5

DM[00][511:0] =
512'hDF39_B3F3_8977_7074_7D09_52F9_AF5D_B427_F5D6_0370_B640_9499_5C3C_F3B2_EEA6_BA43_B8FF_BAD4_71E0_7CEF_EBEA_5458_C6DB_60B8_72F1_D461_9AA1_C1B3_360D_27A8_E8D6_4863
```

~~Figure 77. CRC Data Mask for 527 bit Flit~~



E27 CXL Vendor ID

The CXL Vendor ID assigned by the PCI SIG is 1E98h and replaces the 8086h value used in the CXL rev1.1 and prior specifications. This value should be used for the CXL vendor ID value in CXL Power Management Messages in section 3.1.2 and Figure 13, in Modified TS1/TS2 Ordered Sets for CXL Alternate Protocol Negotiation in Table 53, in the Flex Bus Device DVSEC in Table 58, in the Flex Bus Port DVSEC in Table 62, and all other relevant CXL vendor ID references such as in Chapter 14. A

subsequent errata will show the exact locations affected by this change. Additionally, the legal notice at the beginning of the specification document (after the title page) is amended as noted below.

The copyright in this Specification is owned by the CXL Consortium~~as a Contractual SIG~~. Use of this Specification and any related intellectual property (collectively, the "Specification"), is governed by the terms of the Promoter, Contributor or Adopter Agreement (collectively, "Agreement") as available by request from admin@computeexpresslink.org. This specification may only be used by (a) a party to the Agreement, and in compliance with its terms, or (b) non-parties to the Agreement who have accepted an approved disclaimer of liability and acknowledgement of IP protections.

Any use or implementation of this specification is expressly subject to the terms and conditions described in the CXL Promoter, Contributor, and/or Adopter Agreement, as appropriate for your status. The implementation or use of industry specifications made available by any other standards body, such as PCIe, if referenced in a CXL Final Specification (as such term is defined in those agreements) is not governed by CXL policies and procedures, and may be subject to patent, copyright, and other intellectual property right protection (collectively "IPR"). Licensing of such IPR in a referenced industry specification shall be governed by the policies and procedures of the organization promulgating such industry specification.

This document is provided 'as is' with no warranties, express or implied, including any warranty of merchantability, fitness for a particular purpose, noninfringement of any third-party intellectual property rights, or any warranty otherwise arising out of any proposal, specification or sample. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted herein.

CXL is a trademark of the CXL Consortium~~Contractual SIG~~

All other product names are trademarks, registered trademarks, or servicemarks of their respective owners.

NOTICE TO USERS: THE UNIQUE VALUE THAT IS PROVIDED IN THIS SPECIFICATION FOR USE IN VENDOR DEFINED MESSAGE FIELDS, DESIGNATED VENDOR SPECIFIC EXTENDED CAPABILITIES, AND ALTERNATE PROTOCOL NEGOTIATION ONLY AND MAY NOT BE USED IN ANY OTHER MANNER, AND A USER OF THE UNIQUE VALUE MAY NOT USE THE UNIQUE VALUE IN A MANNER THAT (A) ALTERS, MODIFIES, HARMS OR DAMAGES THE TECHNICAL FUNCTIONING, SAFETY OR SECURITY OF THE PCI-SIG ECOSYSTEM OR ANY PORTION THEREOF, OR (B) COULD OR WOULD REASONABLY BE DETERMINED TO ALTER, MODIFY, HARM OR DAMAGE THE TECHNICAL FUNCTIONING, SAFETY OR SECURITY OF THE PCI-SIG ECOSYSTEM OR ANY PORTION THEREOF (FOR PURPOSES OF THIS NOTICE. 'PCI-SIG ECOSYSTEM"' MEANS THE PCI-SIG SPECIFICATIONS, MEMBERS OF PCI-SIG AND THEIR ASSOCIATED PRODUCTS AND SERVICES THAT INCORPORATE ALL OR A PORTION OF A PCI-SIG SPECIFICATION AND EXTENDS TO THOSE PRODUCTS AND SERVICES INTERFACING WITH PCI-SIG MEMBER PRODUCTS AND SERVICES).